Cost Aware Decision Support for Cloud Data Centres

Sergej Svorobej BSc

This thesis is submitted in accordance with the requirements of Dublin City University for the degree of Doctor of Philosophy

Supervisors: Dr. James Byrne
Prof. PJ Byrne

Submitted Dublin City University, August 2019

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: _____ (Candidate)

Sergej Svorobej

ID No.: 13212938

Date: _____

Table of Contents

List of Abbreviations

DES	Discrete Event Simulation
DVFS	Dynamic Voltage and Frequency Scaling
ICT	Information and Communications Technology
CAGR	Compound Annual Growth Rate
FIFA	Fédération Internationale de Football Association
VM	Virtual Machine
QoS	Quality of Service
TCO	Total Cost of Ownership
SLA	Service Level Agreement
ERP	Enterprise Resource Planning
EBP	Enterprise Buyer Professional
NIST	National Institute of Science and Technology
CPU	Central Processing Unit
RAM	Random Access Memory
HDD	Hard Disk Drive
SSD	Solid State Drive
PDU	Power Distribution Unit
I/O	Input Output
OS	Operating System
IaaS	Infrastructure as a Service
DaaS	Data storage as a Service
XaaS	Everything as a Service
IBM	International Business Machines
QRE	QoS aware Resource Elasticity
CAPEX	Capital Expenditure
OPEX	Operational Expenditure
KVM	Keyboard, Video and Mouse
MTTF	Mean Time to Failure
MAPE-K	Monitor-Analyse-Plan-Execute over a shared Knowledge
FFD	First Fit Decreasing
UMA	Utilisation-based Migration Algorithm
MT-PerfMod	Multi-Tier Performance Model

MT-ResElas	Multi-Tier Resource Elasticity
EU	European Union
FP7	7th Framework Programme
CACTOS	Context-Aware Cloud Topology Optimisation and Simulation
FCO	Flexiant Cloud Orchestrator
KVM	Kernel-based VM
API	Application Program Interface
IOPS	I/O Operations Per Second
MIPS	Million Operations Per Second
RAPL	Average Power Limit
HDFS	Hadoop Distributed File System
PM	Physical Machine
DVFS	Dynamic Voltage and Frequency Scaling
EMF	Eclipse Modelling Framework
PCM	Palladio Component Model
RD-SEFF	Resource Demanding Service Effect Specification
PMS	Palladio Measurement Specification
PRM	Palladio Runtime Model
MDSD	Model-Driven Software Development
PDCM	Physical Data Centre Model
LDCM	Logical Data Centre Model
PLM	Physical Load Model
LLM	Logical Load Model
CDO	Connected Data Objects
VMI	Virtual Middleware Integration
KPI	Key Performance Indicators
VoP	Voice over IP
NIC	Network Interface Card
UI	User Interface
PSU	Power Supply Unit
ERP	Enterprise Resource Planning
HR	Human Resources
IDE	Integrated Development Environment

PUE	Power Usage Effectiveness
NAS	Network Attached Storage
TDP	Thermal Design Power
ROI	Return on Investment
SME	Small and Medium Enterprise

List of Figures

List of Tables

Abstract

Thesis Title: Cost aware decision support for cloud data centres

Author: Sergej Svorobej

Cloud computing became a popular IT and business trend in recent years, due to its scalability, reliability, ease of access and low costs. The demand for cloud services has led to a steep increase in number and size of data centres around the world with more hardware dense, large-scale data centres being build. A modern data centre has become a very complex system with a multitude of interconnected hardware resources that is hard to manage in an autonomic and cost-effective manner.

This thesis presents novel techniques, models and software for estimating the impact of data centre planning decisions on Total Cost of Ownership. The use of a Discrete Event Simulation (DES) framework as a decision support tool is proposed for understanding operational efficiencies and the financial implications of different cloud resource management techniques and different hardware components.

Based on a detailed assessment of the current state of cloud simulation tools, resource management techniques and cost models available for data centre management, an on-the-fly compilation of simulation models for large-scale cloud data centres has been proposed and delivered. This enables an automated approach for extracting all required simulation modelling parameters from existing data centre monitoring tools, improving speed and usability of the simulation approach. A unique integration approach between a production grade cloud optimisation framework and an offline simulation analysis framework, enables direct simulation of cloud resource management policies allowing for experimentation of different “what-if” scenarios using existing production tooling. An improved set of cost models were developed alongside the simulation framework which are capable of calculating costs based on time series data produced by the simulation framework. Thus, taking into account resource management policy effects and enabling a more granular overview of costs when compared to traditional TCO calculation approaches. The simulation framework implementation was validated using real case study data. In addition, the practical use of the proposed cloud data centre decision support approach is demonstrated through presentation of an extended case example, estimating and planning for the impact of a data centre hardware upgrade and resultant system performance and costs.

"When the spirits are low, when the day appears dark, when work becomes monotonous, when hope hardly seems worth having, just mount a bicycle and go out for a spin down the road, without thought on anything but the ride you are taking." –

Sir Conan Doyle, Jan 1896

Acknowledgement

I would like to express my sincere gratitude to both Dr. James Byrne and Prof. PJ Byrne for their supervision, guidance, dedicated time and support over the past number of years.

In addition, I would like to thank to both of my external examiners, Dr Brendan Jennings of Waterford Institute of Technology and Dr. Peter Williams of University of Limerick, and my internal examiner Dr. Martin Crane, for all their valuable and constructive feedback for this thesis.

Also, I want to especially thank my wife, Malgorzata Soltys, for all of her patience, forgiveness, love and support which has been invaluable on this journey.

A huge thank you for the help, knowledge and comradery goes to all my former colleagues from the CACTOS FP7 consortium, with a special gratitude to: Christian Stier, Henning Groenda, Jörg Domaschka, Christopher Hauser, Stefan Wesner, P-O Östberg, Jakub Krzywda, Zafeirios Papazachos and Dimitrios Nikolopoulos. I have learned a great deal by working by their side.

I would like to express my appreciation to all of my friends Xiaoning Liang, Martina Brophy, Ross Chapman, Peter Deeney,Sebastian Lehrig, Anna Gourinovitch, Szu-Hsin Wu, Huanhuan Xiong, Colette Real and Pooyan Jamshidi for their help, advice, a shoulder to cry on and joy of human interaction, making the past years bearable.

I am very grateful to my colleagues in DCU who I can also call friends for their great support over the years, including Theo Lynn, Margaret Galuszynska, Janine Bosak, Muriel Keegan, Frank Fowley and Teresa Hogan. For all the encouragement to not give up and keep on fighting, for all those 3rd floor corridor conversations.

I am grateful to DCU Business School and Irish Centre for Cloud Computing and Commerce (IC4) for providing financial support of the work.

1 Introduction

In the past half century, significant technological progress has been made in the ICT sector (Buyya, Yeo, *et al.*, 2009). Information Technology has proven to be of significant value in both scientific and commercial settings, with increasing world-wide connectivity, information exchange and data processing speeds. The cloud computing paradigm has become a part of a technological and business revolution with continuously growing worldwide demand for cloud services and resources. The demand for cloud computing is predicted to continue to grow over the years and stay relevant to the cloud consumer and cloud providers forming an expanding multibillion euro cloud ecosystem (Arizton, 2018). Global cloud IP traffic is predicted to increase from 1.5 ZB in 2017 and reach 4.8 ZB by 2022 (Cisco, 2018). This growth in demand translates directly to a significant increase in revenue for the cloud sector, where the worldwide cloud computing market is forecasted to reach €261 billion by 2021 (Gartner Inc., 2018). As the demand for cloud computing grows, so too does the cloud data centre market which is predicted to grow at a 28.7% Compound Annual Growth Rate, reaching a total of €58.21 billion by 2023 (Infoholic Research LLP, 2017).

From a business perspective the growth in demand is driven by the key advantages of cloud computing, including for example, significant reduction in costs, fast access to hardware resources with no upfront capital investment, on-demand scalability and new application architecture possibilities (Marston, Li, *et al.*, 2011). From a technical perspective the term “cloud computing” describes a pool of shared computing hardware resources that can be easily configured to run services which are accessible to users remotely via a network connection (Mell and Grance, 2011). Data centres provide facilities where computing hardware running cloud services are physically located. As such, data centres can be described as dedicated premises that host IT hardware equipment in a safe and operational manner. The nature of demand for cloud services requires 24/7 uninterrupted accessibility all year round. To meet such demands, a data centre is built in such a way so as to provide a constant failsafe power supply to the equipment, internet backbone access for fast connectivity, and cooling solutions to dissipate heat produced by the equipment in active use (Geng, 2014). Resources working at a data centre must ensure equipment is always running as intended, fixing any failures that might occur over time, decommission old equipment while provisioning

and installing new equipment as required. The size of a data centre varies depending on its purpose and location limitations. One of the world's largest data centres was recently built on the Citadel Campus in Northern Nevada, and is set to be 668,902 m^2 with top power consumption of 650,000 kW (PRNewswire LLC, 2017). To put these figures into perspective and to understand its scale, this data centre is the size of approximately 93.7 football fields (FIFA, 2016), and just one hour operating at its top capacity will consume the same amount of power as 60 average households would consume in one year (EIA, 2015). Increases in user demand for cloud services have resulted in a consequential rise in the number of data centres around the world as well as to their size. Efficient management of a data centres resources while coping with increasing demand, becomes a challenge at such scale. It is also recognised that from an end user perspective ongoing cost reduction is one of the most critical success factors for cloud computing adoption (Astri, 2015). Thus, modern data centres are expected to deliver high levels of service while maintaining cost competitiveness on a consistent basis. As a consequence, data centres are going through continual evolution, undergoing between three and four major equipment changes every ten years (Mehmi, Sangal, et al., 2016).

1.1 Cloud resource management challenges

As already alluded to, the size and scale of cloud data centres is increasing over time, due to the continual increase in demand for such services. This scale is leading to an increase in the complexity of cloud architecture and to an increase in operational costs (Rimal, Jukan, *et al.*, 2011). To control the delivery of vast cloud resources, a cloud computing provider uses autonomic mechanisms of resource management available through resource virtualisation properties, such as resource provisioning, resource allocation and resource adaptation.

Resource provisioning is defined as the resource assignment to a customer; resource allocation refers to an economic distribution of resources among competing customers; and resource adaptation is defined as an ability to dynamically adjust provided resources based on user requirements (Manvi and Krishna Shyam, 2014). Efficiency and robustness of resource management policies play a significant role in cloud business models affecting Quality of Service (QoS), energy efficiency, carbon dioxide emissions and operational cost. The importance of these data centre dimensions

is evidenced by the large body of research that has developed in these domains over the last decade.

Cost savings was identified as the most important factor behind adoption of cloud technology by end users, with increasing demands for lower priced access to cloud services (Astri, 2015). Hence, cloud infrastructure providers are under continuing pressure to minimise data centre capital and operational costs in order to stay competitive. However, ensuring expected QoS while keeping costs low in a large scale dynamic cloud data centre is a challenging task, with QoS equating to appropriate resource delivery which meets user required demand. The QoS specification for an end user will vary depending on the needs of different customers and is usually specified in an agreed upon Service Level Agreement (SLA). An SLA is part of a binding contract between a cloud infrastructure provider and a client and guarantees a range of expected technical capabilities. Such capabilities usually include network throughput, service availability, processing capabilities, scalability and security standards defining clauses (Singh and Chana, 2015a). An SLA breach can result in reparation costs for the cloud service provider and in severe cases, to a decrease in trust between the two parties, leading to potential business loss (Mistry, Bouguettaya, *et al.*, 2015).

To address these issues, cloud resource management mechanisms have been developed to monitor system QoS, so as to avoid SLA violations by ensuring the resources requested by a user are ready and available to meet user demand at the required time. However, inefficiency in resource management actions can lead to resource overprovisioning which reduces the amount of resources available for the cloud infrastructure provider to sell while also negatively impacting energy efficiency. Due to the scale and complexity involved in combination with dynamic user workloads, developing and fine-tuning a resource management strategy, that is capable of maintaining the desired QoS for a cloud computing system, in a cost-efficient manner is an intricate challenge, and one that has led to a significant body of research to date (e.g Suresh and Varatharajan, 2017; Makrani, Sayadi, *et al.*, 2018; Nadjaran Toosi, Sinnott and Buyya, 2018; Ramchand, Chhetri and Kowalczyk, 2018; Zakarya and Gillam, 2018).

Data centre design plays an important role in achieving an efficient balance between QoS and cost of operation. Cost calculations go beyond operational expenses, underpinning business economic viability including data centre building construction, equipment acquisition, equipment upgrades and cloud resource management strategy.

The main decisions that impact costs are the choice of geographical location, size, design, equipment, number of staff and cloud resource management efficiency (Barroso and Hölzle, 2013). Geographic location dictates the cost of land, construction labour and energy to run the data centre after it is built. Size and design decisions go hand in hand in determining the space ratios for server hosting, cooling and power supply premises. Energy consumption properties of equipment for the server room are important considerations for cost recovery through power saving. However, higher capital expenditure costs (CAPEX) associated with some equipment might not justify the benefits of related energy savings. Cloud data centre cost analysis capabilities can reveal the trade-offs and financial benefits behind different design opportunities. Understanding cost implications help in supporting day-to-day management tasks of cost-effective and competitive business enterprise (Simonet, Lebre and Orgerie, 2016).

This associated high demand for cloud computing and competitor action drive innovation in server hardware equipment design for the hosting of cloud services. New equipment is continually being developed, which for the most part is improving in terms of functionality, capacity and energy efficiency. This constant evolution becomes a challenge for cloud data centre maintenance decisions. A typical data centre servers lifetime is short and spans approximately 3 to 4 years after which time the equipment is considered depreciated and requires upgrading (Barroso and Hölzle, 2013). Upgrade decisions can result in additional cost and if not managed correctly can unknowingly have a negative impact on the existing pool of resources by making existing resource management policies less efficient (Zakarya and Gillam, 2018). Partial server hardware upgrades and data centre expansion through addition of new server equipment creates equipment diversity leading to resource pool heterogeneity. Recent studies of well-known public cloud providers show evidence of significant performance diversity between two VMs deployed in the same data centre due to hardware heterogeneity, leading to increasing concerns over QoS among its customers (Xu, Liu, *et al.*, 2016). Knowing the changes hardware maintenance decisions can have on data centre QoS and cost is a favourable advantage. However, it can be a challenging task due to the large scale dynamic nature of cloud computing.

Given the complexity and importance of resource management, understanding system wide impact is of paramount importance, prior to deployment in the production environment. Due to this complex nature, simulation-based analysis is often used to

support decision making in the designing of appropriately balanced resource distribution policies that meet the specific system performance goals of a specific enterprise (Sinha and Shekhar, 2015). Simulation-based experimentation helps to evaluate and rectify any inconsistencies resource management algorithms may have at design stage, before real system deployment. Simulation-based modelling is realised as a computer-based depiction of the cloud system generated through modelling techniques. This computer-based model captures all cloud computing data centre parameters and all relationships between the relevant entities in the cloud system. Simulation-based approaches are advantageous over real world test bed scenarios as they enable full-scale cloud data centre system experimentation at a lower cost and faster pace than real time (Makaratzis, Giannoutakis and Tzovaras, 2018). Existing simulation tools provide estimations of cloud resource utilisation such as CPU, memory storage and network combined with power consumption calculations based on the utilisation levels. However, research conducted to date shows a lack of automation in system modelling, leading to issues in large simulation scenario creation. In addition, cloud data centre cost calculation remains largely static by not considering the impact of resource management and to date has tended to focus almost exclusively on energy costs.

Costs associated with service delivery are an essential element in all data centre development and change initiatives with a variety of different costing methods utilised. One approach that is particularly prevalent is the Total Cost of Ownership (TCO), method for calculating the cost of owning and deploying data centre equipment when making decisions around IT investments (Ramchand, Chhetri, *et al.*, 2018). Constant innovation in ICT leads to cloud data centres continually evolving in the form of new hardware functionality, cloud service consumption and resource management approaches. Making executive decisions on adjusting resource management policies and equipment upgrade procurement requires insights not only on technical hardware capabilities, but also on the financial implications, which when combined can be used to identify the most appropriate investment options (Geng, 2014).Tools for calculating TCO are composed of a mix of approaches, with some researchers delivering implementation frameworks for calculating costs (Hardy, Sideris, *et al.*, 2011), while others only provide mathematical formulas for calculating different utility costs (Simonet, Lebre, *et al.*, 2016). Such formulas can then be used as part of an implementation or simply as part of additional spreadsheet calculations.

Available TCO calculation approaches focus on a variety of direct and indirect costs such as server and cooling equipment, real estate, maintenance and consumed energy (Simonet, Lebre, *et al.*, 2016). Research conducted to date has not identified TCO estimation procedures which have the capability to take into account specific user workload demand, server hardware capability required to process this workload and compute resource management decisions for dynamic resource allocation. The absence of a direct link between TCO calculations and cloud data centre performance parameters significantly limit the decision-making ability of data centre managers, with the primary focus on only one component, direct and indirect costs. Although, some research suggests that cloud simulation and modelling tools through energy consumption can be used for TCO estimations (Stier, Groenda and Koziolek, 2014) the literature review conducted in this study shows a lack of defined methodology and implementation of such a solution. It is this thesis's proposition that greater cohesion in analysing data centre performance and TCO will lead to more informed decision making. By creating a holistic overview of data centre performance and costs, every change in design can be directly evaluated, explored and reasoned upon, providing decision support based on the QoS and profitability of an enterprise.

1.2 Problem statement

The scope of the thesis is focused on data centre management which includes the costing of property, hardware and provisioned resources and the delivery of appropriate service quality. The challenges of cost and operational efficiency are addressed from the perspective of an infrastructure provider, thus the primary focus of this thesis is on an Infrastructure-as-a-Service (IaaS) business model. Within this study, other cloud computing business models, specifically Software-as-a-Service (SaaS) and Platform-as-a-Service (PaaS), which are not managed by an infrastructure owner necessarily, are considered as application sets deployed to IaaS. This viewpoint enables a direct focus on the specific challenges and opportunities of cloud datacentre infrastructure providers rather than on platform or software service issues, consequentially narrowing the scope of resource allocation processes.

Cloud computing is widely used by businesses all around the world as it delivers on-demand rapid access to a virtually endless pool of compute resources and services. Growth in demand for cloud computing has resulted in growth in the number and size of

data centres globally, creating complex large scale interconnected hardware systems. Competition and low-cost expectation have driven the need for cloud infrastructure providers to manage data centre infrastructure in a lean and effective way, reducing costs of capital and operational expenditures. Hence, cloud infrastructure providers employ resource management frameworks to strive to ensure QoS and SLAs are met, costs are reduced, and profit is increased.

Although a large body of research exists in the area of cloud resource management for IaaS providers, an analysis of the literature in combination with case study discussions with companies reveals a real lack of understanding in relation to the impact of resource management decisions on data centre costs. IaaS resource management policies have the ability to directly influence QoS, resource utilisation and energy consumption of the system, hence affecting operational costs including server hardware selection. However, cloud infrastructure providers presently do not have adequate tools to identify and calculate the dependencies between hardware maintenance, resource provisioning policies, daily resource demand and costs.

Cloud infrastructure providers are under immense pressure to keep resource prices at an affordable and competitive range, reduce CO_2 footprint, reduce costs and increase profitability. This means creating and running lean, a reliable enterprise which is capable of fine tuning resource distribution policies to achieve maximum profit while keeping clients satisfied through QoS compliance. Because of these low-cost expectations from end users, confidence in cost estimation is important. However, cloud data centres are an intricate system of interconnected hardware at a large scale where interdependencies are hard to understand and quantify. Incorrect cost estimation can result in resource under provisioning which can deteriorate QoS, or conversely resource under provisioning can lead to revenue loss. This is in essence a balancing act and one that requires advanced decision support for optimal performance. In an ever-evolving market, there is an expectation of 24/7 cloud infrastructure availability, with many business-critical activities included. Strict SLAs are in place to enforce accountability of cloud resource providers in such instances, including in some cases severe monetary penalties should these SLAs be breached. Therefore, it is essential for business success that a data centres resources are provisioned and priced correctly.

This thesis looks at the data centre resource management1 challenges of infrastructure providers who are not in control of the cloud applications and higher level cloud services that are deployed on top of IaaS. In other words, the infrastructure provider only leases virtualised hardware resources and is not aware of which type of application is deployed by the IaaS user. This assumption feeds into the scope of this research by focusing on the challenges of managing data centre resources in a cost-efficient way, whilst ensuring the QoS for any cloud based application deployed on the provided cloud infrastructure.

To fully understand the range of costs involved and the operational performance of a cloud data centre, a holistic approach needs to be adopted, which takes into account resource demand and management policies and their impact on QoS. Currently, available cost analysis tools are not capable of comprehensively reflecting the level of cloud data centre complexity so as to ensure that the full range of system dependencies are considered in cost calculations. The scale of cloud data centre operations and the fast pace of innovation in the ICT sector also requires a consequential scaled improvement in cost/performance calculation speed and accuracy. The following hypothesis was formulated based on the arguments made in this thesis:

"*The use of a modelling framework integrated with a data centre management environment can enable infrastructure providers to understand better the interdependencies between costs and performance, leading to an enhanced cost-aware decision support for cloud data centres.*"

In summary, given the scale and complexity of cloud data centres it is difficult to determine the relationship between costs, resource management and QoS. The demand in IaaS is driven primarily by the low-cost of service, which puts pressure on infrastructure providers to remain competitive. However, there is a lack of tools available that help to make cost-aware decisions for modern cloud data centres to manage the workload, save energy consumption, and ensure adequate quality of service. Based on this, the thesis research objectives are to:

[1] For the purposes of this thesis, data centre resources refer to hardware, energy, real estate and employees as the scope of this work extends to the overall costs of data centres.

1. Determine the requirements for such a tool.
2. Develop an execution framework.
3. Establish the implications of tool usage for real world cost-aware decision making.

This thesis describes the development of an improved methodology and toolset for more precise and complete cost estimation of cloud data centre, while taking into account associated performance. In completing this research, a number of real world case studies are utilised for i) problem definition; ii) data extraction and iii) framework deployment. This research was undertaken as part of a European 7th Framework project, CACTOS. Details of the project and associated case organisations is presented in more detail in Chapter 3.

1.3 Thesis approach

As already alluded to, the research in this thesis sets out to present a better understanding of cloud data centre management decisions and their associated impact on costs. A cloud data centre is viewed as a large, complex system with a multitude of interconnected elements. It is proposed that to better address these cloud data centre development decisions, cost estimations require a more detailed assessment which is linked more explicitly with system performance. A simulation paradigm is proposed as a solution methodology which allows for modelling such complex systems and delivering an improved degree of TCO accuracy together with an in-depth analysis on system behaviour.

To complete the proposed research, a deep understanding of the inner technical workings of a cloud data centre is required, as well as an understanding of associated resource management techniques and cost association and calculation approaches. In addressing these research tasks, the following research challenges and descriptors outline the main direction of the work presented in the remainder of the thesis:

Domain positioning – To address the problem, the presented work traverses across a number of research areas, with the main areas being cloud computing, resource management, simulation, and cost estimation. At the outset it is paramount to build a clear knowledge foundation for each research domain. A literature review and industry engagement activities were used to acquire expertise and compile requirements which

fed into solution development. Part of the process was to identify key components, which was then used to form a model which captures the core values of a cloud computing systems behaviour relevant for the research objective.

Model development - The current study was concerned with a cloud data centre system model that supplies the simulation platform with all the necessary data to make estimations on the system behaviour over the time of the simulation experiment. The system model can be further divided into two concepts, which include a meta-model and model implementation. A meta-model can be described as a schema, or data map showing what data is located within a model tree and the data type. A data structure is enforced using a meta-model and determines where exactly to write or read certain information about the system e.g. id of a rack or frequency of a CPU. The model implementation is a method chosen to store system model data which complies with a chosen meta-model design. The model development challenge is set to focus on defining the type of data required for the simulation i.e. meta-model design. It is important to identify the scope of the simulation output in order to successfully identify an essential data set for simulation-based estimations. As a first step in the research a list of feasible functional requirements need to be derived from the potential use cases in the project. Once the list is compiled the data can be linked to address each of the requirements. A cloud data centre is a data rich environment that provides an ample amount of metrics, therefore it is important to ensure that only required data is included in the meta model so as to avoid wasting effort on collection of data unused in the estimations. Model development focuses on the technical data capturing hardware and virtual layer properties, as well as, accounting for data type attributes for TCO calculations. Once the meta-model design is finalised, the next challenge is to acquire data and create a model implementation.

Data acquisition - The nature of a simulation model for data centre evaluation, requires that it be a self-contained full set of data describing all system attributes and relationships between components, a partial system model is not sufficient for a simulation experiment. The large-scale of cloud data centres makes it difficult to collect whole data for the purpose of population of a valid simulation model, also making manual model creation a less viable option. The data acquisition challenge addresses the issue by looking at an integration mechanism between the simulation framework platform and a large-scale resource monitoring and cloud data centre metric data collection

framework. Cloud data centre telemetry monitoring tools are widely used for probing hardware health status and performance. By tapping into available telemetry data flows a data centre model for simulation can be created programmatically. However, such an operation requires an appropriate technical model implementation method capable of being continuously and asynchronously updated to save network traffic, ensuring that the latest data is available in the model on demand. Model compression and fragmentation are also an important factor in over the network data acquisition approaches such as this.

Simulation of resource management - Virtualisation of hardware in a cloud data centre provides hardware resource isolation from running customer services. Such loose coupling between software and hardware layers allows for deployment of a VM on any server node that has sufficient resources within the cloud data centre. Cloud data centre providers use resource management frameworks to deploy, migrate and resize VMs in an automated manner due to the scale of resources that are being managed. These resource management frameworks are applied to optimise compute resource delivery to the services in accordance to the policy chosen by a cloud service provider. Reflecting resource management decisions within a simulation experiment is very important as it plays a major role in resource distribution and hence system behaviour. Moreover, testing resource management algorithms within a simulation environment prior to deployment in a real environment is a desired function for the simulation framework and a key focus for this research work.

TCO Estimation - TCO has proven to be a useful decision supporting metric when evaluating various design or upgrade options within cloud data centres (Chang, Lee, *et al.*, 2013). However, when viewed in isolation, it does not provide detailed insights on system performance. In addition, with existing TCO formulas only crude power consumption estimations are made. This thesis is focused on connecting a simulation-based estimation approach to TCO, bringing system performance and cost estimations together. By running simulation experiments more fine-grained power consumption of compute equipment can be made, contributing to more accurate energy cost estimations. An expected output from a cloud simulation platform is time series data showing utilisation of server hardware resources over a period of time specified in a simulation experiment. Before using simulation results in TCO estimation, the resource utilisation must be converted into costs and only then included in the TCO formulas. Large scale

cloud data centre systems yield a large amount of simulation output data that require a programmatic approach for processing.

1.4 Chapter outline

The remainder of this thesis is organised as follows:

Chapter 2 – Literature Review: This chapter provides a foundation for the research presented in this thesis. As the research topic is an amalgamation of cloud data centre technologies, resource management techniques, costs estimation methods and simulation approach, the review covers concepts and thorough research analysis within these areas.

Chapter 3 – Related Project Background – FP7 (CACTOS): This thesis has a bidirectional connection with EU funded CACTOS project. CACTOS chapter describes the project concepts, use cases, tooling and modelling approach of the project. The project description is framing the thesis concepts in the setting of a wider research area.

Chapter 4 – Methodology: This chapter outlines research methodology followed throughout thesis research. Methodology describes individual steps taken from the problem definition, data collection to solution design development validation experimentation and result analysis.

Chapter 5 – Design and Implementation: This chapter defines simulation framework requirements based followed by software architecture design and implementations decisions. Covering design and implementations details for integration between the tooling, cloud data centre runtime and cost models and proposed TCO calculation models.

Chapter 6 – Experimentation and Analysis: The chapter describes the validation and “what-if” experiment design, execution and analysis. The first part of the chapter compares simulation experiment results to the real system measurements. While the second chapter part explores data centre equipment upgrade decision caveats demonstrating the usage of the simulation-driven TCO estimation.

Chapter 7 – Discussion and Conclusions: The chapter provides concluding summary of the research findings presented in the thesis, discussion of the presented approach and future work in the research area.

2 Literature Review

2.1 Introduction

As already discussed in the previous chapter, cloud computing has become a part of a technological and business revolution which is continuing to evolve and grow (Liu, Wang, et al., 2009; Subashini and Kavitha, 2011; Raghavendra and Krishna, 2015, Arizton, 2018). To cater for this demand the cloud data centre market is predicted to grow at a 28.7% Compound Annual Growth Rate, reaching a total of €58.21 billion by 2023 (Infoholic Research LLP, 2017). In addition to prolific growth, cloud data centre operators also have to contend with ever evolving technological solutions with a typical data centre changing technologies three to four times over a ten-year period (Mehmi, Sangal, et al., 2016). This rapidly changing environment coupled with the typically large scale of a cloud data centre and associated system complexity makes it challenging to understand the full extent of management decisions, such as hardware procurement and resource allocations, have on costs. From an evaluation of the domain, available cost calculation methods used in practice, such as TCO, are limited in their ability to take into account the effect of resource optimisation on costs and performance, if at all. Based on this rapidly evolving domain, one of the primary aims of this thesis is focused on developing a more comprehensive methodology and decision support tool for cloud infrastructure providers, which can better account for resource management policies and the consequential impact on costs and QoS.

To address this issue, the following chapter provides a detailed assessment of the literature focusing on the modern cloud data centre as a system composed of actual hardware equipment and facilities with the adopted concept of resource virtualization to avail of resource sharing among provided cloud computing services. Tracing the connection between physical hardware and virtualized resources is fundamentally important for understanding the costs cloud infrastructure providers leverage. As well as highlighting the underlying cloud data centre principles, Section 2.2 emphasises the fast evolution of technology and processes underpinning the need for more advanced, up to date decision supports approach proposed in this thesis. Section 2.3 provides definitions and a breakdown of cloud resource management techniques and challenges. Resource management frameworks have a direct impact on QoS and costs making them

fundamental to the research being undertaken in this thesis. Section 2.4 assesses the literature related to the costs associated with data centres and available calculation methods. The role of QoS and its impact on costs is explored within the section making clearer the connection with resource management policies. Finally, Section 2.5 describes decision support tools and methods available for cloud infrastructure managers. This section reviews literature related to cloud optimisation and cloud simulation tools and their role in assessing the impact of management decisions e.g. changes in hardware configuration and/or changes within the resource managements approaches.

2.2 Inside the modern cloud data centre

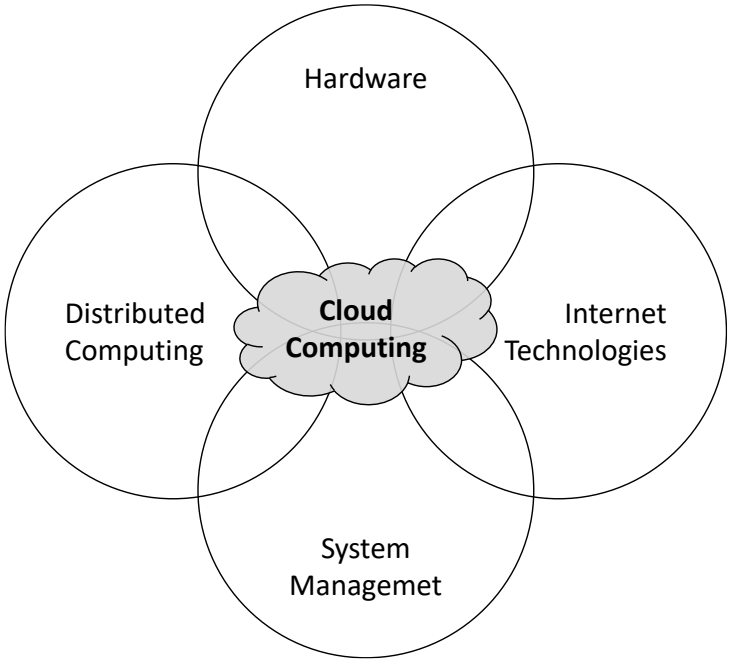


Figure 2-1: Convergence of various advances leading to the advent of cloud (Rochwerger, Vázquez, *et al.*, 2011)

Computing as a utility was defined, well ahead of its time, in 1966 by Canadian technologist Douglas F. Parkhill (Parkhill 1966), with the first use of the term “cloud computing” traced to 1996 (Regalado 2011). The term “cloud computing” covers a wider array of services and technological concepts and can be seen to have many definitions. The often used (Marston, Li, *et al.*, 2011; Galante and De Bona, 2012; Zhao, Peng, *et al.*, 2012; Aceto, Botta, *et al.*, 2013) and best fitting definition for the presented study

comes from the U.S. National Institute of Science and Technology (NIST) (Mell and Grance, 2011) stating that:

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."

The NIST provided definition describes cloud computing as an ecosystem of intertwined technologies. Rochwerger et al. (2011) identifies a set of new advancements (Figure 2-1) in computer hardware, distributed computing, systems management and internet tools that formed the cloud computing paradigm. With the increasing availability and advancement of technology there has been a phenomenal rise in the use of cloud computing systems in recent times (Liu, Wang, *et al.*, 2009; Subashini and Kavitha, 2011; Raghavendra and Krishna, 2015). In 2017 cloud services worldwide generated revenues of around €133.6 billion and were predicted to increase almost two fold by the end of 2021 reaching a total of €261 billion (Gartner Inc., 2018). The use of cloud services in 2016 among enterprises based in EU countries reached 21%, Finland being the highest cloud user with a 57% score (Eurostat, 2016). It is also forecasted that by 2019 more than four-fifths of compute workloads will be executed in the cloud (Cisco, 2014).

2.2.1 Bricks and mortar

Such high demand for computing services has led to extremely large-scale commodity-computer data centres. Data centres became the providers of computing hardware also known as compute resources. An ability to manage and host computing resources on such a large scale served as a necessary enabler for cloud computing (Armbrust, Armbrust, *et al.*, 2009).

Depending on data centre size and various other functional or business decisions the actual premises where hardware equipment is placed can be part of a dedicated standalone building or a section of a building that also fulfils other functions. For day-to-day operations, computing equipment inside the hosting space of a data centre requires access to a constant power source, reliable cooling, and human personnel for maintenance and security. All these vital on premise facilities take up additional space proportional to computing capacity. Data centres accommodate auxiliary premises that are required to support its main function. For staff these premises might include a control

room office area, a canteen, car park and toilet space. As electric power is the primary and vital resource for hardware equipment, data centres would typically host power transformation and distribution equipment together with batteries and diesel power backup generators ready to prevent equipment shutdown in case of main power failure. Computing equipment consumes electricity and dissipates heat, with large scale units also requiring separate premises for hosting cooling equipment. The data centre building must be a secure space. Information that is stored and processed is always considered private therefore direct physical access to the equipment must be strictly limited only to authorised personnel. Security measures are also taken to prevent possible damage from natural disasters such as floods, tornadoes or fire. The building itself should be originally designed or modified to withstand or minimise the damage from possible dangers of the surrounding environment. For example, the fenced perimeter around premises keeps intruders away, raised floors prevent flood damage and various fire suppression systems tackle fire outbreaks quickly (Alger, 2012; Barroso and Hölzle, 2013; Geng, 2014).

Table 2-1 compiled based on information presented by Alger (2012) lists a number of operational data centres built around Europe and the United States and illustrates the relationship between overall building size, the actual hosting space, and power and capacity expressed in cabinet numbers. As can be seen from the table the total building size is on average twice the size of the hosting space itself. This space is needed to host the auxiliary gear mentioned earlier to ensure operational conditions for the cabinets where computing equipment is located (Geng, 2014). From Table 2-1 it can be seen that with the growing number of cabinets, the overall power consumption and data centre size also increases. However due to differences in system design it is possible to achieve higher density solutions that use less space and provide higher cabinet capacity i.e. the Intel data centre in New Mexico, which has more cabinets per square meter than the eBay data centre in Phoenix, Arizona. The trend of building large data centres continues with the current top 10 largest facilities occupying space ranging from 69,677 m^2 to the largest one of 668,902 m^2 (Mitchell, 2017)

Organization	Location	Building Size (m^2)	Hosting Space (m^2)	Cabinets	Power avg. (kW)
Barcelona Supercomputing Centre	Barcelona, Spain	Undetermined	160	48	1,400
Bahnhof	Stockholm, Sweden	1,000	500	140	800
ACT, Inc.	Iowa City, Iowa, USA	622.5	371.6	150	900
Green House Data	Cheyenne, Wyoming, USA	882.6	696.8	200	1,000
eBay	Phoenix, Arizona, USA	3,901.9	1,300.6	256	4,000
Intel	Rio Rancho, New Mexico, USA	1,672	623	268	8,000
Cisco	Allen, Texas, USA	15,050.3	2,573.4	754	4,901
IBM	Research Triangle Park, North Carolina, USA	14,864.5	9,290.3	960	21,000
NetApp	Research Triangle Park, North Carolina, USA	11,612.9	3,065.8	2136	25,000
IO	Phoenix, Arizona, USA	49,981.8	33,445.1	~3,000	120,000

Table 2-1: List of existing data centres (Alger, 2012)

At the core of every data centre is the compute hardware hosting space or server room. The hosting space contains a number of cabinets where the servers and networking equipment are placed onto the racks. Depending on the performed function each server hardware unit typically referred to as a compute node, a storage node or a network node. As the name suggests the compute node is responsible for processing heavy computational tasks. For that purpose, it will be equipped with more central processing units (CPU) containing more processing cores coupled with high capacity random access memory (RAM) units. Storage nodes will have the primary objective of providing and managing storage space, such nodes will be equipped with arrays of Hard Disk Drives (HDD) or Solid State Drives (SSD). Finally, network nodes are responsible for handling the connections between servers by providing packet rule-based routing mechanisms performing job of a programmable switch, firewall or router device. Network nodes are equipped with multiple network connection outlets suitable for communications over fibre optic or copper cables. Electric power is required for servers

to operate and is managed by power distribution units (PDU) which are also mounted onto the racks inside the cabinets. Respectively these different types of hardware nodes form the pool of compute resources often referred to as CPU cores, memory, storage, and network (Alger, 2012; Barroso and Hölzle, 2013; Buyya, Vecchiola and Selvi, 2013; Geng, 2014).

2.2.2 Hardware virtualization

Virtualization allows for separation between data centre hardware and software layers. Buyya et al. (2013) describes virtualization as a core technology for cloud computing that encompasses a collection of solutions allowing for the abstraction of the fundamental elements, such as hardware (CPU, memory, I/O devices), runtime environments, storage, and networking. Virtualization abstracts the real underlying Host system resources and exposes the Guest system(s) via virtual interfaces otherwise known as Virtual Machines (VM). Communications between VMs and real hardware is performed by the thin software layer called a Hypervisor (Hugos, Michael H.; Hulitzky 2011). Agnostic to the guest system the hypervisor makes the host system appear as a different virtual system to the guest (Smith & Nair 2005).

Hypervisor technology was first described as a VM manager by R. Goldberg (1973). The virtualization abstraction allows the valuable feature of running multiple Operating Systems (OS) and software stacks on a single physical platform side by side in isolation (Rochwerger et al. 2011). Once the hardware is virtualised it can be shared among multiple guests increasing resource utilisation. Virtualisation also allows for resource aggregation where multiple hardware node resources are bundled to appear as one single virtual machine. This way resource aggregation delivers performance unattainable by existing physical configuration. Because virtualisation is essentially a representation of hardware employing software logic it also allows for emulation of specific hardware behaviour without actually having it (Jennings and Stadler, 2014). This feature can be used for example in smartphone application development where specific hardware architecture is emulated for testing purposes (e.g. Android emulator (Google, 2016)).

The native (bare metal) hypervisor type is used in cloud computing. It is located directly between the guest and the host hardware (Figure 2-2). This approach provides VM containers with assigned virtualised resources for guest tenants directly without

having a host operating system in the way (Younge, Henschel, *et al.*, 2011). In other words, a hypervisor serves as middle tier layer for a VM (server) to make resource demand calls to the physical hardware. Examples of bare metal hypervisors include vSphere ESX/ESXi (VMware, 2016b), Xen (XenProject, 2016) and Hyper-V (Microsoft, 2016b). On top of deployed virtualised infrastructure cloud services can be built in the form of platforms and/or online applications. The services by proxy are managed using these virtualisation features.

2.2.3 Cloud deployment models and services

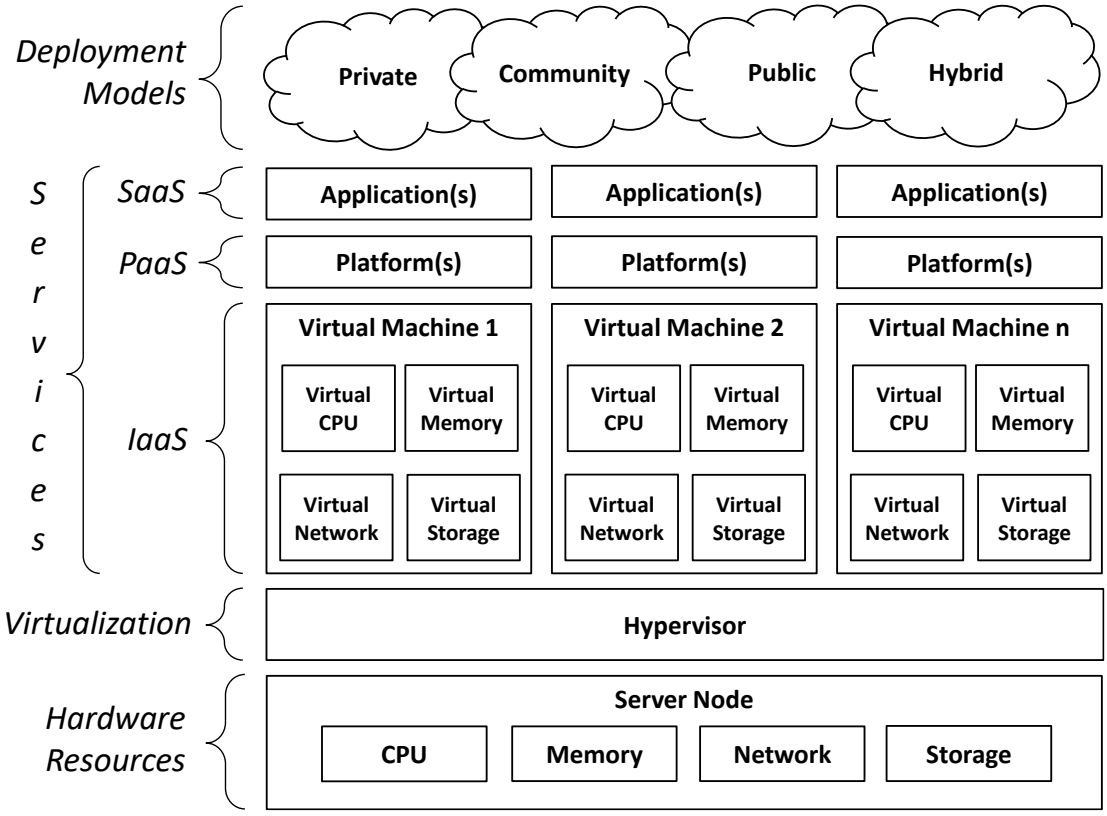


Figure 2-2: General cloud architecture derived from existing studies (Smith and Nair, 2005; Catteddu and Hogben, 2010; Bohn, Messina, *et al.*, 2011; Rochwerger, Vázquez, *et al.*, 2011; Buyya, Vecchiola, *et al.*, 2013)

Cloud deployment models (see Figure 2-2) are created to define properties of the cloud physical infrastructure. By a certain model type it is possible to identify to whom cloud infrastructure is provisioned, who owns, manages, operates and physically hosts

it. Four cloud deployment models as described in the NIST definition of cloud computing article (Mell and Grance, 2011) follows:

- The *Private cloud* infrastructure is provisioned to be used by a single organisation exclusively. The infrastructure can be owned and managed by the same organisation or by a 3rd party vendor. It can exist on or off premises (Buyya, Vecchiola, *et al.*, 2013).
- In the case of the *Community cloud* model the infrastructure is provisioned among a specific community of users that might share common requirements (e.g. geographical location, security, use of specific application (Zissis and Lekkas, 2012)). The infrastructure can be provided by one of the community members or a 3rd party organisation. It can exist on or off premises.
- *Public cloud* deployment model allows access to cloud services to the public. Infrastructure commonly owned by cloud provider organisation or company and resides on their premises (Jamsa, 2013) (e.g. Microsoft Azure (Microsoft, 2016a), Amazon AWS (Amazon, 2016), VMware vCloud (VMware, 2016a)).
- Finally, the *Hybrid cloud* model enables the composition of private, community or public cloud models if they share the same data and application portability standards. For example, if the application running on a private cloud requires more resources than the local infrastructure can provide the load can be scaled out to the public cloud using cloud bursting technology if the appropriate interfaces are supported by the public cloud provider (Dillon, Wu and Chang, 2010).

Cloud services provide different levels of computing utilities for cloud users. The logical hierarchy of the service stack is shown in Figure 2-2 and also reflects the order in which services are virtualised allowing the service user to disregard any management tasks below the service hierarchy (Rochwerger, Vázquez, *et al.*, 2011). As defined by NIST(Mell and Grance, 2011) three core cloud services are:

- *Software as a Service* (SaaS) – can be compared to renting access to software packages that can be used remotely. A consumer has access to

the remotely running application only via a web browser or program interface (e.g., web based email, weather forecast service) (Savolainen, 2012).The control over the underlying hardware resources such as network, servers, operating system or even software execution parameters are not accessible. Once a consumer subscribes to SaaS the service provider is responsible for managing all the software updates, maintenance ensuring sufficient resources are available for requested provisioning (Zhu and Wen, 2012).

- *Platform as a Service* (PaaS) – can be described as renting the programming language environment where a customer can execute their own software code (e.g. Apache Tomcat (Apache, 2015), WildFly (RedHat, 2015), Google App Engine (Google, 2015)). In this case a service consumer has full access rights to the application and hosting environment settings (Pardeshi, 2014). However cloud infrastructure resources including network, servers, operating systems, and storage are strictly managed by the service provider (Hashizume, Rosado, *et al.*, 2013).

- *Infrastructure as a Service* (IaaS) – gives the most control over the resources by allowing the consumer to rent custom provisioned computing resources such as Central Processing Unit (CPU), Network, Memory and Storage in a single entity – e.g. a Virtual Machine (Buyya, Vecchiola, *et al.*, 2013). This service can be compared to renting a remotely running personal computer which can be accessed over the network. The consumer can select and configure the type of operating system together with deployed software services. In this case the consumer is also required to take care of the software stack running on the VM. The service provider is only responsible for resource availability (Hashizume, Rosado, *et al.*, 2013).

With cloud technology adoption, the services start to be portioned more specifically, reflecting more exactly the type of function they provide. For example Dillon et al. (2010) states that the delivery of virtualized storage on demand became a separate cloud service naming it Data storage as a Service (DaaS). Recently in a cloud services survey, Duan et al. (2015) define cloud services as Everything as a Service (XaaS). The

development of services evidently shows the versatility of cloud technology and the choice it can offer to the user.

Thousands of server nodes are hosted inside hyper scale cloud data centres forming a concentration of hardware resources reaching up to 10.6 Million CPU cores (TOP500.org, 2016). These compute resources are joined by network interconnects and virtualization technology into one large heterogeneous pool of shared virtual compute resources that needs to be divided among VMs and distributed between the cloud services provided to end users. With constantly fluctuating demand for cloud services, cloud resource management at such a proportion becomes a complex task that directly impacts cost efficiency of the data centre enterprise (Hameed, Khoshkbarforoushha, *et al.*, 2016).

2.3 Data centre resource management

Separating hardware and software layers using virtualization creates freedom for guest systems (VM's) to be moved seamlessly around data centre physical nodes and allowing dynamic physical resource allocation in accordance to actual user demand (Bousselmi, Brahmi and Gammoudi, 2014). These virtualization features create finer resource management opportunities, which can be tailored to increase compute resource utilisation rates, improve energy consumption, increase service performance and service reliability. Naturally the cost associated with a data centre is related to the resource management approaches and if done correctly financial gains can be achieved making a cloud data centre more cost efficient, profitable and more competitive (Shi and Hong, 2011).

2.3.1 Resource provisionning methods

Service cost has been identified as the most critical success driver for cloud computing (Astri, 2015). Dynamic resource allocation is a core cloud feature allowing for the cloud user to pay only for the resources they need (Xiao, Song and Chen, 2012). Hardware virtualization-based resource provisioning methods such as live migration, VM co-location and elasticity are used by cloud infrastructure providers to distribute available resources.

Live migration. The virtualised environments operate as isolated entities. Isolation makes it possible to seamlessly migrate VMs to different hardware and separate

resources between multiple guests on the same host (Buyya, Vecchiola, *et al.*, 2013). Containing the server within a VM also allows for live migration of VMs (Clark, Fraser, *et al.*, 2005). During live migration, running VMs can be relocated from one physical node to another seamlessly without server downtime. This is a powerful feature that can be used to improve fault tolerance, simplify system maintenance and most importantly allow dynamic physical resource balancing and development of smart scheduling policies (Jennings and Stadler, 2014).

Co-location. By placing multiple VM's on the same node cloud providers can balance between service performance and resource utilisation. By achieving greater resource utilisation cloud providers can for example achieve energy savings when consolidating VMs on fewer nodes and essentially have more free resources that can be sold (Buyya, Vecchiola, *et al.*, 2013; Ahmad, Gani, *et al.*, 2015).

Elasticity. The concept of elasticity is one of the main features of cloud computing which allows cloud consumers to increase or decrease provisioned resources dynamically (Galante and De Bona, 2012). Resource virtualisation allows for more dynamic workload management than with traditional configurations, where the application is executed directly on the server nodes hardware (Armbrust, Armbrust, *et al.*, 2009). The virtualised system can be scaled *horizontally* and v*ertically*. Horizontal scaling refers to the spawning of a parallel array of mirrored VMs and load balancers allowing system workload to be seamlessly spread out, thus removing the pressure for the single VM instance (Vaquero, Rodero-Merino and Buyya, 2011). Vertical scaling refers to the adjustment of resources that are allocated to a single VM. At first this method was used mostly for disk space allocation, but now can be seen to be used for memory reclamation techniques such as "ballooning" (Rochwerger, Vázquez, *et al.*, 2011; VMware, 2011).

The scale at which resources must be provisioned is too large to be done manually by hand, hence the autonomic resource management systems are used to fulfil this purpose (Armstrong, Espling, *et al.*, 2015).

2.3.2 QoS aware autonomic resource management

The large scale of modern data centres, resource heterogeneity and underlying interdependencies make resource management a complex task. In addition, fluctuations in user demand render VMs as dynamic entities in their resource requirements and

lifecycle. To deal with constant new VM arrivals, existing VM closures or scaling, cloud providers must implement efficient self-adaptive resource provisioning policies (Jiang, Perng, *et al.*, 2012).

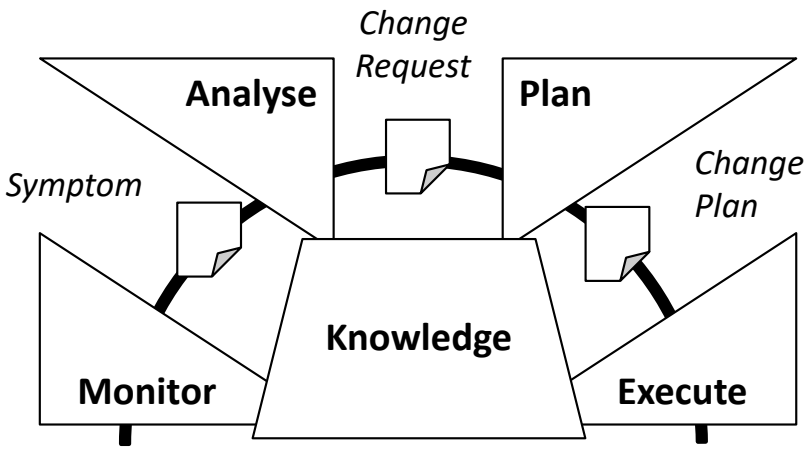


Figure 2-3: MAPE-K feedback loop (IBM, 2005)

Orchestrated regulation of cloud infrastructure is achieved by following the MAPE-K autonomic control feedback loop method (Huebscher and McCann, 2008). The concept MAPE-K was introduced by IBM (Bantz, Bisdikian, *et al.*, 2003) and stands for *Monitor*, *Analyse*, *Plan*, *Execute*, and *Knowledge*. As shown in Figure 2-3 the system is being monitored for specific symptoms, for example resource utilisation levels, network throughput or application behaviour patterns. Next, this monitored data is being analysed against defined policies. If the current state meets the policy conditions a change request is sent towards the planning phase. During the planning phase a change plan is produced that is then put into action at the execution phase. All of these steps are based on knowledge of system inner workings. Knowledge can consist of accumulated historical data, system topology or other combinations of sources or known environment states (Hariri, Khargharia, *et al.*, 2006).

Continuous resource management and allocation processes in cloud data centres fall under the definition of autonomic self-adaptive systems (Youseff, Butrico and Da Silva, 2008). Originally the concept of autonomic systems was taken from human biology where the human body self-regulates internal processes according to information taken from the central nervous system. For example, heart rate and breathing will automatically increase in case of increased physical load to deliver oxygen faster to the muscle tissues. However reactions within the human body are involuntary whereas in IT

systems they are decided by system administrators (IBM, 2005). The objectives of cloud data centre resource regulation are driven by agreed upon quality criteria that provided cloud services must meet to satisfy the needs and expectations of their customers. Such criteria are frequently referred to as Quality of Service (QoS) or Service Level Agreements (SLA) where the latter often represents a legal contractual agreement between customer and cloud service provider (Goudarzi and Pedram, 2011).

Different needs of cloud service consumers require a definition of QoS requirements beyond the availability of the service parameter (Quarati, Clematis and D'Agostino, 2016). Over the years cloud resource management platforms focused on the range of parameters to be improved. In a systematic review of autonomic resource management in cloud computing Singh & Chana (2015a) identified an array of QoS parameters as shown in Figure 2-4.

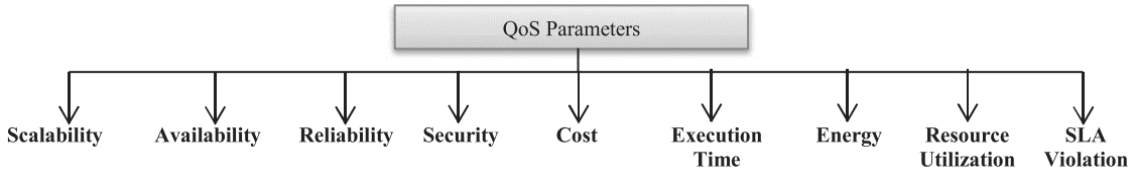


Figure 2-4: Taxonomy based on QoS parameters (Singh and Chana, 2015a)

Based on the QoS parameters identified by Singh & Chana (2015a) the following section explores each of these QoS parameters further in more detail:

Scalability is the requirement for the cloud service to adapt dynamically to the increase or decrease in user demand using virtualization elasticity properties. The QoS monitored parameters can include the cloud service response time, number of virtual machines and time *to scale* (Kaur and Chana, 2014; Singh and Chana, 2015b).

Availability parameter defines the ability of system to ensure uninterrupted access to the cloud service even in the case of internal hardware failures within the cloud data centre. The QoS monitored parameter is service uptime and is measured in percentage per amount of time (Emeakaroha, Fatema, *et al.*, 2016). Some business-critical applications might not tolerate more than five minutes of downtime which makes availability a particularly technically challenging QoS target (Lango, 2014).

Reliability defines the system performance consistency. Cloud data centre resources are typically shared among multiple hosts and some performance degradation

is expected due to neighbouring VM interference (Ragusa, Robinson and Svorobej, 2013). System reliability parameter refers to the objective that the same request should be processed within an expected *timeframe* regardless of the workload induced by co-located VMs (Corradi, Fanelli and Foschini, 2014).

Security is the QoS parameter defining the data protection within the cloud data centre. The measurement of QoS is defined as the number of security features a cloud provider guarantees at a different service level. For example at the SaaS level access control, communication and software security must be ensured, at the PaaS or IaaS level the virtual cloud protection, data security and image security might be important and finally at the physical data centre level the network protection and hardware security might be important QoS parameter (Zissis and Lekkas, 2012; Singh and Chana, 2015a).

Cost as a QoS metric is defined by means of the amount of money that can be spent on a cloud service per hour (Singh and Chana, 2015a). Sharing resources among multiple cloud users can reduce IT costs significantly, compared to the conventional in-house dedicated server hardware, still costs remain an important QoS indicator (Khajeh-Hosseini, Greenwood and Sommerville, 2010).

Execution Time parameter captures the duration of time which is required by the system to completely process a particular workload or task. The system must assign an appropriate amount of compute resources to meet the target QoS deadline for the job execution. This is a challenging task as it is hard to gauge workload requirements in advance (Hwang and Kim, 2012).

Energy constraints depicts the amount of energy consumed by the cloud resources to execute the workload (Singh & Chana 2015a). Because energy consumption impacts directly upon the operational costs of a cloud data centre and the environment, being energy efficient is an important metric to manage (Chihi, Chainbi and Ghedira, 2013). Energy efficiency can also be included into an SLA contract. In such case the core measurable aspects include energy ensumption (kilowatts per hour per VM) and carbon emission (kg CO2 per hour per VM) (Ziegler, 2012).

Resource Utilisation is defined as the ratio between the time the cloud resource spends on processing workload and the resources idle time (Singh and Chana, 2015a). On one hand, low resource utilisation is a sign of wasteful overprovisioning, while on the other hand high utilisation can be a sign of bottlenecks caused by insufficient resources

which might lead to poor QoS in some of the other metrics. Finding a balance in resource utilisation is an important task for autonomic resource management (Calheiros, Masoumi, *et al.*, 2015).

SLA Violation is a metric of breaking agreed service parameters. An SLA will define constraints to the performance characteristic bounds, created for a specific customer application running in the cloud infrastructure. If the SLAs are broken then the cloud provider can suffer penalties which were negotiated when the agreement took place (Kaur and Chana, 2014; Lango, 2014; Singh and Chana, 2015a).

In the cloud computing domain, optimisation models should fit the QoS multi-objective resource allocation constraints that impact distribution of computing power, storage capacity, memory size, network bandwidth, power consumption and other factors in order to achieve optimal system configuration (Chaisiri and Niyato, 2009; Wang, Liu, *et al.*, 2012). Cloud data centres on average consists of tens of thousands of physical compute nodes where virtualised services are placed. This composition represents a large, heterogeneous system containing thousands of virtual and physical components that exhibit stochastic relational behaviour traits. Due to such high complexity and loose object dependency, cloud resource management proves to be a multifarious task which is usually addressed by using cloud orchestration frameworks (Behavior, Copil, *et al.*, 2011).

2.3.3 Cloud resource orchestration frameworks

To address cloud resource management challenges, the related research community has invested a significant amount of effort in creating cloud orchestrating frameworks (Ahmad, Gani, *et al.*, 2015). Common approaches of cloud resource orchestration frameworks that follow MAPE-K feedback loop were analysed. The most recent literature associated with this topic is presented in the following analysis.

Resource orchestration originated in grid computing where computing resources were used to process large amounts of incoming user requests identified as jobs (Krauter, Buyya and Maheswaran, 2002). These incoming jobs needed to be distributed efficiently to available resources. The load balancing purpose in cloud computing remained the same, the incoming user requests were required to be distributed onto an array of VM's in the most efficient manner, while keeping the resource utilisation high to reduce costs (Shams, Powell., *et al.*, 2010). A number of survey studies have also been

presented with a focus on particular types of resource management layers. For example the cloud services orchestration study of Bousselmi et al. (2014) focused on cost-efficient decisions of hardware level orchestration of IaaS and software level orchestration of SaaS. Firstly, they compared SaaS resource scheduling, provisioning and deployment algorithms. Secondly, cloud resource management systems were compared as part of IaaS orchestration solutions. The survey of Ahmad et al. (2015) looks specifically at the VM migration and consolidation frameworks. They state that circumscribed resource utilisation leads to high operational costs of cloud data centres. They review the inner workings of consolidation frameworks by reviewing the architecture, resource assignment policies and VM co-location criteria. Furthermore, resource scheduling is also analysed by comparing the improvements made in network bandwidth, energy efficiency and storage migration overhead reduction. A survey on cloud computing elasticity by Galante & De Bona (2012) presents comprehensive information on the state of the art and classifications on elasticity mechanisms, based on an analysis of both commercial and academic solutions. This classification identifies the benefits of elasticity solutions as cost, performance, energy and increase in infrastructure capacity. Coutinho et al. (2014) presents a systematic review that addresses definitions, metrics and tools for measuring, evaluation of elasticity, and existing solutions. This study identified the main groups of metrics that are related to elasticity as: allocation, capacity cost, QoS, resource utilisation, scalability and time. Next, it classified elasticity solutions by the method (i.e. horizontal scaling, vertical scaling, migration) and by model (i.e. reactive, proactive/predictive).

There is a broad array of academic papers presenting different orchestration frameworks. To further probe the research directions for cloud resource optimisation Table 2-2 has been compiled. The purpose of this table is to present a study of publications describing orchestration frameworks and evaluate these orchestration frameworks under the headings of monitoring, heuristics and algorithms, and validation. The headings were chosen to reflect steps of the earlier described MAPE-K control loop i.e. Monitoring, Analyse, Plan, Execute. The monitoring heading is divided into utilisation and profiling subheadings, which refer to the data types that orchestration frameworks base actions on. The utilisation data type implies that there is no additional data processing, just the raw monitoring resource utilisation data extracted from the system. The profiling section indicates that some monitored data processing and profiling is being

completed to predict future workload direction. The section for heuristics and algorithms is classified according to the type of resource management function the framework is performing i.e. workload load-balancing, VM scaling, initial VM placement and VM consolidation. Lastly, the validation column shows the orchestration framework deployment environment that authors chose to demonstrate the viability and performance of their approach. The simulation method refers to use of a cloud simulation framework for deploying resource management algorithms and the test bed refers to a real system that consists of physical hardware. The test bed is usually a small subset of virtualized nodes joined together by the network which is capable of running cloud services.

Orchestration Framework	**Monitoring**		**Heuristics and Algorithms**				**Validation / Deployment**	
	Utilisation	Profiling	Load Balancing	Scaling	Initial Placement	Relocation / Consolidation	Simulation	Test Bed (real system)
DMA on IBM Director (Khanna, Beaty, *et al.*, 2006)	X					X		X
pMapper (Verma, Ahuja and Neogi, 2008)	X				X	X	X	X
Polyphony (Shams, Powell., *et al.*, 2010)			X					X
vManage (Kumar, Talwar, *et al.*, 2011)	X				X	X		X
DMF (Liu, Mao, Van der Merwe, *et al.*, 2011)	X			X1	X1	X		X
COPE (Liu, Loo and Mao, 2011)	X					X	X	
TROPIC (Liu, Mao, Chen, *et al.*, 2011)	X				X			X2
VM Scheduler (Xiao, Song, *et al.*, 2012)	X	X		X			X	X
LiveCloud (Wang, Liu, *et al.*, 2012)	X		X					X

An Adaptive Hybrid Elasticity Controller (Ali-Eldin, Tordsson and Elmroth, 2012)	X	X		X			X2	
A Coordinated Reactive and Predictive Approach (Moore, Bean and Ellahi, 2013)	X	X		X				X2
ACDC (Svärd, Li, *et al.*, 2014)	X				X	X	X	X
A Resource Elasticity Framework for QoS-aware Execution of Cloud Applications (Kaur and Chana, 2014)	X	X		X				X
OpenStack Neat (Beloglazov and Buyya, 2014)	X	X			X3	X		X
Apex Lake (Metsch, Ibidunmoye, *et al.*, 2015)	X	X		X1	X1	X1		X
CWMF (Singh and Chana, 2015b)	X	X	X				X	
Profiling-Based Workload Consolidation and Migration (Ye, Wu, *et al.*, 2015)	X	X			X	X		X2
Cool Cloud (Zhang, Hsu and Chang, 2015)	X				X		X	X
Open NFV (Krishnaswamy, Krishnan, *et al.*, 2015)	X	X	X					X
A General and Practical Consolidation Framework in CloudNFV (Pham, Tran, *et al.*, 2015)	X	X			X3	X	X	
UMA (Chen, Chen, *et al.*, 2015)	X				X	X	X	
Energy-Efficient Resource Allocation and Provisioning Framework (Dabbagh, Hamdaoui, *et al.*, 2015)	X	X	X				X	
Experimental Analysis on Autonomic Strategies (Dupont, Lejeune, *et al.*, 2015)	X	X		X				X
RPM (Sedaghat, Hernández-Rodriguez and Elmroth, 2016)	X	X				X	X	

Dynamic Software Consolidation (Tchana, De Palma, *et al.*, 2016)	X					X		X
Auto-healing Service Framework (Li, Li, *et al.*, 2017)	X			X1	X1			X
CMS (Awada and Barker, 2017)	X	X			X	X		X
ETSO (Mechtri, Ghribi, *et al.*, 2017)	X				X		X	X
Agent-based Workflow Monitoring, Prediction and Adaptation Framework (El-Kassabi, Serhani, *et al.*, 2018)	X	X			X	X		X
The Big Data On Cloud Agile Provisioning Framework (Lu and Zhou, 2018)	X			X	X	X		X
1 – No orchestration algorithms are presented within the publication, but are possible to integrate 2 – Workloads induced using replayed traces 3 – Bin packing best fit used, possible other heuristic integration								

Table 2-2: Identified cloud computing orchestrator frameworks taxonomy from relevant literature analysis

Cloud orchestration frameworks rely on live resource data monitoring, where pooled data is then analysed to make a decision in relation to any resource adjustment action that might be needed. As per Table 2-2 the monitoring abilities are divided into two subsections: utilisation and profiling. Such distinction is made to differentiate between orchestration frameworks that act only upon live monitoring data (Svärd, Li, *et al.*, 2014) and resource orchestration frameworks that also predict system behaviour by collecting, analysing and profiling historical data (Xiao, Song, *et al.*, 2012). Some monitoring frameworks deploy custom data collection and monitoring agents on each node to monitor the resource consumption of physical nodes and each VM running on that node. The locally collected information is then aggregated at the node cluster level or at the global data centre level depending on the centralised or decentralised framework architecture design (Beloglazov and Buyya, 2014; Metsch, Ibidunmoye, *et al.*, 2015; Sedaghat, Hernández-Rodriguez, *et al.*, 2016). Other frameworks integrate seamlessly with the existing cloud managing tools providing the additional level of decision support. In order to do that the event triggers are created in the form of certain resource thresholds or periodic time based reoccurrence (Khanna, Beaty, *et al.*, 2006).

The workflow management technique described by Singh et al. (Singh and Chana, 2015b) uses monitored data for machine learning to regulate decisions based on specific rules. By predicting system behaviour using historical monitoring data one can take proactive measures in avoiding delays in resource assignment. This approach is dominant in the resource elasticity governing frameworks. For example the QoS-Aware Resource Elasticity (QRE) framework (Kaur and Chana, 2014) has separate component application behaviour which is used to predict arrival rates of different task classes for the upcoming time interval. Ali-Eldin et al. (2012) also relies on these proactive methods to estimate the future cloud service load based on historical application behaviour data.

From the analysis presented in Table 2-2 it can be seen that the resource orchestration frameworks are developed to address resource management at different levels of cloud architecture by using specific heuristics and algorithms. Internal cloud data centre management can be performed by controlling the following resource allocation policies:

- Workload allocation policies
- Application allocation policies
- VM scaling policies
- VM placement and relocation(consolidation) policies

The cloud orchestration framework heuristics and algorithms presented in Table 2-2 are represented visually in Figure 2-5 as allocated to their respective layers of workload manipulation. On the top layer, the workload allocation policies effecting load balancer decisions can influence which application instance will receive jobs to execute, thus keeping the overall workload evenly distributed (Shams, Powell., *et al.*, 2010; Wang, Liu, *et al.*, 2012; Krishnaswamy, Krishnan, *et al.*, 2015). One layer down the application allocation policies decide which VM in the cloud data centre is most suitable for hosting a new application instance. A single VM can host multiple applications without compromising the performance as long as there are enough resources for all co-located application instances (Jennings and Stadler, 2014). In the example applications shown in red and green were co-located where application instances in blue were deployed to a separate VM. Another layer down is the orchestration policies that control vertical and horizontal scaling of VM to allow additional resource provisioning if the workload increase

and resource de-provisioning if the workload decreases (Ali-Eldin, Tordsson, *et al.*, 2012; Xiao, Song, *et al.*, 2012; Moore, Bean, *et al.*, 2013). Lastly, VM placement and relocation policies decide which physical hardware equipment will host the VM which was just created or provide the VM relocation decision and destination (Liu, Loo, *et al.*, 2011; Beloglazov and Buyya, 2014; Svärd, Li, *et al.*, 2014). By orchestrating resources at each level the focus of each orchestration policy is to satisfy QoS constraints of the cloud services (Singh and Chana, 2015a).

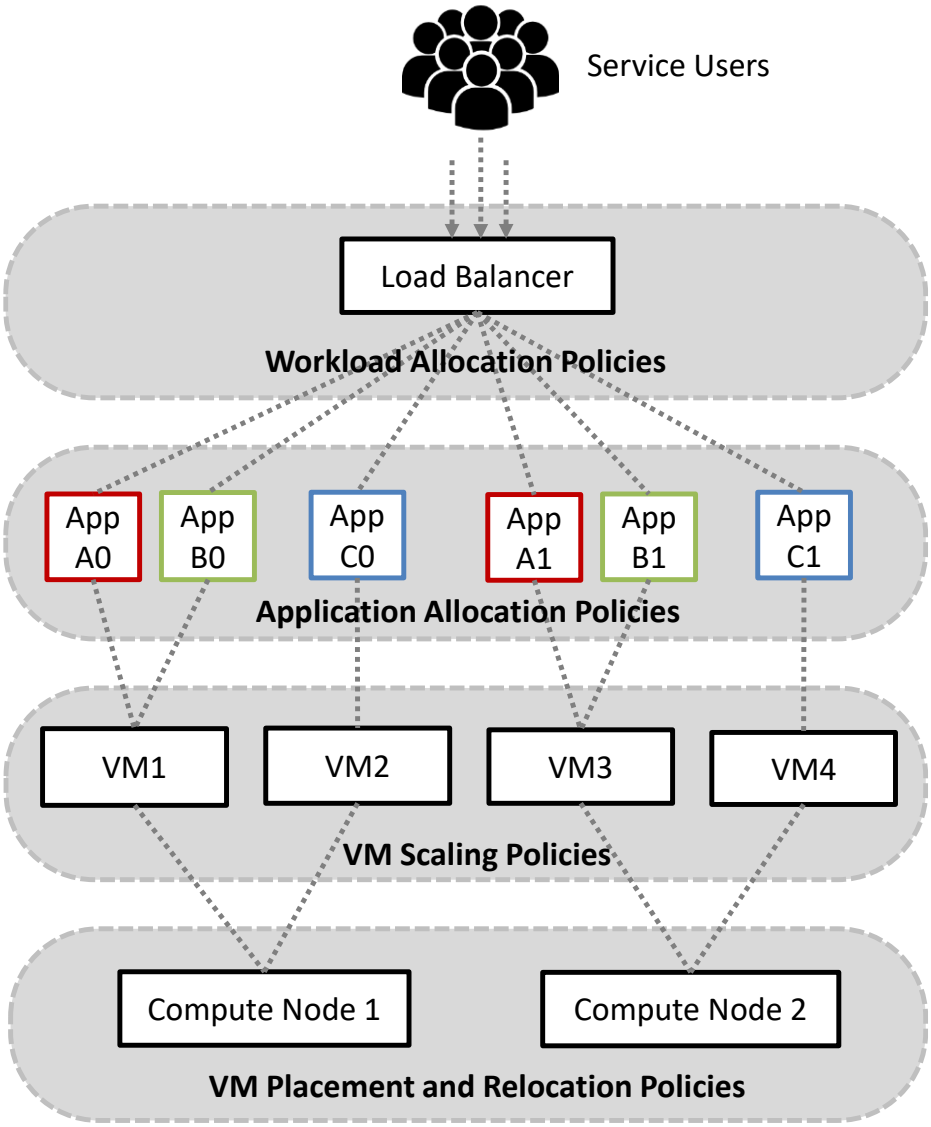


Figure 2-5: Generic example diagram of resource orchestrating policies application points in DC

To study the performance of the proposed resource orchestration frameworks an evaluation exercise is performed to ensure that the framework behaves in an expected manner. During the validation experiments the system behaviour metric is collected and measured against desired QoS parameters (Ahmad, Gani, *et al.*, 2015). From the resource orchestration framework analysis presented in Table 2-2 the validation step can be executed in two ways by using simulation or by deploying the orchestration framework to the real system. From a total of 25 analysed frameworks fourteen were validated using only test bed deployment, seven were validated using simulation and the remaining four were validated using both simulation and test bed deployment methods. The individual choice between using simulation and/or the physical system was not explicitly stated in all 25 presented studies, but the implied reasoning behind choosing simulation over the test bed was due to following reasons: (1) faster experimentation processing times (Verma, Ahuja, *et al.*, 2008) (2) ability to test orchestration within a large system setup with at least 1000 VMs or more representing a real data centre size (Liu, Loo, *et al.*, 2011) (3) inability to access a real cloud data centre (Singh and Chana, 2015b). Benefits of using a real system for running experiments was expressed as the need for real hardware or service behaviour measurement data as it was also an unknown part of an experiment. For example, with Polyphony resource orchestration frameworks (Shams, Powell., *et al.*, 2010) where the focus is on the I/O bound job execution within the public Amazon cloud where cloud storage service performance is a key for the frameworks success and thus needs to be measured in the real system.

Low cost is one of the anticipated benefits cloud computing offers to its end users. Reduction in service costs comes from virtualized compute resource sharing among multiple cloud users where unused capacity can be utilized more effectively, reducing wasteful idle time (Bobroff, Kochut and Beaty, 2007). The more resources a cloud provider is able to sell, the more potential profit can be generated. Cloud resource orchestration frameworks play a significant role in distributing resources, hence choosing the right resource management framework is important for “squeezing” the maximum use of available resources, generating income and meeting QoS constraints (Hameed, Khoshkbarforoushha, *et al.*, 2016). However, before putting up a price tag on any of the cloud services and coming up with the pricing strategies, cloud service providers should know how much it costs to acquire and operate all the data centre infrastructure

(Thanakornworakij, Nassar, *et al.*, 2012). The following section discusses the types of costs and methods for costs calculations available for cloud data centre.

2.4 Data centre costs

The 1990's IT revolution created concerns over IT cost containment, justification and effectiveness measurement (Drury, 2001). A Gartner report published in the 1990's had shown that the total cost of an IT asset over its entire lifetime will often exceed the original asset procurement costs. These revelations made the pre-existing Total Cost of Ownership (TCO) methodology a discipline that became increasingly relevant to the IT sector (Lycette and Lowenstein, 2010). Calculating TCO consist of direct and indirect cost variables that incur during the life cycle of the of an IT asset. These costs include, but are not limited to process of asset acquisition, deployment, maintenance, support and operation (Bailey and Heidt, 2003). TCO figures can also be split into the notion of Capital Expenditure (CAPEX) and Operational Expenditure (OPEX). The CAPEX costs are made upfront and account for the acquisition of permanent physical assets that can benefit business for a significant period of time e.g. a computer, while OPEX is a reoccurring periodical expense for running (operating) the asset e.g. the electricity cost consumed by the computer (Maguire, 2008).

Electric power is a crucial resource of data centre operation. Maintaining constant energy flow is a significant cost for the enterprise and challenge for the infrastructure provider. For example, infrastructure overhead can consist of large-scale generators, transformers and battery based uninterruptible power supplies (UPS) (Greenberg, Hamilton, et al., 2008). To ensure service availability in an event of power loss, a network of onsite power generators and uninterrupted power supplies are often deployed to ensure failsafe service operation all year round. Studies show that even a temporary downtime can lead to serious revenue loss, thus availability is a core metric for infrastructure providers (Coutinho, De Carvalho Sousa, *et al.*, 2014). Energy consumption is immense, due to the large scale, with cloud data centres required to handle high voltage connections directly from the national power provider. For this reason, cloud data centres deploy onsite transformer stations to consume high voltage energy flow, which, in some cases, stems from alternative energy suppliers (Choi, Lee, *et al.*, 2015). Deploying and maintaining these additional power management solutions generates extra overhead for the cloud data centre.

At the Initial cloud data centre build planning stage precise TCO cost calculation plays an important role for budgeting (Drury, 2001). Building a cloud data centre from scratch is a huge undertaking and the incorrect estimation of funds needed to build and sustain work of a cloud data centre can result in costly delays or a complete halt of the whole project due to its infeasibility (Geng, 2014). Later, after successful data centre release into production, the periodic evolution of hardware gradually will provide new opportunities for cloud services and will introduce changes of cloud user demands. In order to stay competitive and to provide relevant levels of services by meeting changing user demands, data centre management must periodically evaluate new products introduced by hardware manufacturers. It is estimated by experts that on average every ten years a cloud data centre will undergo between three and four major equipment changes (Mehmi, Sangal, *et al.*, 2016), therefore TCO calculation plays an important, continuous role in providing a solid base for building a business justification case when choosing between a range of new available IT implementation options (Geng, 2014).

In the modern age the size of the data centre varies according to the needs of the enterprise it is built to support. The hardware infrastructure installation can reach significant proportions, for example where a large and complex network could consist of a hundred thousand nodes occupying a building on their own (Mitchell, 2017). To accurately calculate the cost of acquisition, provision and management of a data centre for any involved party is a demanding task especially when applied to large-scale environments (Pandi and Karthick, 2016). In recent years, different TCO models have surfaced specifically aimed at the data centre management sector.

The common direct cost metric components that prevail within available TCO calculation frameworks in the literature are presented in Table 2-3. The top three cost components for cloud data centres shared amongst six publications are: depreciation, compute power cost and cooling power cost. Furthermore, five TCO calculation models also include server equipment cost and personnel cost parameters. Fewer TCO models separately include costs for cooling and network equipment acquisition, facilities, software, real-estate and server spares. Even fewer account separately for possible loan repayments and the cost of power equipment. From Table 2-3 it can be concluded that general cost metrics remain the same for most of data centre TCO calculation cases, however some of the methods have a more detailed breakdown of the cost components.

Therefore, before adopting one or other TCO framework one must closely examine if the selected framework covers the required cost components on a case by case basis.

Direct Cost Metric	Number of Publications	Source Publication(-s)
Depreciation (Amortisation) Cost	6	1(Li, Li, *et al.*, 2009), 2(Hardy, Sideris, *et al.*, 2011), 3(Greenberg, Hamilton, *et al.*, 2008), 4(Patel and Shah, 2005), 5(Simonet, Lebre, *et al.*, 2016), 6(Barroso and Hölzle, 2013)
Compute Power Cost	6	1(Li, Li, *et al.*, 2009), 2(Hardy, Sideris, *et al.*, 2011), 3(Greenberg, Hamilton, *et al.*, 2008), 4(Patel and Shah, 2005), 5(Simonet, Lebre, *et al.*, 2016), 6(Barroso and Hölzle, 2013)
Cooling Power Cost	6	1(Li, Li, *et al.*, 2009), 2(Hardy, Sideris, *et al.*, 2011), 3(Greenberg, Hamilton, *et al.*, 2008), 4(Patel and Shah, 2005), 5(Simonet, Lebre, *et al.*, 2016), 6(Barroso and Hölzle, 2013)
Server Equipment Cost	5	1(Li, Li, *et al.*, 2009), 2(Hardy, Sideris, *et al.*, 2011), 3(Greenberg, Hamilton, *et al.*, 2008), 5(Simonet, Lebre, *et al.*, 2016), 6(Barroso and Hölzle, 2013)
Personnel Cost	5	1(Li, Li, *et al.*, 2009), 2(Hardy, Sideris, *et al.*, 2011), 4(Patel and Shah, 2005), 5(Simonet, Lebre, *et al.*, 2016), 6(Barroso and Hölzle, 2013)
Cooling Equipment Cost	3	2(Hardy, Sideris, *et al.*, 2011), 3(Greenberg, Hamilton, *et al.*, 2008), 5(Simonet, Lebre, *et al.*, 2016)
Network Equipment Cost	3	1(Li, Li, *et al.*, 2009), 3(Greenberg, Hamilton, *et al.*, 2008), 5(Simonet, Lebre, *et al.*, 2016)
Facilities Cost (cables, KVM, etc.)	3	1(Li, Li, *et al.*, 2009), 4(Patel and Shah, 2005), 5(Simonet, Lebre, *et al.*, 2016)
Real-Estate Cost	3	1(Li, Li, *et al.*, 2009), 2(Hardy, Sideris, *et al.*, 2011), 4(Patel and Shah, 2005),
Software Cost	2	1(Li, Li, *et al.*, 2009), 4(Patel and Shah, 2005)
Server Spares Cost	2	2(Hardy, Sideris, *et al.*, 2011), (Sousa, Lins, *et al.*, 2015)
Loan Interest Rates Cost	1	6(Barroso and Hölzle, 2013)
Power Equipment Cost	1	3(Greenberg, Hamilton, *et al.*, 2008)

Table 2-3: Core TCO calculation variables mentioned in the literature

The work of Patel & Shah (2005) summarizes in a broad sense the key drivers behind every TCO model as the sum of space, hardware power, cooling and operation (Equation 1). Cost of space refers to the physical space acquisition or rental of all the data centre facilities. The cost of hardware power includes energy consumption for both network and compute resources. However, the cost of cooling with related power consumption is calculated as a separate group. The cost of operation variable covers expenses incurred in relation to day to day operation demands such as personnel, compute equipment depreciation and software licenses.

$$Cost_{total} = Cost_{space} + Cost_{power\ hardware} + Cost_{cooling} + Cost_{operation} \quad (1)$$

To account for cloud virtualisation and resource elasticity Li et al. (2009) proposes a similar TCO model that calculates costs based on eight parameters: server, software, network, support and maintenance, power, cooling, facilities and real estate. But, because cloud resources are virtualised the basic unit of cloud is no longer physical hardware resources, instead the virtualized resources assigned to the services by hypervisors should be considered. Hence, the study proposes a three-layer model to calculate utilisation costs as depicted in Figure 2-6.

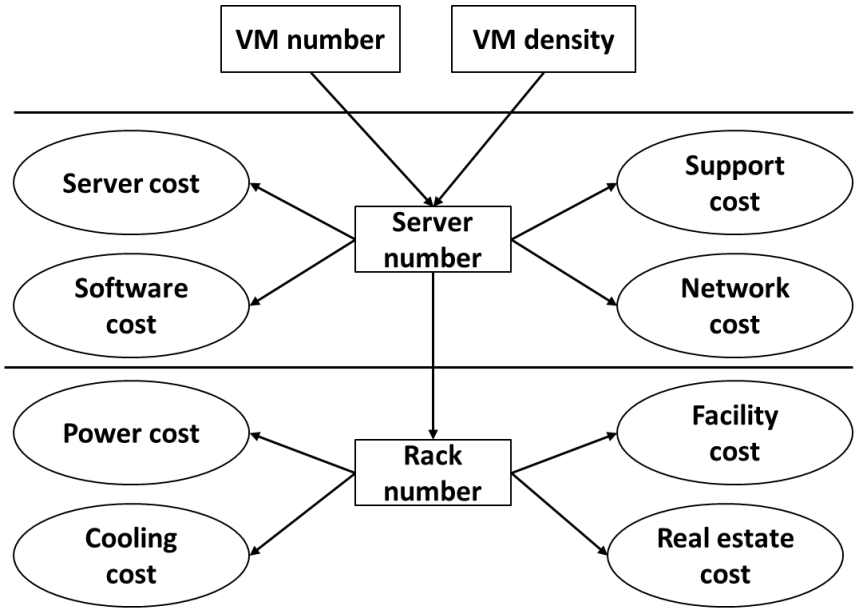


Figure 2-6: Three-layer model to calculate utilisation cost (Li, Li, *et al.*, 2009)

The first level of the model takes two additional measurements of the number of VMs and VM density to derive the utilisation costs. The number of VMs is obtained from an estimation of the amount of cloud service customers that will be using the system, and VM density depends upon the capacity of the hosted servers which can be obtained from the manufacturer's manual or by in-house testing. Then the VMs are assigned to servers using the function of bin packing algorithms (Brown, 1979). Finally, in the second and third model levels, costs are calculated according to the time fraction hardware is utilised for supporting functions of assigned VMs.

The data centre equipment can fail or experience age related performance degradation. Since both aspects directly influence the costs of a cloud data centre, Hardy et al. (2013) created the TCO model and framework to account for these additional costs. Their model introduces the notion of Mean-Time-To-Failure (MTTF), hot spares and cold spares. The cold spares are various system hardware components that are put aside to be used for repairs when the active component fails. The number of cold spares is estimated by using the MTTF value which stands for the operational time estimation for each product group. The number of hot spares is estimated based on the system workload where for some of the equipment performance degradation needs to also be accounted for. Figure 2-7 shows the logical impact of the cost of spares on TCO forming cost parameters. The authors argue that their TCO model can evaluate different data centre design options, helping to reduce costs and reduce the impact on environment by reducing power consumption.

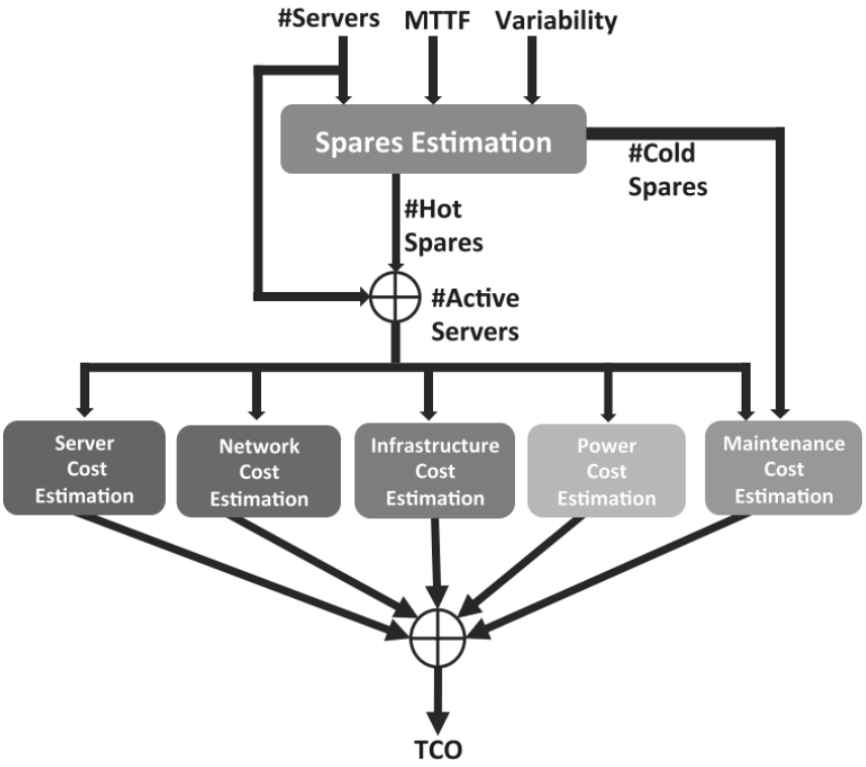


Figure 2-7: Kernel framework overview (Hardy, Kleanthous, *et al.*, 2013)

Greenberg et al. (2008) divides a cloud data centre TCO into server, power, network and infrastructure costs. The work is focused on indirect TCO improvement through better design decisions in network agility, resource allocation and geo-distribution. According to the authors computing resource fragmentation into few smaller geo-distributed data centre sites can be more performance and cost beneficial comparing to traditional large monolithic cloud data centre site. Following the same geo-distribution narrative, Simonet *et al.* (2016) introduces a cost model for distributed cloud computing which accounts for servers and storage, network, power, cooling, maintenance and facilities costs. What is unique for this TCO model is that it calculates the total cost of network as the sum of intranet costs and the internet costs, where the internet costs consist of amortized price of the backbone infrastructure (Equation 2). The cost of the internet backbone is a logical addition to the distributed cloud data centre TCO model because the element of connectivity between different sites plays an important role in such setup.

$$Cost_{internet} = P_{backbone} * A_m(A_s) \qquad (2)$$

Calculating TCO allows for an understanding of the cost distribution within the cloud data centre and can be used to derive a cost saving strategy that will be beneficial over the longer term. However, when managing compute resources the TCO metric must be used with caution to avoid overall system performance deterioration (Bailey and Heidt, 2003).

2.4.1 QoS impact on costs

The primary objective for Cloud providers is to rent out as much resources as possible to its customers at a reduced operational cost while still providing adequate service performance guaranteed by QoS or SLA metrics (Aceto, Botta, *et al.*, 2013). However, imbalance between system performance and resource provisioning can result in higher costs for cloud computing providers having a knock-on effect on revenue and competitiveness in the market.

Servers with subcomponents of CPU, Memory, and Storage are estimated to contribute towards approximately 45% of total data centre costs. Given such a steep server lifetime cost it is more beneficial for cloud computing providers if all servers are utilised by customers and producing revenue (Greenberg, Hamilton, *et al.*, 2008). However, if a customer's workload is spread thinly over the server nodes this leads to low server utilisation rates and in a data centre low resource utilisation is one of the main factors responsible for power inefficiency (Goudarzi and Pedram, 2012).

To combat power inefficiency and low server utilisation rates, resource orchestration frameworks employ VM consolidation policies to saturate physical nodes with VMs (Chen, Chen, *et al.*, 2015). Such dynamic virtual resource allocation allows for power savings by switching off unused servers and increasing utilisation per server (Goudarzi and Pedram, 2012; Jennings and Stadler, 2014). Unfortunately, VM consolidation can also lead to service performance degradation resulting in QoS and SLA violations (Lin, Chen and Lin, 2014).

If the agreed SLA constraints are violated the cloud service provider is liable to pay large sums of penalties to the customer in the wrong (Goudarzi and Pedram, 2011). It is a sensitive topic as the service performance can have a significant impact on customer revenues. For example, Google reported an experimentation incident where an increase in its search results display time by 500 ms led to a 20% loss in revenue. Similarly, Amazon noticed a correlation where an additional delay of 100 ms led to 1%

decline in sales (Greenberg, Hamilton, *et al.*, 2008). Hence, the SLA also serves as one of the input constraints for resource provisioning strategies and needs to be accounted for in cloud data centre cost calculations.

To summarise, for cloud data centre cost management it is crucial to have in place an effective resource provisioning policy that balances resource utilisation, energy efficiency and QoS. Estimating cloud computing system behaviour under different management strategies can be a challenging task due to the large size and high complexity of the system (Singh and Chana, 2015a). To help conduct such analysis simulation and optimisation decision support tools can be used to provide estimations on QoS performance (Becker, Becker and Meyer, 2013) and are further examined in the following section.

2.5 Simulation and optimisation decision support tools

Cloud computing is a self-adjusting autonomous system consisting of many different elements, where interactive behaviour influences overall system behaviour (Calheiros, Ranjan and Buyya, 2011). Optimisation policies for VM placement and VM consolidation must be tested and tuned for different scenarios before deployment in the production environment. These tests must be carried out to make sure that expected optimisation objectives are met and that the QoS and agreed SLAs will not be broken (Lin, Chen, *et al.*, 2014). To conduct the tests a small subset of isolated compute nodes, called test beds, can be used to observe the optimisation effects. However, the size of test beds is usually very small when comparing to real cloud data centre scale, which makes it difficult to extrapolate test results in a reliable manner (Garg and Buyya, 2011; Sotiriadis, Bessis, *et al.*, 2013). Also, it takes a lot of time to perform optimisation experiments using a test bed, as these experiments require significant amounts of time to be setup and executed in real time. For example in the study of the resource orchestration framework pMapper, the 60 different experiments required for the study were estimated to run for four months in a real system (Verma, Ahuja, *et al.*, 2008). In order to address the shortcomings of test beds, simulation frameworks can be used for cloud data centre behaviour evaluations (Son, Dastjerdi, *et al.*, 2015). Cloud optimisation in this chapter is presented in the context of resource optimisation algorithms that are used within cloud management frameworks. These algorithms can be adopted to use

with simulation data making it possible to integrate cloud optimisation decisions within cloud simulation frameworks (Calheiros and Ranjan, 2011).

2.5.1 Cloud optimisation

The process of cloud resource optimisation takes its roots from the branch of applied mathematics and numerical analysis. The purpose of global optimisation is to reach a system state where the minimum of effort would yield maximum possible results, the so-called discovery of optimum system parameters. Finding the optimum is the task of optimisation algorithms that are subject to a set of constraints or criteria defined by a function with single or multiple objectives (Fonseca and Fleming, 1995; Horst and Tuy, 1996; Weise, 2009).

In general, optimisation algorithms can be classified as deterministic or probabilistic. Deterministic algorithms are used to solve problems where clear relationships between system components exist. However, when the relationships are not straightforward, and the search space dimensionality is high, the probabilistic approach is preferred. Due to system complexity the non-deterministic algorithm families such as Monte Carlo (Mak, Morton and Wood, 1999) do not guarantee global optima, instead they focus on finding the best option within a given time. A technique called a heuristic function is used to guide the probabilistic optimisation algorithms in deciding which solution candidate to test next. Heuristics are designed to be aware of specifics of the domain and are integrated as part of the algorithm. This domain “knowledge” allows it to pick the most likely search space candidate routes reducing wasted computational effort (Fonseca and Fleming, 1995; Horst and Tuy, 1996; Weise, 2009; Li and Guo, 2011).

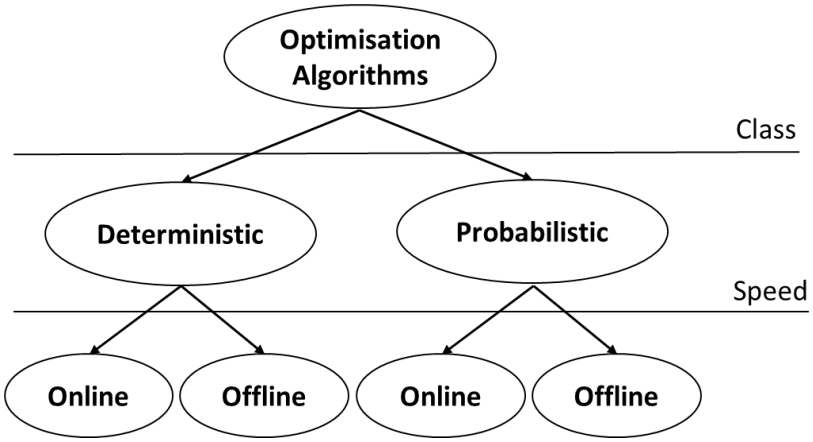


Figure 2-8:Optimisation algorithms classification summary (Weise, 2009)

Different decision speed requirements further divide the optimisation techniques into: Online and Offline optimisations. Online optimisation is applied to address tasks that require quick decisions within milliseconds or minutes (e.g. load balancing). These types of optimisation operations are carried out repetitively, focusing on the operation speed instead of optimality. In other words providing "good enough" solutions due to severe time constrains (Li and Guo, 2011; Li, Xu, *et al.*, 2011). Offline optimisations, on the other hand, have more time for calculations and are executed rather rarely (e.g. creation of a class schedule for an upcoming semester) focusing on obtaining optimal or close to optimal result (Weise, 2009). A cloud data centre on average consists of tens of thousands of physical compute nodes where virtualised services are placed (TOP500.org, 2016). This composition represents a large, heterogeneous system containing thousands virtual and physical components that is classified as a NP-hard optimisation problem (Li, Xu, *et al.*, 2011). Given the nature of such a nondeterministic environment it can be implied that cloud optimisation algorithms belong to the probabilistic algorithm class and the MAPE-K autonomic control method adoption require remediation actions to be executed with online speed.

When it comes to the cloud computing domain, optimisation models are required to fit the multi-objective resource allocation constraints which includes computing power, storage capacity, memory size, network bandwidth, power consumption and other factors in order to achieve an optimal system configuration defined by QoS (Chaisiri and Niyato, 2009; Wang, Liu, *et al.*, 2012). Cloud resource management allows for major benefits of global virtualized resource scheduling and on demand elastic resource provisioning within short periods of time (Ali-Eldin, Tordsson, *et al.*, 2012; Jennings and

Stadler, 2014). Further the literature presents examples of recent algorithms and a broad analysis of existing optimization algorithms and frameworks (Jennings & Stadler 2014; Manvi & Krishna Shyam 2014; Coutinho et al. 2014; Ahmad et al. 2015).

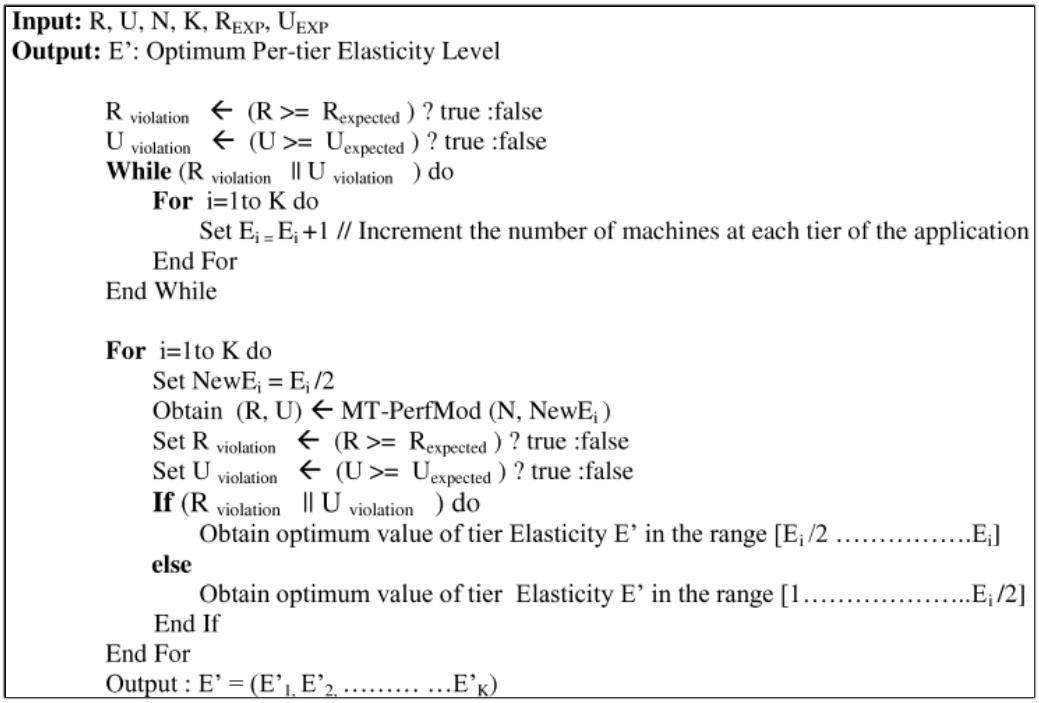
The area of global virtualized resource scheduling covers the admission control and migrations over time within the cloud data centre (Jennings and Stadler, 2014). The core of which is to be able to reduce resource overprovisioning by improving resource utilisation and using fewer hardware resources to perform the same tasks. For example the work of Tchana et al. (2016) focuses on optimizing dynamic VM consolidation to achieve greater energy efficiency and cost saving by combining software consolidation and VM consolidation. Their software relocation algorithm dynamically provisions resources for the application running within the VM, but in a case where the resources are insufficient on the current VM the application is migrated to another suitable. Compared to the baseline use case where average CPU utilisation was 12% their optimized scheduling approach reduces the number of VMs from 101 to 85 and number of hardware hosts from 66 to 9. The virtualized resource optimisation approach of Sedaghat et al. (2016) proposes a topology-aware placement scheduler framework which discovers favourable VM placement locations within cloud data centre. The proposed resource discovery algorithm as an input receives the lists of events that need to be processed and admits these events according to their priority on the best ranked placement candidates. The authors compared their frameworks performance against the First Fit Decreasing Sum (FFD-sum) of (Lee, Prabhakaran, *et al.*, 2010) and found that their solution improved the average CPU and Memory utilisation and reduced the computational cost of finding a placement candidate. The process of migrating VMs from one host to another can significantly decrease its performance as it requires additional system resources and time to be completed (Xu, Liu, *et al.*, 2014). The work of Chen et al. (2015) uses a dynamic consolidation technique to reduce power consumption, guarantee QoS and reduce power consumption while taking into account migration overhead. The proposed Utilisation-based Migration Algorithm (UMA) identifies stable hosts for VM migrations while reducing migration time and power consumption. In their method, at first migration the candidates list is created using the Best Fit with Decreasing bin packing algorithm (Yue, 1991), then using Tabu Search (Glover, 1989) to prune the candidate list. The experiment results of the UMA performance were compared with the

performance of the MinPower policy (Beloglazov and Buyya, 2012) showing a 77.5%-82.4% drop in VM migrations which led to 39.3%-42.2% reduction in power consumption.

Global VM allocation policies are deliberately built to address particular resource awareness such as: network aware placement (Wang, Meng and Zhang, 2011; Ilkhechi, Korpeoglu and Ulusoy, 2015; Lee and Park, 2015); cost aware placement (Sharma, Shenoy, *et al.*, 2011); energy aware placement (Biran, Corradi, *et al.*, 2012; Goudarzi and Pedram, 2012; Jing, Ali, *et al.*, 2013; Quang-Hung and Thoai, 2015). The latter two (cost and energy), being indirect resources that are derived from system utilisation data (Jennings and Stadler, 2014).

Cloud applications are typically composed of multiple components, each of which can be individually horizontally or vertically scaled to meet user demands (Coutinho, De Carvalho Sousa, *et al.*, 2014; Jennings and Stadler, 2014). The biggest challenge lies in modelling large-scale distributed application behaviour based on the diverse needs of user demands. Once the application model is created it can then be used to meet QoS, by applying the right scalability method at the right time (Kaur and Chana, 2014).

To create dynamic resource provisioning a combination of a proactive and reactive elasticity approach was taken by Ali-Eldin et al. (2012). They introduce adaptive hybrid controllers that dynamically change the number of VMs that are assigned to a running service. The actions of the controllers rely on the service capacity estimations based on the constructed queuing theory model (Bause and Dortmund, 1993). The authors compare their results with the results of a regression-based elasticity engine and report that proposed hybrid elasticity controller performs better by over allocating from 32% to 15% less resources and reducing SLA violation rates in the range of 2.1 to 1.48 times.



Input: R, U, N, K, R_{EXP}, U_{EXP}
Output: E': Optimum Per-tier Elasticity Level

$R_{violation}$ ← (R >= $R_{expected}$) ? true :false
$U_{violation}$ ← (U >= $U_{expected}$) ? true :false
While ($R_{violation}$ || $U_{violation}$) do
  For i=1to K do
    Set $E_i = E_i + 1$ // Increment the number of machines at each tier of the application
  End For
End While

For i=1to K do
  Set $NewE_i = E_i / 2$
  Obtain (R, U) ← MT-PerfMod (N, $NewE_i$)
  Set $R_{violation}$ ← (R >= $R_{expected}$) ? true :false
  Set $U_{violation}$ ← (U >= $U_{expected}$) ? true :false
  If ($R_{violation}$ || $U_{violation}$) do
    Obtain optimum value of tier Elasticity E' in the range [E_i /2 …….E_i]
  else
    Obtain optimum value of tier Elasticity E' in the range [1……..E_i /2]
  End If
End For
Output : E' = (E'$_1$, E'$_2$, …… …E'$_K$)

Figure 2-9: Per-tier resource elasticity "MT-ResElas" algorithm (Kaur and Chana, 2014)

The work of Kaur et al. Kaur & Chana (2014) demonstrates the application of a QoS aware resource elasticity framework. Their approach is based on analytical models of multi-tiered application behaviour that also accounts for user workload variability. Underpinning this research is the queueing network based Multi-Tier Performance Model (MT-PerfMod) and Multi-Tier Resource Elasticity (MT-ResElas) algorithm. At first, the MT-PerfMod calculates the application response time and resource utilisation, then, the MT-ResElas algorithm (shown in Figure 2-9) calculates the number of VMs required at every tier of the application. The authors measured the proposed approach by comparing the total amount of VMs with other related techniques, and the time it took to make scaling decision and costs. These comparative results indicate a lower amount of VMs used to satisfy the QoS constraints, in turn leading to the lower costs in almost all conducted experimentation.

The above cloud data centre resource optimisation algorithms demonstrate significant gains in energy efficiency, resource utilisation and QoS compliance. However measuring the performance of an optimization algorithm in the live environment testbed is not always feasible due to the small size and time constraints (Verma, Ahuja, *et al.*,

2008). Therefore cloud simulation tools are becoming increasingly important for determining the effects of optimisation strategies on the systems QoS performance (Buyya, Ranjan and Calheiros, 2009).

2.5.2 Cloud simulation

Discrete Event Simulation (DES) software has evolved from complex procedural language algorithms used in the 1960s to high level multi-application oriented simulators with graphical model building capabilities. These technological advancements made simulator platforms easy to use, but less flexible often fitting only specific domains (Bowden, 1998; Kelton and Law, 2000). Cloud computing simulators fall under the "Application oriented simulator" category, meaning they are developed most specifically for the cloud application domain (Zhao, Peng, *et al.*, 2012; Ahmed and Sabyasachi, 2014). Modern cloud simulation frameworks are built with a focus on performance prediction of cloud data centres. The model elements within such simulators already include pre-built elements and their defined interdependencies pertinent only to cloud (e.g. VM, elasticity policies, hypervisor) (Calheiros and Ranjan, 2011; Núñez, Vázquez-Poletti, *et al.*, 2012; Long, Yuqing and Qingxin, 2013).

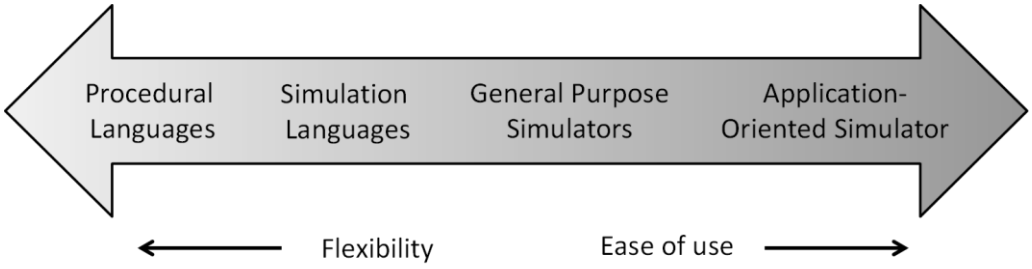


Figure 2-10: Updated spectrum of simulation software

A cloud computing data centre can be classified as a complex system and is defined by a large number of mutually interacting parts that behave in complex ways. To understand the behaviour of the system one must understand the behaviour of individual parts and how they act together forming the behaviour of the whole. These relationships between interacting parts possess distinct properties, for example, emergence, interdependence, and nonlinearity (Corrado, 2019).

Emergence refers to the relationship between the individual details and the larger view of the system. Specifically, determining the individual details that are more

important for the larger view of the system from those that are not (Bar-Yam, 1997). Complex systems exhibit properties that are not easy to predict by analysing the behaviour of their individual component properties in isolation. These emergent properties become more important in software systems as complexity grows in input variation, connection pattern and number of tiers in the system (Szabo and Teo, 2012). Understanding the interdependence between interacting objects in the system aids recognition of indirect effects, which can then be included in the model description (Bar-Yam, 1997). For example, an interdependence between software and infrastructure layers of cloud computing can be used for enhancing security features of the system (Muñoz, Gonzalez and Maña, 2012). Nonlinearity describes the phenomena of disproportionate relationships between cause and effect, or in other words when the same input can produce different output due to a change in the underlying system state (Marinescu, 2016). Complex systems are usually made by combining multiple heterogenous components together and therefore are difficult to test and troubleshoot (Carrozza, Loffreda and Manetti, 2012). To predict behaviour of such sophisticated systems, techniques of modelling and simulation can be employed to analyse interactions of individual events within the complex system (Patelli, Feng, et al., 2017).

Development of simulation solutions for cloud has been influenced by the distributed-computing paradigm termed grid computing. Grid computing also consists of multiple connected compute nodes and is used for non-interactive processing of batch workloads (Foster and Kesselman, 1998). Earlier DES efforts in this domain focused on simulation tooling support which was provided for uniformly aggregating and sharing distributed heterogeneous resources within large-scale applications, such as in the fields of science, engineering and commerce (Sulistio, Cibej, *et al.*, 2008). Simulation of grid computing involved modelling of hardware resource topology and evaluating effects of workload (job) scheduling algorithms (Krauter, Buyya, *et al.*, 2002; Li and Lan, 2005; Umale, 2013). Various grid computing simulators have been developed (Sulistio, Cibej, *et al.*, 2008) and are presented in literature, such as OptorSim (Bell, Cameron, *et al.*, 2002), MONARC (Legrand and Newman, 2000), SimGrid (Legrand, Marchal and Casanova, 2003), GridSim (Buyya and Murshed, 2002) and MicroGrid (Song, Liu, *et al.*, 2000). However, these alone cannot provide an environment which can be directly used by the cloud computing community (Zhao, Peng, *et al.*, 2012). In an extension,

Ostermann et al. (2011a) created the GroudSim platform which is capable of experimenting with grid and cloud environments.

To better understand the current cloud simulation landscape a systematic review of cloud simulation platforms was conducted using the IEEE Xplore digital library (IEEE, 2017). During the review a total of 256 papers were reviewed - 45 papers relate to the design, development and extension of simulation platforms with the remaining 211 describing the simulation platform research application. It is interesting to note that 218 papers relate to CloudSim, extensions or derivative simulators which equates to 85% of the total papers reviewed.

The first publication on cloud computing simulation research using open source platforms appears in 2009 with the introduction of the CloudSim toolkit (Buyya, Ranjan, *et al.*, 2009). From 2009 onwards, publications on the topic have increased consistently, largely driven by papers relating to CloudSim or the introduction of new cloud simulation platforms. By 2015, cloud simulation papers using open source platforms had become a regular topic in computer science publications having grown from only two in 2009, peaking at 78 in 2015 and remaining steady at around 50 yearly publications in 2016, 2017 and 2018 to September (see Figure 2-11). Presented scientific publication trends reflect the interest in cloud computing generally (Markets and Markets, 2016) and the growth of cloud computing simulation tool adoption.

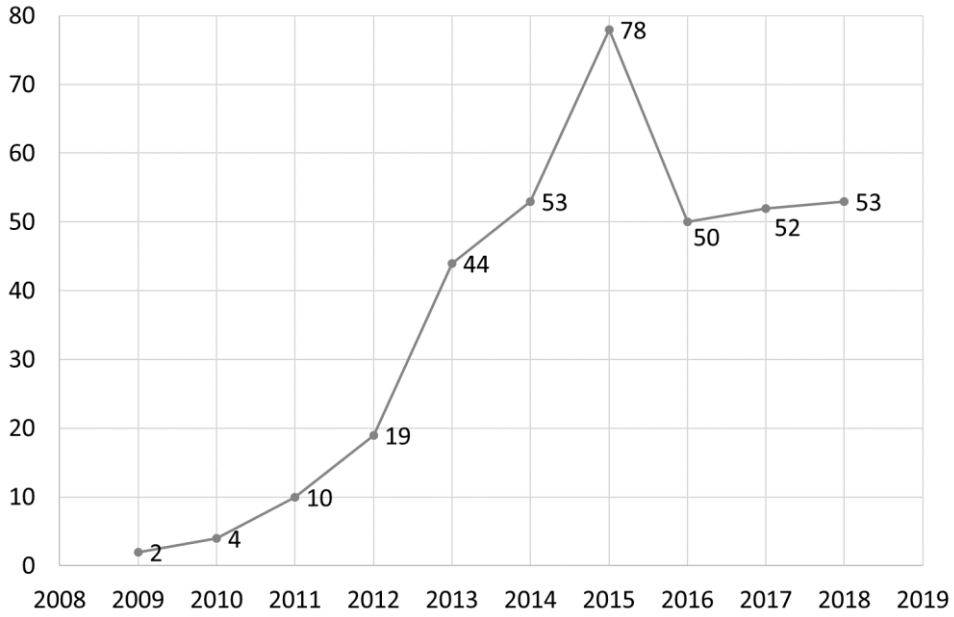


Figure 2-11: Cloud simulation research using open source platforms by publication outlet and year

The review identified 39 distinct cloud simulation platforms (see Table 2-4), 22 of which are presented in publications as extensions of CloudSim. Other cloud simulation platforms are built on top of existing DES engines, for example ICanCloud and CloudNetSim++ are built on top of the Omnet++ (OpenSim, 2015) DES engine, GreenCloud uses NS-2 (NS-2, 2014) as its underlying DES platform and CactoSim is using the Palladio simulator (Rathfelder and Klatt, 2011). Cloud simulation frameworks like CloudSched, CloudSim, DCSim[1] GDCSim, and SimGrid were all built from first principles explicitly for the purpose of cloud computing simulation.

Bazaar Extension* (Pittl and Schikuta, 2016)	DISSECT-CF (Kecskemeti, 2015)
CACTOSim (Ostberg, Groenda, *et al.*, 2014)	EMUSim* (Calheiros, Netto, *et al.*, 2012)
CDOSim* (Fittkau, Frey and Hasselbring, 2012)	GDCSim (Gupta, Banerjee, *et al.*, 2014)
CEPSim* (Higashino, Capretz and Bittencourt, 2015)	GreenCloud (Kliazovich, Bouvry, *et al.*, 2010)
Cloud2Sim* (Kathiravelu and Veiga, 2014)	GroudSim (Ostermann, Plankensteiner, *et al.*, 2011b)
CloudAnalyst* (Wickremasinghe, Calheiros and Buyya, 2010)	ICanCloud (Castañé, Núñez, *et al.*, 2013)
CloudEXP* (Jararweh, Jarrah, *et al.*, 2014)	iFogSim* (Gupta, Dastjerdi, *et al.*, 2016)
CloudNetSim++ (Malik, Bilal, *et al.*, 2015)	MDCSim (Lim, Sharma, *et al.*, 2009)
CloudReports* (Sá, Calheiros and Gomes, 2014)	MR-CloudSim* (Jung and Kim, 2012)

CloudSched (Tian, Zhao, *et al.*, 2015)	NetworkCloudSim* (Garg and Buyya, 2011)
CloudSim (Calheiros and Ranjan, 2011)	SimGrid (Legrand, Marchal, *et al.*, 2003)
CloudSimDisk* (Louis, 2015)	SimIC (Sotiriadis, Bessis, *et al.*, 2013)
CloudSimSDN* (Son, Dastjerdi, *et al.*, 2015)	SPECI (Sriram, 2009)
CMCloudSimulator* (Alves, Batista, *et al.*, no date)	TeachCloud* (Jararweh, Alshara, *et al.*, 2012)
DartCSim* (Li, Jiang, *et al.*, 2012)	Ucloud* (Sqalli, Al-saeedi, *et al.*, 2012)
DCSim[1] (Tighe, 2012)	WorkflowSim* (Chen and Deelman, 2012)
DCSim[2] (Chen, Liu and Chang, 2012)	DBCOF* (Kumar, Malik, *et al.*, 2017)
RecapSim* (Byrne, Svorobej, *et al.*, 2017)	Edgenetworkcloudsim* (Seufert, Kwam, *et al.*, 2017)
IsV2C (Kim, Han, *et al.*, 2017)	EdgeCloudSim* (Sonmez, Ozgovde and Ersoy, 2017)
CloudSim Plus* (Filho, Oliveira, *et al.*, 2017)	
** Derivatives or extensions of CloudSim* *[1] This refers to DCSim by* (Tighe, 2012) *[2] This refers to DCSim by* (Chen, Liu, *et al.*, 2012)	

Table 2-4: Identified cloud computing simulation tools

To give an insight into each of the available simulation frameworks, in regards to their abilities to estimate cloud computing QoS parameters, each of the identified simulation platforms was analysed according to the Singh & Chana (2015a) QoS taxonomy discussed earlier in section 2.3.2. Findings of the analysis are summarized in Table 2-5.

Although security is an important QoS parameter, it is not directly addressed in the identified cloud computing simulation tools in Table 2-4 and is thus not evaluated in Table 2-5. Although not included directly in this analysis security plays a key QoS role. For further details the reader is referred to Fernandes et al. (2014), which provides a comprehensive survey of security issues in cloud computing environments listing top threats such as data breaches, data loss, and account service traffic hijacking. To mitigate security risks Al Morsy et al. (2010) derives general recommendations to be implemented for the cloud computing solutions, such as identity and access management, secure software development lifecycle, encryption key management. Because networking is a big part of cloud computing some of the network security simulators can be used to test network-based attacks. For example, the NeSSi[2] network security simulator can be used to generate attack data for the purpose of evaluating the effectiveness of detection algorithms (Grunewald, Bye, *et al.*, 2011).

Table 2-5 presents the (Singh and Chana, 2015a) QoS parameters and identifies which are supported by each of the different simulation platforms with the exception of

security which has not been included in this table. Each of these individual parameters are discussed in more detail in the sections following the table.

Simulation Framework	**QoS Parameters**[§] (Singh and Chana, 2015a)							
	Scalability	**Availability**	**Reliability**	**Cost**	**Execution Time**	**Energy**	**Utilisation**	**SLA Violations**
CACTOSim (Ostberg, Groenda, *et al.*, 2014)	YES	NO	YES	YES	YES	YES	C, M, S	NO
CloudNetSim++ (Malik, Bilal, *et al.*, 2015)	YES	NO	YES	YES	YES	YES	C, M, S, N	YES
CloudSched (Tian, Zhao, *et al.*, 2015)	YES	NO	YES	YES	YES	YES	C, M, S, N	NO
CloudSim* (Calheiros and Ranjan, 2011)	YES	YES	YES	YES	YES	YES	C, M, S, N	YES
DCSim[1] (Tighe, 2012)	YES	NO	YES	YES	YES	YES	C, M, S	YES
DCSim[2] (Chen, Liu, *et al.*, 2012)	NO	NO	YES	NO	YES	NO	S	NO

SPECI (Sriram, 2009)	NO	YES	YES	NO	YES	NO	N	NO
SimIC (Sotiriadis, Bessis, *et al.*, 2013)	NO	NO	YES	YES	YES	YES	C, M, S	YES
SimGrid (Legrand, Marchal, *et al.*, 2003)	YES	NO	YES	NO	YES	NO	C, N	NO
MDCSim (Lim, Sharma, *et al.*, 2009)	YES	NO	YES	YES	YES	YES	C, S, N	NO
ICanCloud (Castañé, Núñez, *et al.*, 2013)	NO	NO	YES	YES	YES	YES	C, M, S, N	NO
GroudSim (Ostermann, Plankenstein er, *et al.*, 2011b)	NO	NO	YES	YES	YES	NO	C, N	NO
GreenCloud (Kliazovich, Bouvry, *et al.*, 2010)	NO	NO	YES	YES	YES	YES	C	NO
GDCSim (Gupta,	NO	NO	YES	YES	YES	YES	C	YES

Banerjee, *et al.*, 2014)								
DISSECT-CF (Kecskemeti, 2015)	NO	NO	YES	YES	YES	YES	C, M, S, N	NO

*Utilisation short codes: **C** – CPU, **M** – memory, **S** – storage, **N** – network*
[1] *This refers to DCSim by* (Tighe, 2012)
[2] *This refers to DCSim by* (Chen, Liu, *et al.*, 2012)
** CloudSim with all its extensions*
§ - Security parameter was excluded from the table as it is not part of identified cloud simulation frameworks

Table 2-5: QoS parameters addressed by existing simulation platforms

Scalability. Simulation of cloud computing scalability properties is largely addressed by introducing the opportunity of custom scheduling for user generated tasks. The concept is to employ an algorithm which will load balance workload among VMs in the cloud data centre, adding or removing VMs accordingly i.e. CloudNetSim++ (Malik, Bilal, *et al.*, 2015), CloudSched (Tian, Zhao, *et al.*, 2015), CloudSim (Calheiros and Ranjan, 2011) and SimGrid (Legrand, Marchal, *et al.*, 2003). Simulation platforms like CACTOSim (Ostberg, Groenda, *et al.*, 2014) and DCSim[1] (Tighe, 2012) introduce the notion of an application model as an additional workload separation layer between user generated requests and VMs. The application consists of a load balancer and one or more identical application instances that can be spread across multiple VMs in a cloud data centre (Tighe, 2012). The presence of an application model in simulation allows for more detailed application component interdependency exploration.

Availability. The problem of system availability is addressed by simulating fault occurrence. By simulating component failures, one can explore system behaviour and introduce appropriate remediation actions. For example the FIM-SIM (Nita, Pop, *et al.*, 2014) is an extension for the CloudSim (Calheiros and Ranjan, 2011) platform which allows for the modelling of fault events and injecting them during the simulation experimentation run. The types of events include failure of the physical host, VM failure and job failure and can be injected using different types of distributions. The SPECI (Sriram, 2009) cloud simulation framework focuses solely on hardware failure generation and self-recovery through autonomic resource orchestration.

Reliability. System reliability is modelled by accounting for variations in system behaviour using empirical probability distributions. The distributions are used to model application behaviour expressed as variation in resource demand and component failures (Castañé, Núñez, *et al.*, 2013; Jararweh, Jarrah, *et al.*, 2014) Probability distributions introduce stochastic element to that is useful for seeing trade-offs between execution time and costs (Ostermann, Plankensteiner, *et al.*, 2011b)

Cost. Cloud computing simulation cost estimation research can be divided into two major groups: calculation of cost per kW/h of energy used and calculation of cost per cloud resource unit used. Cloud simulation platforms such as CACTOSim (Ostberg, Groenda, *et al.*, 2014), CloudNetSim++ (Malik, Bilal, *et al.*, 2015), CloudSched (Tian, Zhao, *et al.*, 2015), CloudSim (Calheiros and Ranjan, 2011), DCSim (Tighe, 2012), SimIC (Sotiriadis, Bessis, *et al.*, 2013), MDCSim (Lim, Sharma, *et al.*, 2009), ICanCloud (Castañé, Núñez, *et al.*, 2013), GreenCloud (Kliazovich, Bouvry, *et al.*, 2010), GDCSim (Gupta, Banerjee, *et al.*, 2014) and DISSECT-CF (Kecskemeti, 2015) calculate costs based on data centre hardware energy consumption. To view costs of cloud computing from the user perspective GroudSim (Ostermann, Plankensteiner, *et al.*, 2011b) allows for the simulation of pricing policies that calculate cost per gigabyte of file transfer and the time interval costs per used cloud resources. Similarly CloudNetSim++ (Malik, Bilal, *et al.*, 2015) simulation platform allows for calculation of cloud service costs for users, based on pricing policies i.e. Lowest Pricing, Deadline based and Energy based. The billing is calculated based on the resource usage time and any SLA penalties violated by the user. Apart from energy costs, CloudSim (Calheiros and Ranjan, 2011) also supports modelling of cloud market components where different resource trading policies of cloud providers can be simulated. The market cost model consists of two layers PaaS and SaaS. The PaaS layer calculates customer expenses associated with renting of CPU, Memory, Storage and usage of network bandwidth. The SaaS model calculates the price per number of application service requests. Interesting to note, that discussion of GreenCloud (Kliazovich, Bouvry, *et al.*, 2010) simulation platform also mentions the possible beneficial cost trade-offs between different types of network cables for designing cloud data centre infrastructure, but these costs are not included in the simulation models.

Execution Time. As shown in Table 2-5 all of the available cloud simulation platforms support calculation of how much time it will take to execute a task. Each task

submitted by users requires a certain amount of compute resources to be completed, hence the required processing time duration is calculated with relation to available resources. For example, if CPU can process 1,000 Million Instructions per Second (MIPS) and a task consumes 10,000 MIP the time needed to execute the task is 10 sec (Calheiros and Ranjan, 2011; Núñez, Vázquez-Poletti, *et al.*, 2012).

Energy. Energy consumption is an important cost factor that greatly impacts resource management decisions in cloud data centre (Malik, Bilal, *et al.*, 2015; Singh and Chana, 2015a). As shown in Table 2-5 most simulation tools support estimation of energy consumption. The typical power consumption model includes weighted sum of CPU, Memory, Disk and Network utilisation where CPU workload intensity impacts the overall power draw and therefore calculated as an integral of energy consumption function over the period of time (Tian, Zhao, *et al.*, 2015).

Resource Utilisation. Estimation of resource utilisation plays an important role in the insights of cloud data centre performance. From Table 2-5 it is evident that cloud simulation frameworks have different granularity levels when it comes to modelling cloud data centre resources. CloudNetSim++ (Malik, Bilal, *et al.*, 2015), CloudSched (Tian, Zhao, *et al.*, 2015), CloudSim (Calheiros and Ranjan, 2011), ICanCloud (Castañé, Núñez, *et al.*, 2013) and DISSECT-CF (Kecskemeti, 2015) include models for all available resources i.e. CPU, Memory, Storage and Network. The next largest group is the simulation frameworks that model CPU, Memory and Storage utilisation i.e. DCSim[1] (Tighe, 2012), CACTOSim (Ostberg, Groenda, *et al.*, 2014) and SimIC (Sotiriadis, Bessis, *et al.*, 2013). Then, the GroudSim (Ostermann, Plankensteiner, *et al.*, 2011b) and SimGrid (Legrand, Marchal, *et al.*, 2003) provide models for simulation of CPU and Network resources. The GDCSim (Gupta, Banerjee, *et al.*, 2014) and GreenCloud (Kliazovich, Bouvry, *et al.*, 2010) simulate the utilisation of CPU, DCSim[2] (Chen, Liu, *et al.*, 2012) simulates only the Storage components with SPECI (Sriram, 2009) focusing only on simulation of Network utilisations.

SLA Violation. Breaking the threshold of an SLA is modelled within CloudNetSim++ (Malik et al. 2015), CloudSim (Calheiros & Ranjan 2011), DCSim[1] (Tighe, 2012), SimIC (Sotiriadis, Bessis, *et al.*, 2013) and GDCSim (Gupta, Banerjee, *et al.*, 2014) simulation frameworks. The workload execution time calculation is the main metric for estimating the breach of SLA where the cloud provider guarantees a certain level of performance for its services. If the workload cannot be completed within certain

amount of time defined by the SLA due to the shortage of resources, or a failure then an SLA breach is detected (Tighe, 2012; Gupta, Banerjee, *et al.*, 2014). In case of CloudNetSim++ (Malik et al. 2015) the roles are reversed and SLAs are enforced by the cloud service provider onto the cloud user. The model assumes that each user will purchase a certain amount of resources and anything used extra will trigger an SLA violation which will be billed for using separate pricing rates.

The cloud data centre system models can be adjusted to correspond to the actual scale of a large data centre, where equipment properties within models can also be changed to evaluate new equipment capabilities without making an actual purchase. Experimentation in a simulated environment is typically far less expensive economically than using a real testbed (K. Preston White and Ingalls, 2011). Additionally, such experimentation is repeatable and potentially scalable in terms of addressing the simulation of larger-scale systems (Núñez, Vázquez-Poletti, *et al.*, 2012). In addition, experimentations can be performed in a more timely fashion, and risks with respect to stochastic inputs can be taken into account (Maguluri, Srikant and Ying, 2012; Sedaghat, Hernández-Rodriguez, *et al.*, 2016). However it is noted by Sakellari and Loukas (2013) that while simulation offers a number of advantages especially in terms of such scalability and experiment repeatability, it is still based on assumptions and simplifications that might not fully represent an actual cloud. For this reason, it still might be preferable in some circumstances to use real cloud testbeds in place of simulation or to validate results developed in simulated environments. Sakellari and Loukas (2013) provide an overview of such testbeds and software frameworks for setting up such cloud testbeds.

2.6 Conclusions

This chapter set about providing a comprehensive overview of the cloud computing domain with a particular focus on the role of the data centre. As can be seen from the analysis there has been and will continue to be continued rapid growth in the demand for cloud computing services into the future. As described by Arizton (2018), the demand for cloud computing is predicted to continue to grow over the years and with cloud providers forming an expanding multibillion euro cloud ecosystem. The success of cloud computing can be attributed to overall technological progress in the ICT sector, but one of the major user adoption drivers is the significantly lower price of computing resources in public clouds, when compared to hosting an in-house infrastructure. Astri

(2015), recognised this when suggesting that that for an end user, ongoing cost reduction is one of the most critical success factors for cloud computing adoption. Such unprecedented demand in cloud services has led to the construction of many large cloud data centre facilities consisting of thousands of nodes (servers) all around the world. Cost control in such intricate environments plays a very important role allowing cloud operators to compete on service pricing while maximising profits. However, the scale and complexity of cloud data centres combined with innovation pace in ICT makes this a challenging task. As an indicator of the pace of change Mehmi, Sangal, et al., (2016), suggest that a typical data centre will change technologies three to four times over a ten-year period.

Where data centres struggle, is in obtaining a comprehensive understanding of the trade-offs between data centre costs (e.g. CAPEX and OPEX) and performance. For example, cloud computing infrastructure providers use resource virtualization which delivers a bundle of resource provisioning methods (e.g. VM migration, VM co-location) which can be used for resource management. These tasks are typically managed by cloud resource orchestration frameworks due to the typical scale and complexity of cloud computing data centre resource management. Efficient cloud resource management should lead to increases in resource utilisation rates, reduced costs and increased revenue, hence giving a competitive edge to the cloud infrastructure provider. However, inefficient resource management can lead to service performance degradation and potentially profit loss. Hence, performance parameters (such as QoS and SLAs) must be adhered to requiring the data centre operator to find the correct balance between optimal cost effectiveness and appropriate service levels. As mentioned in the literature, both over and under provisioning can have negative consequences on costs. Obtaining the correct balance between these two positions is a challenge that has been identified in the literature. In an attempt to address this challenge, cloud simulation and optimisation has proven to be an effective decision support method for fine tuning resource management frameworks. Optimisation algorithms can deliver resource allocation decisions and simulation can be used to predict the QoS impact of these decisions prior to deployment on a live system. This is evidenced by the increasing number of simulation based solutions presented in the literature from 2015 onwards. However, following a detailed review of the literature it can be seen that although it is known that QoS parameters directly impact data centre costs (CAPEX and OPEX) there is a clear disconnect

between this and the existing cloud simulation platforms cost analysis capabilities. In particular, there is a deficit in the suit of tools in the provision of a more comprehensive overall cost estimation for a cloud data centre based on its specific technologies and logical setup. This is an issue that is under review in this thesis.

In assessing the literature, the TCO method was identified as an approach which has been utilised in relation to data centre costing. Upon review it can be seen to allow for the capture of the full costs of a data centre, accounting for both operational and capital expenditure costs for example energy cost, cost of equipment, cost of building and labour. However, a significant limitation of the available TCO calculation methods presented to date in the literature is that they address data centre costs in isolation and of those identified, none account for the impact of resource management on costs and QoS. Furthermore, it was found that the reviewed TCO frameworks still rely on manual data entry using web-based interfaces or spreadsheet software which is propositioned as a significant drawback for dealing with the complexity of large scale heterogeneous systems like modern cloud data centre, which are continuously evolving. The presented data centre TCO frameworks are more applicable to once off analysis and less appropriate for use on an ongoing basis.

The remainder of this thesis sets about addressing these research challenges both as presented in this chapter and also confirmed by the industrial partners with whom this research was conducted. More specifically the remainder of the thesis sets out to address the research gap in current data centre decision support methods by the development of a unique methodology and associated application framework for data centre cost/performance impact analysis. The methodology will address the complexity of cloud data centre design, resource management and cost, providing a reliable decision support information for real world cloud infrastructure operators. As identified in the literature there are issues associated with the use of testbeds to evaluate data centre cost/performance. Given the ever increasing data centre scale, complexity and rate of change, the creation of granular models of such entire systems would not be feasible without automation. As such, the proposed methodology in this thesis contributes significantly to the possibility of conducting full data centre simulation based analyses due to a focus on model build aspects that can be automated. With up-to-date models of the ‘as-is’ situation readily available (through automated model build), data centre

managers can then potentially make and evaluate parameter changes and ultimately make better informed decisions.

3 Related Project Background – FP7 (CACTOS)

3.1 Introduction

The research presented in this thesis was conducted in conjunction with the European Union research project, CACTOS. This chapter provides both the background of the CACTOS project and a description of how this thesis fits within the broader agenda of the CACTOS project, including its contribution to the research direction of the European Union (EU) (European Commission, 2007, 2013, 2015). While a description of the overall CACTOS project is provided for completeness, it should be noted that parts of what are presented in this chapter (in particular work relating to CactoSim) have been developed partly as an outcome of work presented in this overall thesis.

Featuring within the remit of software engineering, services, and cloud computing, the CACTOS proposal won funding during the FP7-ICT-2013-10 competitive call for research. This FP7 project ran for three consecutive years (October 2013 to October 2016) as a multi-national collaboration between Germany, Sweden, the UK and Ireland (European Union, 2017). The acronym “CACTOS” stands for **C**ontext-**A**ware **C**loud **T**opology **O**ptimisation and **S**imulation. The project examines cloud computing from a data centre operator perspective and, as the name suggests, is based around three core concepts:

1. *Context–awareness.* The models of system behaviour, expressed through cloud services and compute resource demand interdependencies, allows individuals to assess the impact of co-located placement and scheduled application workload. These models enable individuals to predict the future resource requirements at both virtual and hardware levels, which in turn, enhances the understanding of the quality of service and application service level agreement adherence demands.
2. *Topology optimisation.* Cloud data centre topology optimisation mechanisms are employed to create a self-managing system (Kephart and Chess, 2003) that is able to map compute resources to the VMs to satisfy user demand within other quality of service constrains. This includes objective driven VM placement and VM consolidation vertical and horizontal resource elasticity controls.

3. *Simulation.* The discrete-event simulation (DES) platform provides a means to evaluate the impact of topology optimisation strategies against the set quality of service and cloud service level agreement goals. DES techniques are used to model the large-scale heterogeneous data centre infrastructure and then simulate system behaviour creating the decision support foundations for evaluation and fine-tuning of resource optimisation parameters.

CACTOS addresses the challenge of managing modern cloud systems at its scale and complexity while taking into account different cloud workloads and infrastructure heterogeneity. The types of cloud applications vary from a simple single VM deployment to complex multiple VM deployments where the application is comprised of a cohort of geographically distributable logical components. Determining the behaviour of an individual application component and overall application performance in such complex configuration becomes an issue because of the non-linear relation of low-level system monitoring data (e.g. CPU utilisation, memory availability) to the high level QoS objectives (e.g. response time, cost, reliability). In addition, the cloud workloads and hardware heterogeneity play main roles in the cloud system behaviour modelling. Different types of applications utilise cloud resources in different manners; one can have short bursts of resource demands to serve large quantities of user requests, whereas another might have fewer users, yet job size can be significantly larger and require a greater amount of resources to process. Similarly, the demand of different types of resources will vary per cloud user as different types of computational tasks will require different proportions of CPU, storage, memory and network bandwidth. All these cloud system non-linear interdependencies must be considered when simulating and optimising cloud topology.

CACTOS creates an integrated framework that has the ability to effectively capture and analyse dynamic workloads; optimise virtual and physical resources (e.g. increase utilisation, reduce energy consumption); deploy management decisions into the cloud data centre; and provide comprehensive “what-if” decision analysis using simulation. When combined the project goals, ethos, and integrated tools provide rich research data sources that are used for both energy and cost analysis, the latter of which is the main research focus of this thesis. The context-awareness delivers real cloud data

centre system behaviour information derived from raw monitoring data. The topology-optimisation techniques demonstrate the direct effect of management decisions on QoS parameters. And finally, the simulation framework provides a mechanism for designing and building experiments for broader use case analyses. This scholarly work has direct research relevance due to its association with CACTOS as an EU approved ICT research project.

As mentioned, the work presented in this thesis was conducted as part of a collaborative initiative, known as the CACTOS project. The thesis author led work which focused on the tasks for simulation framework creation, such as extracting system requirements, software design and development, integration and result validation. Hence, the system infrastructure models (Section 3.6), overall software architecture design, tools development and integration (Sections 5.3.1, 5.3.2, 5.3.3 and 5.3.4) were completed collaboratively with CACTOS project partners. The thesis author participated in work across the project in order to deliver essential integration between project components, and obtain access to the data and tools, which were paramount to the research. For example, the simulation framework integrates with the data collection framework. The collaboration enabled the collection of data that was sufficient for building simulation models, thus ensuring the progress of simulation framework development. The same applied to integration with the optimisation framework; it was crucial to agree the format and synchronisation points of bilateral data exchange for simulation to “understand” optimisation steps and for optimisation to read simulation outputs. To summarise, the work presented in this thesis is intertwined with the overall CACTOS project tasks and final software artefacts, which largely focused on the work of simulation framework development. Acquired data and jointly developed tools from the project were then used in the thesis to fulfil the research hypothesis and objectives listed in Section 1.2.

3.2 Use cases

Cloud data centres are being used to host a wide variety of services, each of which has unique design and resource demand requirements. To capture such cloud application diversity CACTOS addresses requirements of three different scenarios: business analytics, scientific computing, and enterprise grade applications.

The business analytics use case presents a challenge by way of public cloud infrastructure providers optimising resource distribution when the type of the applications executed in the rented VMs is unknown. The challenge for the CACTOS project is to manage such systems by constructing resource demand prediction models via a monitored data analytics approach.

The scientific computing scenario provides an obstacle in managing workloads with high resource demand. CACTOS focuses on quantum chemistry software Molpro (Werner, Knowles, *et al.*, 2012) that is used for advanced molecular electronic structure calculations. These calculations can take hours and there is a risk of loss in the case of insufficient memory or storage availability. Fear of re-running experiments and losing time due to insufficient resources causes scientists to oversupply resources which reduces system usability. CACTOS aims to optimise resource provisioning for such workloads in order to free up unused resources for other users.

The enterprise application scenario looks at the resource management for large, multi-tiered cloud applications with functions distributed across multiple VMs. Such applications would have a load balancing component that distributes the load across multiple nodes eliminating bottlenecks due to lack of resources. CACTOS' role is to provide dynamic scale to enterprise application components depending on user demand and according to the SLAs in place.

3.3 Test beds

To aid the design and testing of CACTOS proposed resource management solutions, two independent testbeds were made available to the consortium members. The first testbed was divided into two clusters containing 3 and 8 compute nodes respectively and operated by Flexiant Cloud Orchestrator (FCO) (Flexiant, 2017) cloud management platform. The second testbed was comprised of a single cluster containing 20 compute nodes managed by OpenStack (OpenStack, 2017) cloud platform.

3.4 Methodology overview

CACTOS uses resource utilisation indicators out of monitored data and applies management actions for controlling distribution of cloud resources. Thus, the cloud data centre is modelled by using a sensor-actuator approach where the monitored sensor

data is contained within the infrastructure topology and the resource load models and the actuator model represents optimisation plans that consist of recommended optimisation actions that are being passed onto the system. Such a modelling approach is used to create a closed Observe-Plan-Act control loop, shown in Figure 3-1. This approach corresponds to the MAPE-K control loop described by IBM (please refer to section _ of the literature review) where the cloud data centre running applications and resources are being continuously monitored. Based on this monitored data, optimisation actions are derived and then enacted to meet certain QoS objectives.

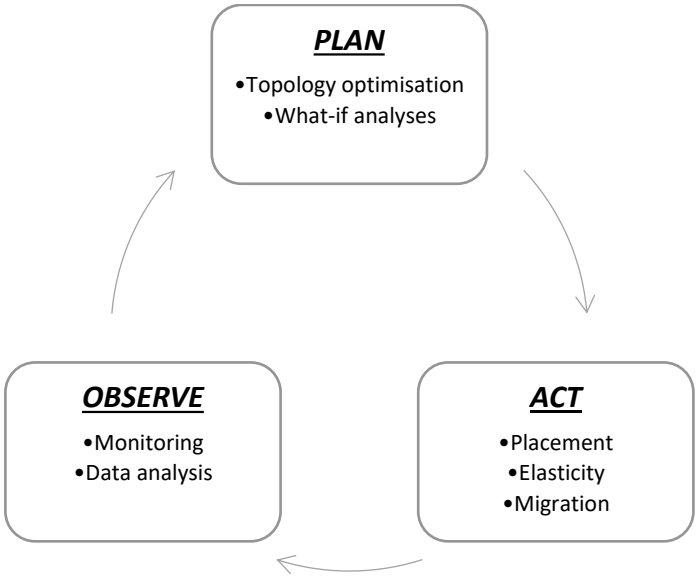


Figure 3-1: CACTOS closed Observe-Plan-Act control loop

Within CACTOS the observation step can be divided into two actions: monitoring and data analysis. During the monitoring phase cloud data centre data is being captured using infrastructure topology and load models. Infrastructure topology models describe the location and attributes of hardware, virtual machines, and applications. Load models contain the current system utilisation data at the hardware and virtual levels, capturing fine grained individual resource utilisation and capacity (e.g. CPU, memory, storage) as well as indirect properties like energy consumption. The data analysis creates application behaviour models using historical monitoring data or the benchmarking techniques to annotate application-level resource demand patterns. When building a cloud application behaviour model, it is important to capture the relation between application performance and the available resource capacity plus the dimension of the deployment characteristics

that is part of the cloud system. The CACTOS application models are built by analysing resource request patterns in the context of cloud environment deployment and execution.

The essence of optimisation planning is within intelligent arrangement of atomic cloud scheduling actions that lead to the intended system behaviour. However, planning complexity is significantly increased by the scale of the cloud computing system and the conflicting objective functions which should be satisfied at the same time. For example, one might want to maximise computational throughput, resource utilisation and cost efficiency, but at the same time minimize energy consumption, heat emission, application response time and cloud service SLA violations. To deal with these challenges, CACTOS breaks down cloud system topology optimisation into multiple levels. The employed optimisation solution is applying application level fine tuning whilst also taking a holistic view of the current and predicted system performances. Optimisation actions can be triggered during routine periodic system inspections as well as in reaction to failure or other QoS threshold indicators. To complement the real time self-organising decision support system, CACTOS also includes a simulation toolkit as means to further fine tune its algorithms and estimate system performance during various edge cases (e.g. power outage, resource demand bursts, resource failures). Simulation allows individuals to gauge the resilience of the cloud data centre optimisation strategy in different circumstances via “what-if” analyses before real system deployments.

Hypervisors offer full management controls that allow to dynamically start, pause, resume, scale, reconfigure, migrate and terminate virtual machines without high performance penalties over the exposed network API. CACTOS creates optimisation plan actions using the hardware-enabled virtualization techniques provided by the hypervisors (e.g. XEN, KVM). The optimisation plans are then submitted by the optimisation engine to the appropriate middleware implementation components, which in turn, integrate with cloud platforms like OpenStack (Open Stack, 2017) and Flexiant Cloud Orchestrator (Flexiant, 2017).

Planned project methodology was executed in three distinct steps: development of tools, definition of cloud infrastructure models, and integration. Tools were created to fit the *Observe-Plan-Act* control loop which allowed for system continuous monitoring analysis and self-management through optimisation objectives. The cloud infrastructure models were created to serve as a data exchange standard among all the software components. After defining the infrastructure system models, tools could be integrated

with each other and used within third party environments for data collection and cloud platform control.

3.5 Tools

The CACTOS project methodology is designed to ensure cloud data centre resource autonomic management by employing the *Observe-Plan-Act* control loop. To execute these control steps, bespoke software components (tools) were designed i.e. data collection framework - CactoScale, optimisation framework - CactoOpt and simulation framework – CactoSim (see Figure 3-2). Integration of all three frameworks provides a coherent cloud data centre context-aware topology optimisation solution.

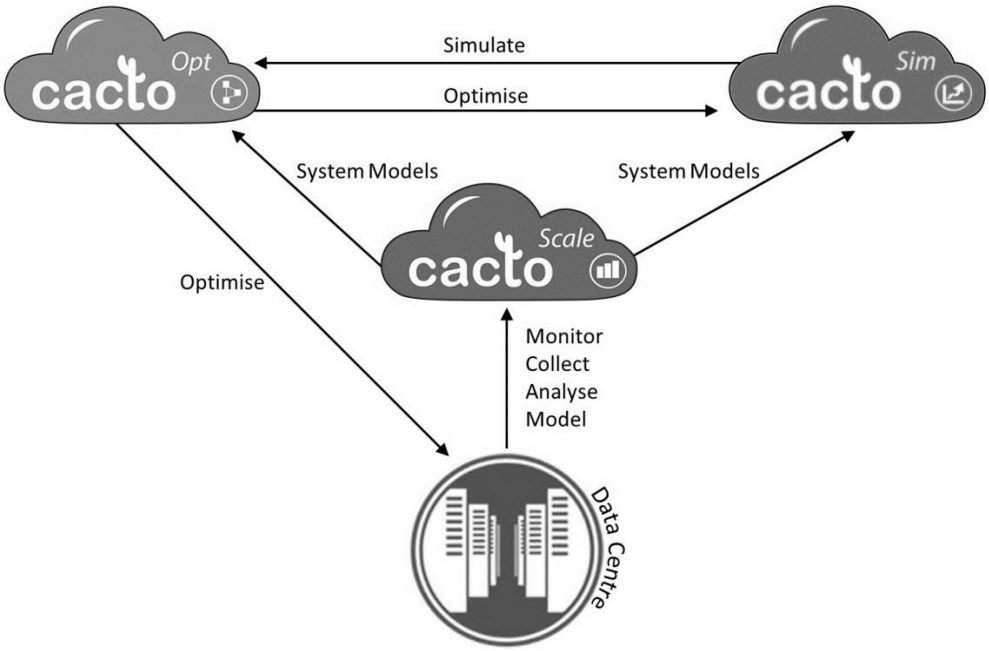


Figure 3-2: CACTOS tooling overview

As shown in the high-level tooling overview diagram in Figure 3-2 the cloud data centre is constantly monitored by CactoScale. The monitored data, such as system performance, virtualization configuration, and services behaviour, is being collected and analysed in real time. From the data analysis, models of the system are created. These models then can be used by CactoSim and CactoOpt. CactoOpt periodically retrieves system models to decide whether any optimisation actions are needed and, if so, optimisation suggestions are produced for the data centre as a resource management strategy. CactoSim uses system models to build simulation experiments and can use

CactoOpt optimisation policies within these experiments to analyse system performance impact.

Cloud computing data centres are complex systems with multiple interconnected hardware and virtual components that create service critical interdependencies. To manage such intricate systems cloud data centre operators must have access to highly scalable tools allowing collection and processing of information on vital system components.

3.5.1 Data collection framework (CactoScale)

The data collection framework, CactoScale, was designed to address the need for a scalable solution capable of processing live data streams and system logs that describe the state of every vital system resource i.e. measurements of CPU, memory, I/O devices, network and energy.

Different types of resources have different approaches to measurement and measurement metric characteristics. CPU utilisation is measured within time intervals by looking at the percentage of time the allocated CPU core spends on processing instructions. For example, if the measured time interval is 1sec and within this time probe 0.5sec CPU was busy processing tasks then it can be said that CPU utilisation is 50%. Memory performance indicator characterises the amount of memory available and the memory used by the running applications. Memory bandwidth can also be determined using integrated memory controllers that count the number of bytes that is written or read to the memory modules. Storage performance is measured in I/O operations per second (IOPS). The metric captures the number of reads and writes that storage locations can handle per second while also accounting for seek time delay of the non-contiguous data location. Network performance is measured by clocking the time it takes for the packet to reach its destination, which is referred to as latency. Each network link has its bandwidth capacity and if it is highly utilised higher latencies can occur resulting in delays of service delivery. Power consumption data can be extracted from the system by using external digital multimeters that measure voltage and current, or by probing embedded sensors of different components. For example, Intel introduced Average Power Limit (RAPL) (David, Gorbatov, *et al.*, 2010) sensors that allow low level measurement of CPU and memory power consumption.

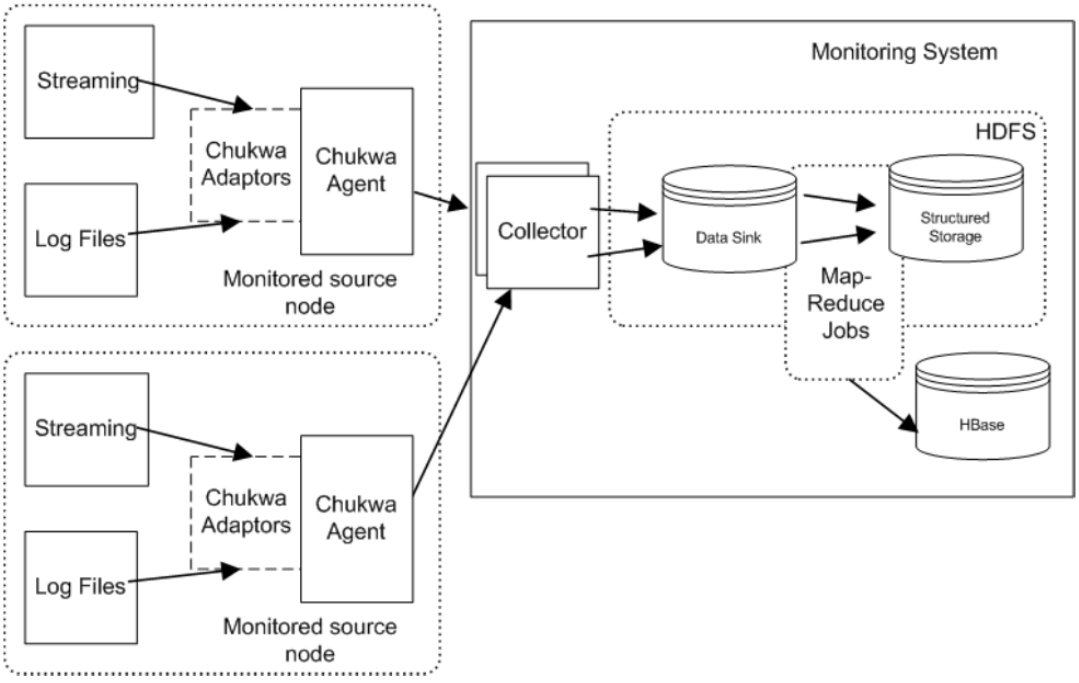


Figure 3-3: CactoScale Chukwa monitoring tool (Papazachos, Bharbuiya, *et al.*, 2015)

CactoScale is based on the Apache Chukwa and Hadoop Distributed File System (HDFS) and so inherits all its functional features such as distributed processing of large datasets across multiple computer clusters; the ability to scale from one to thousands of machines; and the provision of high availability by detecting and remediating any occurring failures within the Hadoop cluster (Apache Software Foundation, 2017). Chukwa provides flexible data monitoring, analysis and display capabilities. The design of Chukwa, as shown in Figure 3-3, enables capturing system performance data from multiple sources at each monitored node within the cloud data centre. Next, the Chukwa agent sends the aggregated data to the available collector. Finally, collectors then send the data to the HDFS for processing and storage.

3.5.2 Optimisation framework (CactoOpt)

CactoOpt is a scalable, real-time cloud data centre optimisation framework. CactoOpt is designed to fulfil optimisation goals such as decreasing operational costs of the cloud data centre, increasing data centre generated profits, and keeping customers satisfied with the provided service. These high-level goals are broken down into more detailed objective functions that are implemented within CactoOpt:

- The **Load Balancing** function seeks to minimise the load of the specific node in the data centre and the overall average load of all nodes by distributing workload evenly across all available compute infrastructure. This function was designed to ensure that quality of service is met and VMs' access to resources is not contended.
- The **Consolidation** function seeks to minimise the number of physical nodes used for hosting VMs without overbooking existing physical resources and maximise number of free hardware nodes to accommodate future system load.
- The **Energy Efficiency** objective function seeks to minimise power consumption of the cloud data centre. On par with consolidation this function minimises the number of nodes being used by the system while also considering the energy consumption properties of the available hardware. This means choosing more energy efficient hardware to run the workload while other nodes can be shut down or put to sleep.
- The **Resource Fragmentation** objective function strives to minimise resource underutilisation by reducing amount of stranded resources (e.g. where all the CPU cores on a node are booked by hosted VMs but some free memory remains). The function merges the resource leftovers to form bigger resource chunks suitable for other VMs to use.
- The **QoS Violations** objective function seeks to minimise the number of QoS violations that occur for the delivered cloud service. Demands of running services can be translated into amount of resources they need to possess to achieve desired performance. The function ensures that the appropriate resources are provided to minimise QoS violation numbers.

To achieve the set objective functions, the optimisation framework must continuously balance resource provisioning to cater for the planned events, predict re-occurring events, and react to unpredicted events. The planned events are known events that are initiated by a data centre operator usually strategically scheduled at the time of lower workload. Planned events can include, but are not limited to, hardware maintenance, hardware upgrade, and software updates where the nodes need to be taken completely offline. Given the known time of such events, the optimisation

framework can prepare and react in advance. The predicted events such as weekly workload pattern change can also be managed proactively. Knowing the workload patterns, CactoOpt can gradually provide or remove resources in advance to meet the demands without QoS degradation. The unpredicted events, such as hardware failures, flash crowd resource demands and requests for new VM admission from IaaS customers, can only be handled in a reactive manner. CactoOpt is designed to detect unpredicted events early and apply appropriate system remediation or reconfiguration actions.

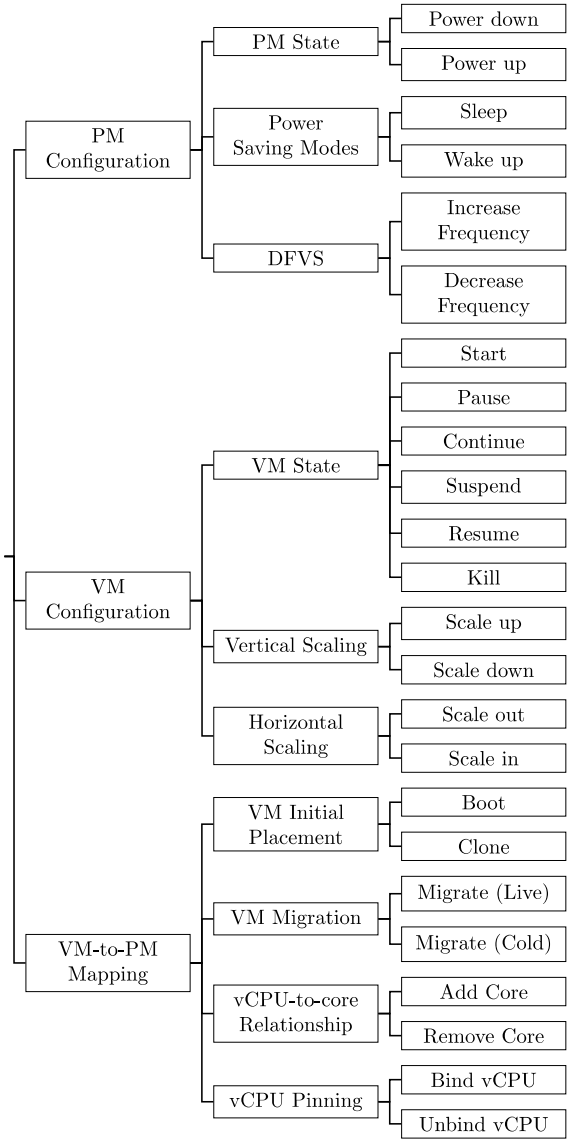


Figure 3-4: Actuators taxonomy (Krzywda, Rezaie, *et al.*, 2015)

To fulfil the objective functions while simultaneously dealing with the described events, the optimisation framework has access to a list of actuator actions shown in Figure 3-4 in a form of hierarchical taxonomy. CactoOpt has control over three main categories of actuators within the cloud data centre: Physical Machine (PM) configuration, VM configuration, and VM to PM mapping. PMs can be powered up or down for maintenance, put in or out of sleep mode for energy saving and its CPU and peripherals performance can be adjusted via Dynamic Voltage and Frequency Scaling (DVFS) to avoid overprovisioning also saving energy. VM configuration actuators allow to change between VM states, add more resources via vertical scaling, and load balance cloud services by adding more VM instances via horizontal scaling. The actions of VM to PM mapping govern the initial VM placement actuators such as booting or cloning VMs, migrating VMs from one PM to another, and managing access to CPU cores by adding and pinning specific cores to VMs.

3.5.3 Simulation framework (CactoSim)

As state previously, the part-development and use of CactoSim as well as integration with other components form part of the main outcomes/objectives of this thesis, and CactoSim presented in this section for completeness as part of the overall CACTOS project. CactoSim is built on top of two open source solutions, a software architecture simulation tool called Palladio (Palladio, 2017) and its plugin called SimuLizar (Becker, 2016) which adds support for self-adaptation rules modelling.

Palladio is a toolkit for modelling and testing software architecture implemented in Ecore using the Eclipse Modelling Framework (EMF) (Brosch, Koziolek, *et al.*, 2012). The core of this approach is Palladio Component Model (PCM) which allows for the modelling of entities for component-based software systems that can be linked together as complex complete software solutions and tested for performance and reliability. The performance and reliability predictions are made using a resource demanding service effect specification (RD-SEFF) which includes dependencies between provided and required services of a component, notions of resource usage, data flow and parametric dependencies (Reussner, Becker and Happe, 2011).

Palladio bases the performance of the component-based software system upon four factors:

- Implementation of the component

- Required services
- Deployment platform (i.e. hardware, middleware, networks)
- The way components are used

Each of above factors can be modelled in specific sub-models and commutatively form a PCM instance as shown in Figure 3-5. The Component Model consists of component specifications including a model of their behaviour. The Composition Model describes the structure of the system by the composition of the components to be used. In the Deployment Model the components of the system are allocated to physical resources. Finally, the Usage Model contains the workload induced by the system's end-users.

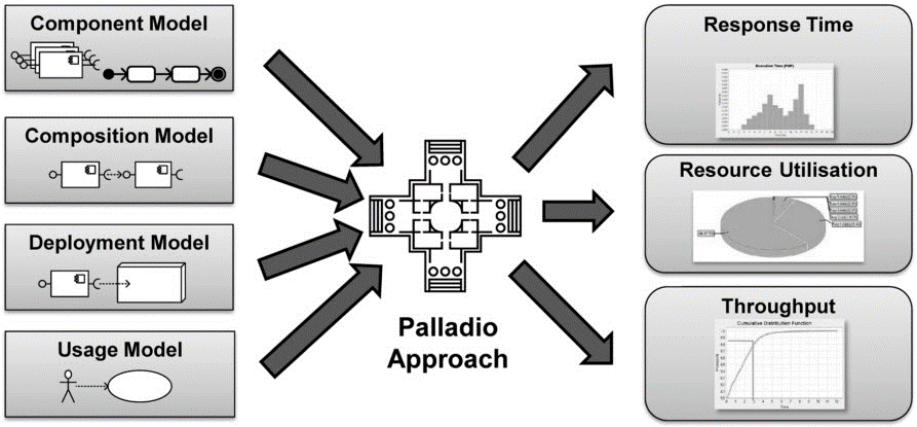


Figure 3-5: Palladio overview (Palladio 2017)

As a simulation output, PCM produces analysis information which supports the evaluation of different modelled system performance attributes including response time, maximum throughput, resource utilisation, and Quality of Service (QoS) levels which align with the previously listed four performance factors (Rathfelder and Klatt, 2011).

SimuLizar is a Palladio plug-in under development with a self-adaptive system modelling approach based on the PCM. It introduces two new modelling artefacts:

- Palladio Measurement Specification (PMS) – domain specific language describing the data collection points called sensors.
- Self-adaptation rules – used to define conditions and actions for model adaptations.

Implementation and functions of SimuLizar can be explained via the diagram given in Figure 3-6.

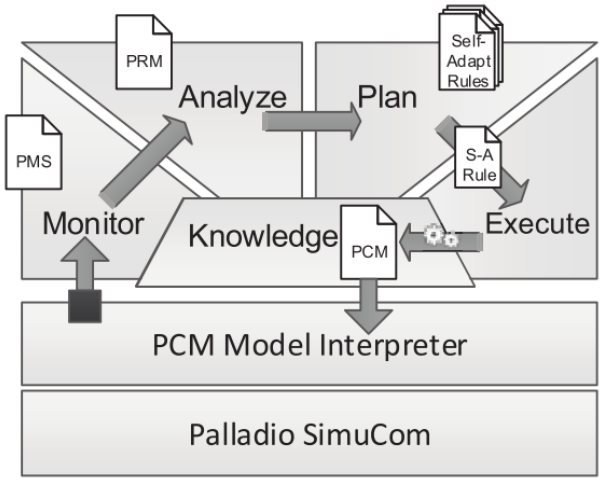


Figure 3-6: SimuLizar architecture (Becker et al. 2013)

During each step of the simulation process, PMS defined data from sensors is being written to the Palladio Runtime Model (PRM) container. The update of PRM executes a planning phase by launching self-adaptive rules. If the conditions defined in self-adaptive rules are met, then the PCM adaptations are executed. After the PCM transformation, the next simulation step commences using a new adapted model version (Becker, Luckey and Becker, 2013).

Apart from the detailed software modelling capabilities, PCM provides the means to model a nested hardware environment. This is an important feature for cloud computing data centre modelling because in a real environment, hardware nodes are placed into physical racks and divided into clusters, while PCM provides a logical resource division or grouping based on resource types i.e. a cluster containing only ARM processors.

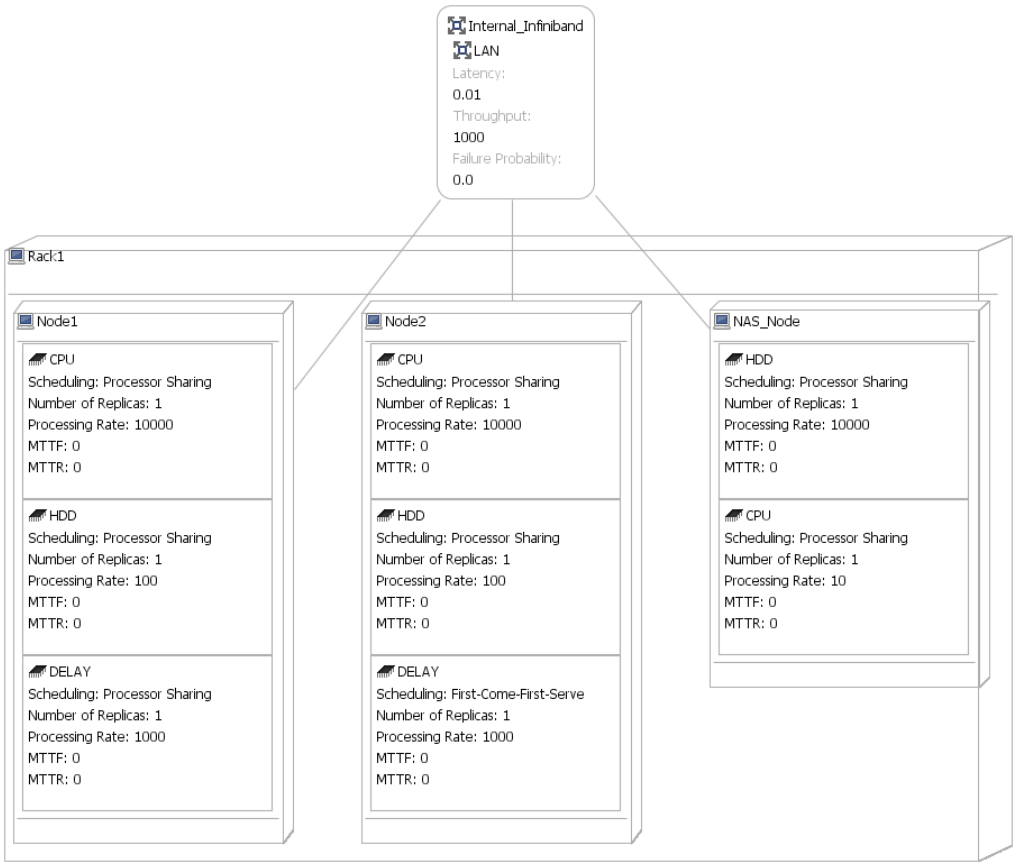


Figure 3-7: Palladio hardware topology model

Figure 3-7 shows a simplified example of a hardware topology model of a single rack inside the data centre. The “Rack1” model consists of three nodes, “Node1”, “Node2” and “NAS_Node”, which are interconnected by the “Internal_Infiniband” network. In similar fashion, the hardware topology model can be scaled to mirror configuration of thousands of nodes inside the large data centre.

CactoSim plays an important role within the CACTOS project by providing an analysis tool that is capable of forecasting system behaviour beyond the scale of available testbeds. In such a manner, the cloud data centre operator is able to have a simulation based wholesome system analysis to determine possible performance bottlenecks under various controlled experimental conditions e.g. unforeseen spike in user demand, critical hardware failures. Simulation data can help to pinpoint weaknesses in the system configurations ahead of real system deployments, thus minimising the chance of a system performance and QoS degradation.

3.6 Cloud infrastructure models

The CACTOS cloud infrastructure models capture configuration, properties and attributes of cloud data centre infrastructure physical and virtual (logical) components. The models describe the content and dependencies of the physical hardware layer and the virtualised layer and capture resource utilisation measurements of both physical and virtual layers. Further, models are used to communicate system state information across all three tools - CactoOpt, CactoScale and CactoSim; using the same data structure allows avoidance of data inconsistencies during information exchange and reduces integration efforts.

The CACTOS Infrastructure models were developed continuously throughout the project using a Model-Driven Software Development (MDSD) process and implemented using an Eclipse Modelling Framework (EMF) (Eclipse Foundation, 2018). The MDSD approach allows individuals to define the meta-model of the system, which contains relationships between model components and attributes, similarly to class diagrams used in software design.

As alluded to earlier, the scale of the system requires a large degree of automation to populate and update data contained within the models. Also, the use of simulation framework output is fed to the optimisation framework to substitute real system data flow. However, it is human agents that are expected to extend, parameterise and make decisions on the final experimentation results. The model implementation decision using MDSD and EMF technologies allows for models to be used both by human agents and software driven agents like CactoOpt and CactoScale through the integration interfaces.

3.6.1 Physical data centre model

The Physical Data Centre Model (PDCM) holds the structure of the cloud data centre physical infrastructure. The model components are arranged in the same hierarchical order where the root of the model is a definition of a single data centre entity that contains nested racks under which lies the node entities. Network components are realised in the form of switches that contain a list of links between other connected entities. Switches can be positioned at any level of the model to allow flexibility in line with the different networking options available in real systems.

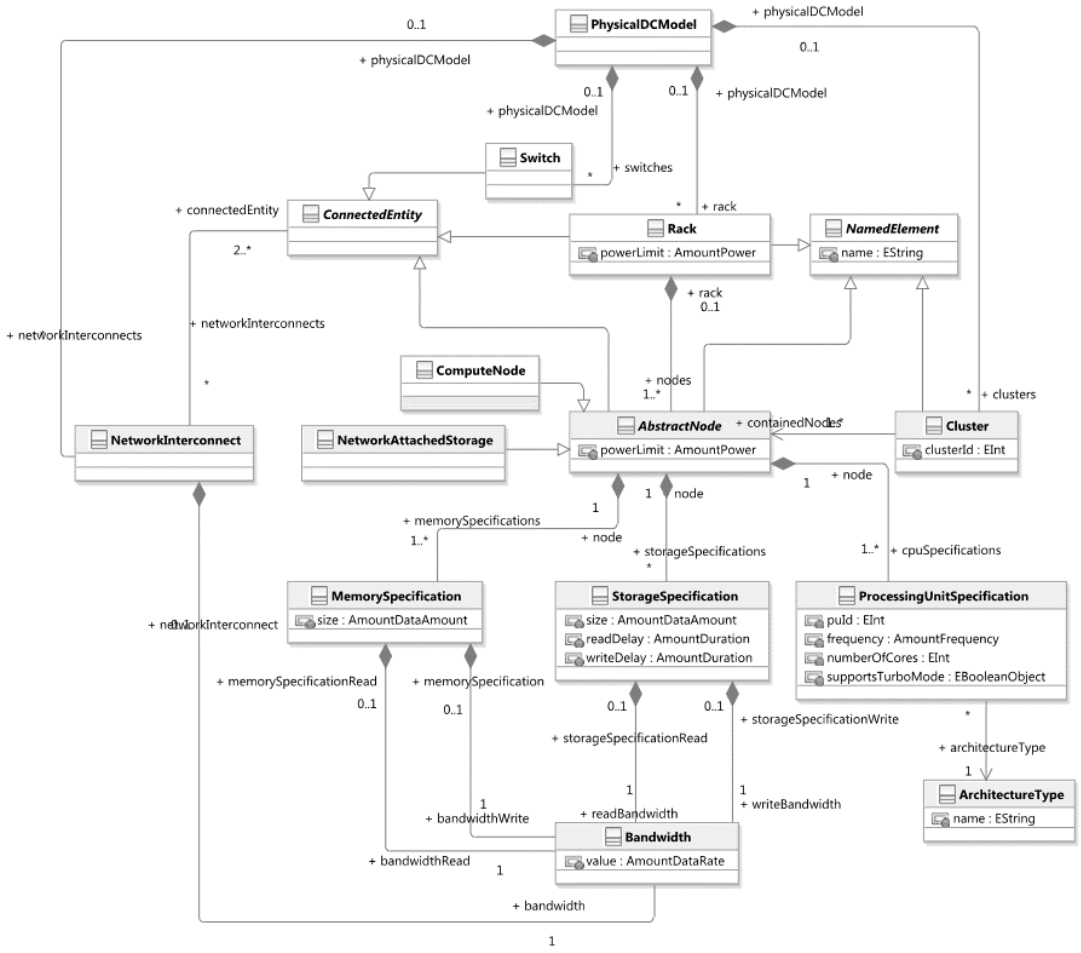


Figure 3-8: CACTOS core Physical Data Centre Model (PDCM) (Groenda, Stier, Östberg, *et al.*, 2014)

As shown in the architectural model blueprint in Figure 3-8 the focus of the model lies in capturing processing capabilities of physical nodes inside a data centre. The main cloud resources i.e. CPU, memory and storage are described respectively as MemorySpecification, StorageSpecification and ProcessingUnitSpecification classes. MemorySpecification holds attributes of available memory size and the read and write bandwidth. The StorageSpecification class holds the same attributes as MemorySpecification plus attributes for capturing the metric of read delays and write delays that are relevant to Hard Disk Drives (HDD). The hardware capabilities of a CPU are contained in the ProcessingUnitSpecification class which captures CPU frequency, number of available cores, turbo boost support, and type of architecture e.g. X86, X86_64, ARM.

3.6.2 Logical data centre model

The Logical Data Centre Model (LDCM) captures the configuration of a logical layer of cloud data centre which is created by infrastructure virtualisation. LDCM also maps hardware resources described in PDCM to the virtual resources linked through the hypervisor component.

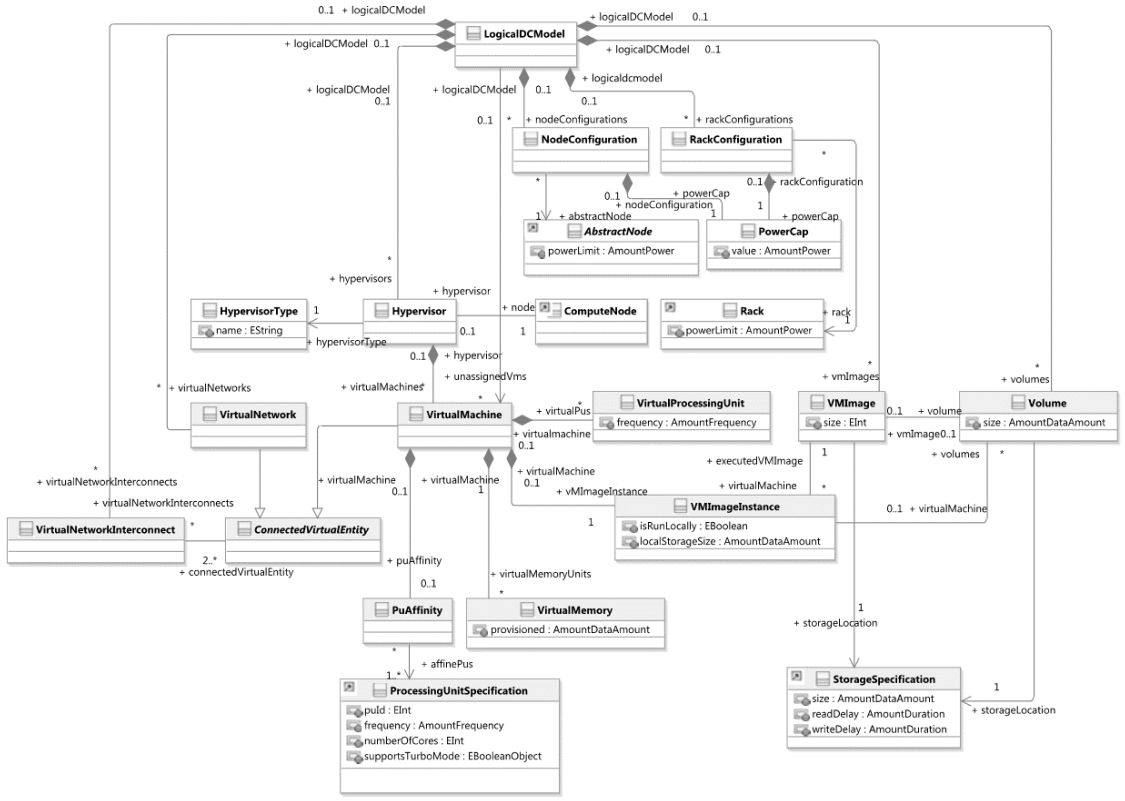


Figure 3-9: CACTOS Logical Data Centre Model (LDCM) (Groenda, Stier, Östberg, *et al.*, 2014)

The LDCM has a dual purpose: first is the description of VM characteristics and second is the description of virtual components overlay over physical hardware. The class diagram in Figure 3-9 portrays the LDCM structure that centres around the VirtualMachine class objects and its attributes. Real system VMs can be linked to one or more physical CPU units as reflected in the model where the ProcessingUnitSpecification from PDCM is linked by using PuAffinity model class. Virtualized CPU, however, is modelled as VirtualProcessinUnit class with the assigned frequency attribute. The bootable image of VM is described using VMImageInstance class that defines the size of the storage, the location of execution, and the link to the

StorageSpecification of PDCM. The virtual network components are also covered by the logical model in a similar fashion which describes logical links between the virtualized components and the relation between the virtual network and the physical network topology.

3.6.3 Physical load model

CACTOS Physical Load Model (PLM) captures the load levels of physical processing resources via the utilisation metrics. The PLM also overlays the hardware components described in the PDCM, meaning that every single recorded measurement has to have an underlying hardware resource.

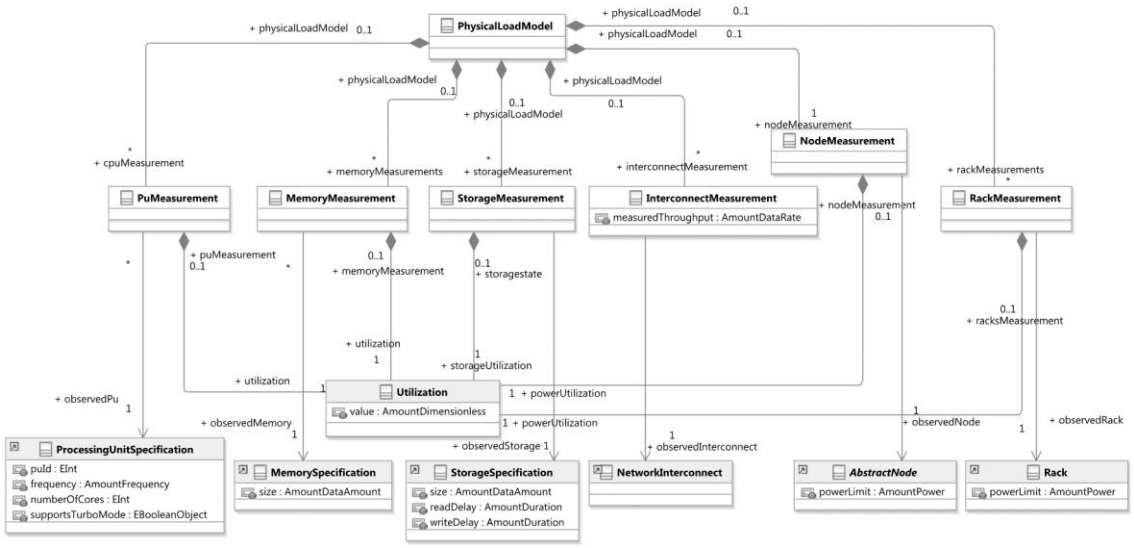


Figure 3-10: CACTOS Physical load model (Groenda, Stier, Östberg, *et al.*, 2014)

As shown in Figure 3-10, the PLM focuses on the hardware resources by having model class entities for CPU, memory, storage, network and power measurements. The measurement classes are all linked to the Utilisation entity that has a Value attribute capable of holding different types of dimensions e.g. MB/s, GB, MIPS.

3.6.4 Logical load model

Similar to the PLM, the CACTOS Logical Load Model (LLM) captures the utilisation of virtual system resources within each VM. These virtual resources consist of virtual memory, virtual processing units, storage volumes, and virtual network connections topology between VMs.

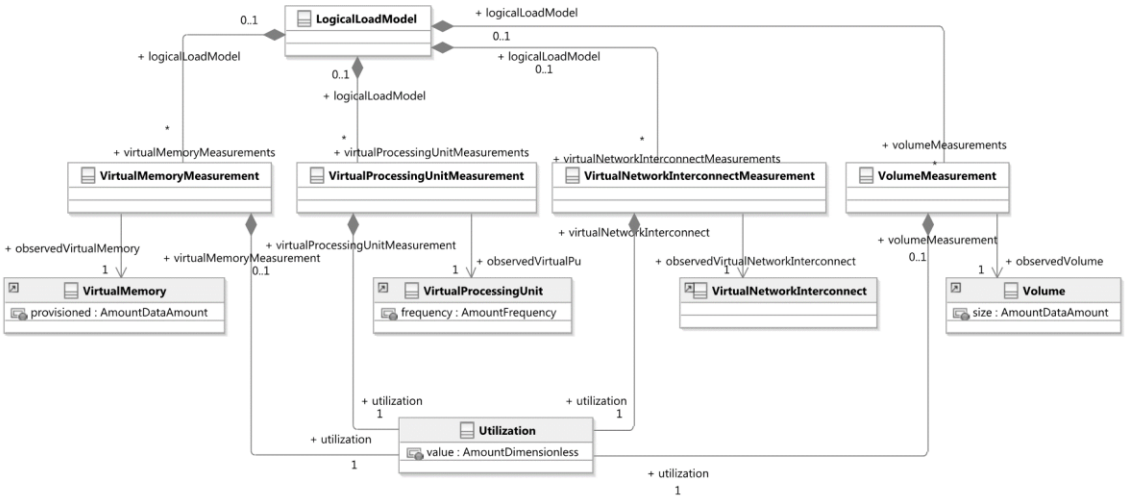


Figure 3-11: CACTOS Logical load model (Groenda, Stier, Östberg, *et al.*, 2014)

The LLM diagram shown in Figure 3-11 captures resource measurements using class components of VirtualMemoryMeasurement, VirtualProcessingUnitMeasurement, VolumeMeasurement, and VirtualNetworkInterconnectMeasurement. These mentioned model classes contain links to Utilisation class which holds the actual corresponding resource utilisation value.

3.7 Integration

As described in chapter 3.5, the CACTOS project delivers three tools for cloud data centre management – CactoScale, CactoOpt and CactoSim. The CactoScale tool is designed for collecting information about cloud data centre physical and logical infrastructure topology and current utilisation state. CactoOpt optimizes cloud physical and virtual system layers based on a set of objectives. Finally, CactoSim is able to simulate large scale system behaviours by estimating cloud data centre QoS performance, energy, and cost efficiency beyond testbed size limitations.

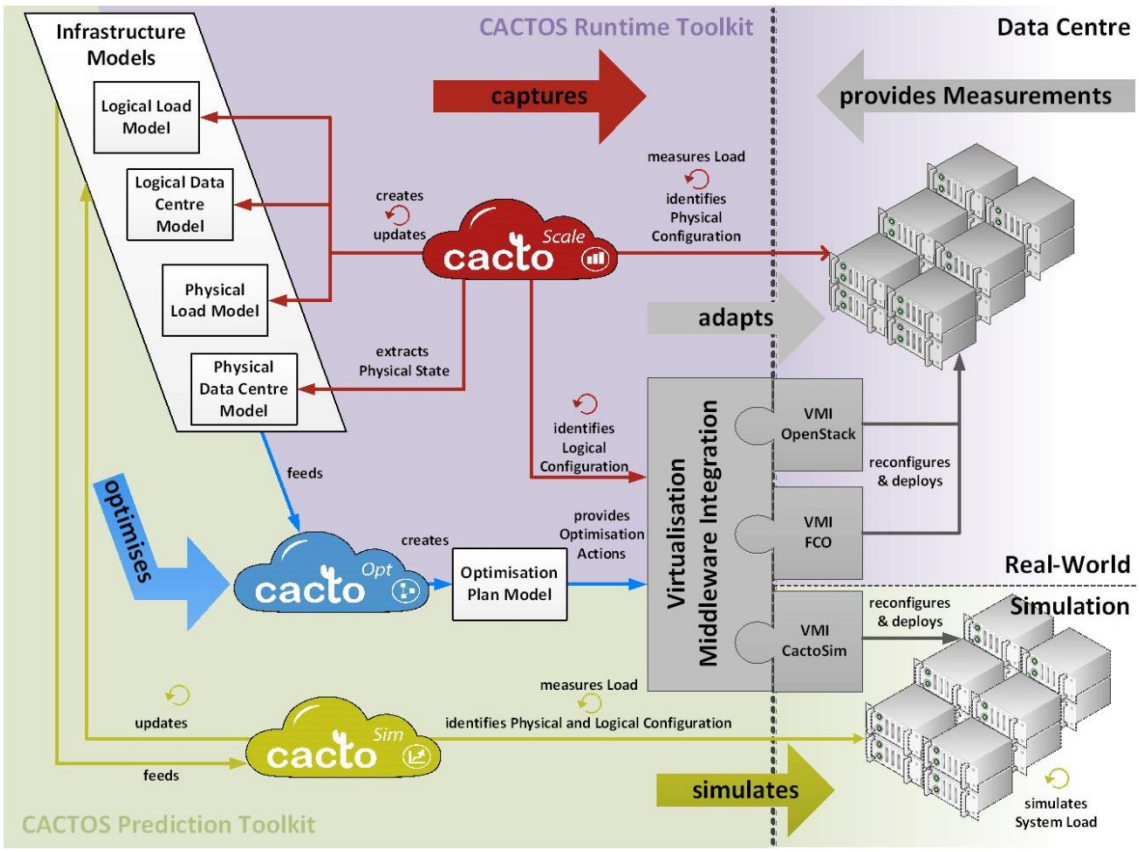


Figure 3-12: CACTOS tooling integration landscape overview (Groenda, Stier, Krzywda, *et al.*, 2014)

From the beginning of the CACTOS project, the CactoScale, CactoOpt and CactoSim tools were built with mutual integration in mind which was enabled by the use of Runtime toolkit and Prediction toolkit. Both Runtime and Prediction toolkits rely on the data collection framework to provide information about the system that is being optimised or simulated.

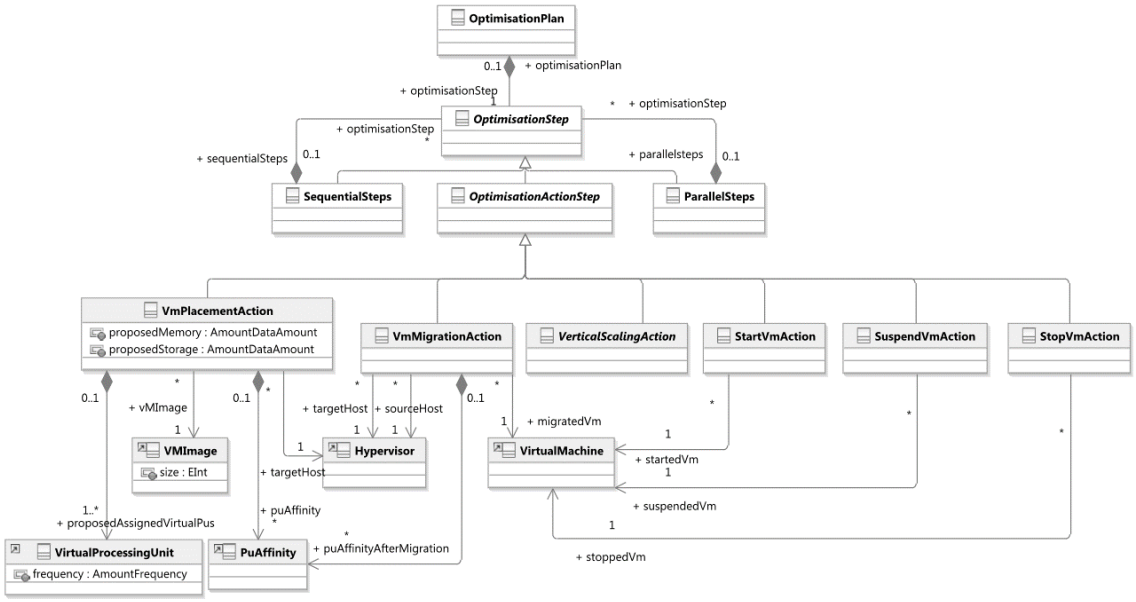


Figure 3-13: Optimisation plan model (Krzywda, Ali-Eldin, *et al.*, 2014)

The CACTOS Runtime Toolkit is designed to be integrated into the live cloud data centre system and operate in real-time. As shown in Figure 3-12, CactoScale periodically measures system load and collects system configuration data. The collected information is then populated in adherence with the structure cloud infrastructure models (i.e. PDCM, LDCM, PLM and LLM) and managed via the Eclipse Connected Data Objects (CDO) (Eclipse Foundation, 2017) model repository. CactoOpt pulls the infrastructure models from the CDO repository and calculates whether optimisation actions are needed based on the selected optimisation objective. If the optimisation algorithm deems optimisation actions necessary, the optimisation plan, which contains optimisation action steps, is created. The optimisation plan is also realised as an EMF model (as shown in Figure 3-13) and is designed to correspond to the actuators taxonomy described earlier in Figure 3-4. To translate the optimisation actions into corresponding cloud management system commands, the Virtual Middleware Integration (VMI) adapters were developed for FCO and OpenStack testbed deployments available within the project. In such a way, CACTOS Runtime Toolkit is able to create a control loop capable of autonomous cloud data centre infrastructure optimised management.

The CACTOS Prediction Toolkit which is based on CactoSim simulation framework is designed to be used offline and in concurrence with the running cloud data centre system. As shown in Figure 3-12, the simulation can be used seamlessly to run

optimisation policies as if they were executed in a real data centre environment. This is achieved by seamless integration between the data collection framework CactoScale, optimisation framework CactoOpt and simulation framework CactoSim. CactoSim is able to fetch system models (PDCM and LDCM) from the CDO repository populated by CactoScale and use them to build the simulation experiment. Since CactoSim is based on the Palladio simulation framework, the model-to-model transformations are dynamically executed between CACTOS cloud infrastructure models and Palladio Component Models which ensures seamless model continuity (Stier and Groenda, 2016). Since the optimisation framework is running in real time and simulation running in the simulated time there must be a synchronisation stop performed while optimisation algorithms are executed. Hence, during a simulation run, the CactoSim is using SimuLizar “Check and Execute” method which allows the simulation to be paused and all the simulated results at that particular simulation time to be accessed. During the pause, CactoSim creates load prediction models (LLM and PLM) based on these simulated measurements and sends these models to CactoOpt. Since CACTOS uses the same infrastructure models in runtime and during design-time, CactoOpt produces a runtime identical optimisation plan (shown in Figure 3-13) based on the simulated data. Next, the optimisation plan is enacted within simulation by using CactoSim VMI adapter. Finally, the simulation, which now reflects the implemented optimisation decisions, is resumed. This data exchange between CactoSim and CactoOpt is performed periodically within the duration of the simulation experiment, as it would be executed in the real system. This way, CACTOS Prediction Toolkit is able to simulate large scale self-adaptive cloud data centre systems by taking into account resource management decisions solicited by the optimisation framework. This allows the efficiency of different optimisation policies in regard to the costs, energy efficiency and QoS constraints to be determined during design-time and before physical deployment.

3.8 Conclusions

CACTOS is an ICT (software engineering, services and cloud computing) themed project founded by the EU. The project, which ran over a three-year period (2013 to 2016), aimed to create context-aware cloud topology optimisation and simulation solutions for improving cloud data centre resource management. This aim was achieved by generating integrated Runtime and Prediction toolkits which can interface with the live

data centre. The runtime toolkit is tasked with executing real-time optimisation actions in order to improve cloud resource management. Whilst the Prediction toolkit is designed to execute design-time simulation based “what-if” analysis experiments in order to predict system behaviour under different optimisation policies prior to real system deployment.

The CACTOS project is integral to this study as it provided access to real data, tools, experiment testbeds, and most importantly, to the experts within this area. By collaborating with the industry partners of the CACTOS consortium, the issues of cloud data centre resource management are formulated into use case requirements. The requirements contain detailed data about system set-up and monitoring data extracts which form the basis of the project and of this study. During the project lifetime, a set of tools were created and integrated in order to address the issues earlier defined by use case owners. Series of test scenarios were then executed using project testbeds to ensure the legitimacy of provided solutions, enriching existing datasets with the experimental results. The project was accomplished through close collaboration between industrial and academic partners who are experts in the cloud computing area and who have generated a substantial number of academic peer reviewed application papers. More so, project progress was closely reviewed on a yearly basis by the EC assigned expert panel in order to ensure the quality of the produced output. The use of CACTOS as a platform for this study ensured its relevancy to current issues within the cloud computing industry, its validity in regard to used data and tools output, and its quality as evidenced by continuous evaluation of field experts.

The thesis research is based on the tools and models collaboratively developed within the CACTOS project by consortium members. The proposed simulation-based framework integrates tightly with the data collection and optimisation framework, hence a large degree of joint effort was needed to align system models and software architecture of project components.

4 Methodology

4.1 Introduction

This chapter outlines the steps taken to address the research challenges outlined in Chapter 0. A research methodology provides a plan for solving the proposed research problem (Rajasekar, Philominathan and Chinnathambi, 2014). Section 4.2 provides an overview of the research method used within this thesis and Section 4.3 describes the data collection approach that was implemented at different steps of the methodology.

4.2 Research methodology

The primary objective of this thesis is to demonstrate how cost analysis of resource allocation policies can benefit cloud data centre management decisions. The contribution of the thesis is the development of an advanced simulation framework followed by the introduction of an innovative aspect of cloud data centre cost modelling. This chapter describes the methodology which underpins the research presented in this thesis.

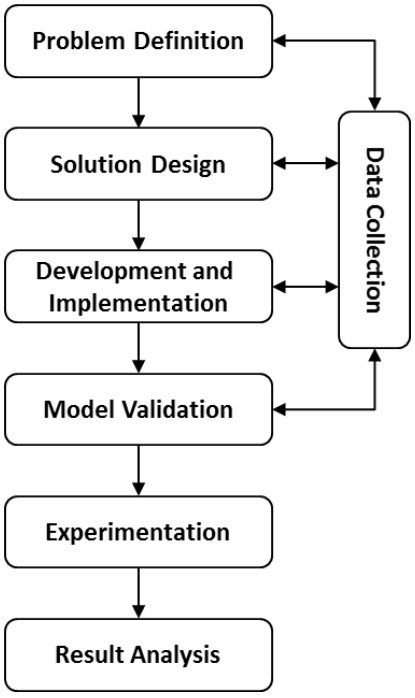


Figure 4-1: Research method

As illustrated in Figure 4-1, the methodology represents a process that is defined by a number of key process steps as outlined hereafter:

1. *Problem Definition*. The problem definition phase is required to collect all the relevant information about the issue that needs to be addressed. By analysing the collected information, the problem can be clearly described in detail which enables the identification of a research gap. The research challenges were identified by the literature review and case study analysis. The literature review of the cloud computing self-aware resource management domain is presented in Chapter 2. This literature review serves as a source of concept definition while also providing a state-of-the-art overview of the field. The case studies are defined by engaging with the industry stakeholders through CACTOS project collaboration (see Chapter 3). The case study data collection elicitation effort and techniques are described in Section 4.3.

2. *Data Collection*. The data collection step was executed in parallel with most of the research stages. As shown in Figure 4-1 each of the related methodology steps have a bidirectional connection with the data collection procedure. Based on the stage of the research, different types of data were collected about the use case and the cloud data centre environment. The received data then shaped the work direction taken in each thesis step. More information on the elicitation techniques and the collected data is provided in Section 4.3.

3. *Solution Design*. This step represents a blueprint of the simulation framework design, including its methods, system architecture and the action plan needed to resolve the defined problem. It also outlines the information format and direction flow between components as defined within the proposed architecture design. Since the simulation study is at the core of this thesis, the solution design step also includes an overview of key simulation model development stages outlining the conceptual system model design. More detailed description of the solution design step is presented in Chapter 5.3.

4. *Development and Implementation*. During the development and Implementation step, the designed solution was created in the form of a prediction toolkit. The toolkit components can be divided into three distinct software artefacts: programmed model, simulation framework and integration with data collection, and optimisation frameworks. The programmed model refers to the implementation method of the system model that holds the components and attributes of the designed conceptual model. The simulation framework refers to the implemented software stack representing a coherent DES platform capable of simulating cloud computing data centre behaviour. The creation of integration pathways between data collection and optimisation toolkits was also a crucial step in the solution development process by connecting simulation with real-system data and resource management approaches. The development and implementation phase also describes the software development methodology and code management tools used throughout the process. Created software artefacts are required to comply with certain scalability requirements in order to be able to simulate large scale cloud data centre systems; such tests are also carried out as a quality control procedure. Further details on software artefacts and the chosen software development and implementation process can be found in Chapter 5.4.

5. *Model Validation*. This step provides an evaluation of how accurate the simulation models are compared to real-world system behaviour. The validation is performed by designing a set of controlled experiments that represent actions defined within the problem definition stage. These experiments are then executed in the controlled environment of available testbeds and subsequently also in the simulated environment. Collected results from both real-system and simulation are then validated side-by-side to determine if simulation results are accurate enough to represent real system behaviour (See Chapter 6.2 for more details).

6. *Experimentation*. During the experimentation phase, multiple simulation-based experiments are designed and carried out to test representative scenarios. Firstly, the validated simulation models are extrapolated to fit

the proportions of a real cloud data centre size; and secondly, different resource optimisation policies are applied within each designed experiment. By keeping all the cloud computing data centre physical and virtual infrastructure system configurations consistent, and varying only the optimisation techniques, the optimisation effects can be isolated and compared.

7. *Result Analysis*. At the final stage, the findings are then analysed through side-by-side performance comparison. The performance is presented as QoS measurements such as energy efficiency, cost and resource utilisation. The results are presented in Chapter 6 and the final conclusions of the thesis can be found in Chapter 7.

This academic work is based on an in-depth understanding of the cloud computing data centre domain, management tools, costs, and the associated challenges. The data collection effort serves as a foundation upon which the whole study lies and shapes the direction of work throughout the thesis. To assure study validity, it is important to describe the data collection approach underlining the elicitation techniques more comprehensively. This is presented in the following section.

4.3 Data collection approach

The primary case study data sources for the thesis were provided by the CACTOS project (described in Chapter 3). The challenges within the thesis came from the presented project case studies' requirements, toolkit development, and integration requirements. The initial high-level description of concepts and issues, encountered by practitioners in operating a cloud data centre, were defined during the CACTOS project proposal writing phase. The consortium was formed with an intention to solve the defined problems by proposing a work plan and a general approach for solving defined issues. After the project proposal was accepted and funding was granted by the EU, a deeper, more detailed definition of the challenges and solutions had to be defined in order for project work to commence. The initial project proposal, literature review (see Chapter 2), and further data collection techniques manifested into a solid foundation for the thesis problem definition. Later, more detailed and case study specific data was obtained using a list of elicitation techniques during solution design, development and implementation, and model validation stages.

The list of elicitation techniques and data capturing formats were chosen based on the requirements elicitation survey by Zowghi & Coulin (2005) and techniques' descriptions by the International Institute of Business Analysis (2015). The chosen elicitation techniques include:

- Interviews
- Brainstorming
- Sequence Diagrams
- Data Modelling

Data capturing formats:

- Data Flow Diagrams
- Use Cases
- User Stories
- Metrics and Key Performance Indicators (KPI)
- Interface Analysis
- Time Series
- Programmatic System Models

Part of the CACTOS project was the set-up of collaboration channels among partners i.e. mailing lists, a document sharing repository, weekly teleconferences, and quarterly partner face-to-face meetings. Structured interviews were conducted over internal email where an established set of questions were aimed at the specific partner or a group of partners and direct answers to such emails were provided in-line with the question. Semi-structured interviews were conducted over the Voice over IP (VoIP) online collaboration platform and in person during the face-to-face meetings. Although there was an agenda for each encounter, free discussion around the topics on the agenda or any other business occurred. Face-to-face meetings would be scheduled for a duration of 3 to 4 days and often during this time period partners would be divided into groups to work on a specific task. The summaries of interviews were recorded as meeting minutes and information provided was used to inform the research questions and other elicitation actions.

Figure 4-2:CACTOS consortium face-to-face meeting, Umea, Sweden, March 2015

The stakeholders within the CACTOS project provided user stories which captured the current system setup and users' issues and needs going forward. Metrics and KPIs such as response time, energy consumption, and cost were included in each user story as important business drivers. As an output, the documents defining the full range of validation goals and metrics was compiled (Jelden, Domaschka, *et al.*, 2014; Hauser, Domaschka, *et al.*, 2016).

Once the stakeholders' expectations and requirements were defined, brainstorming sessions occurred. These sessions were used to share ideas among the consortium partners in an effort towards solving an issue identified by a stakeholder interview or user story. All of the ideas were recorded and rated by session participants who selected the most appropriate, feasible idea. During the brainstorming sessions, data flow diagrams were constructed and used to define the scope of the system, the data sources, and the transformation and movement within the different system components. A blackboard was used to draft system components, in the shape of boxes or circles, and arrows indicating the direction of data flow. At the end of the sessions the final diagram picture was taken and then digitised using vector graphic software.

Elicitation technique	**Purpose**
Interviews	Interviews were used to develop understanding of use cases and system constraints. The information obtained was further

	used to create system functional and non-functional requirements.
Brainstorming	Brainstorming among research stakeholders was used to tap into collective domain knowledge and discuss multiple methods and techniques available to address the tasks at hand. These sessions allowed stakeholders to analyse and select from solution approaches in a fast and concise manner.
Sequence Diagrams and Interface Analysis	Due to the large degree of integration between project components, and the use of some standalone processes, sequence diagrams were used to document the communication path for certain tasks. These diagrams served as a way of identifying the flow of data and data types within the system, thus aiding system architecture design.
Data Modelling	This approach was used to create a unanimous system model representation used as a single point of data collection and description. The model represented conceptual system behaviour and physical system features and boundaries, as well as behaviour constrains between system elements. These models were built by combining previously mentioned elicitation techniques and by collecting monitoring data directly from the system.

Table 4-1: Elicitation techniques used in the thesis

As a requirement, different software components of the project must communicate with each other by exchanging data and control instructions. The elicitation technique of interface analysis was applied to determine the data interfaces between the components and to define actual application programming interfaces (APIs). Interface analysis helped to identify the number of interfaces needed in the system, their purpose, data types and volumes, and the method of interface implementation. The technique of sequence diagram was used in conjunction with interface analysis to visualise the information passed between the objects. The diagram was used to represent the type of

the call made to the interface (synchronous or asynchronous) in chronological order. Visualising the interaction between API components makes it easy for other project participants to understand the functionality and improve API approval and adoption. A summary of each elicitation technique purpose is presented in Table 4-1.

During the development and implementation stage, a monitoring tool (CactoScale) was deployed into the testbeds. The toolkit allowed for live data collection which involved capturing cloud data centre behaviour and topology. The collected data was periodically stored to a database, which then could be used to create detailed system models. The database served as a source of information which facilitated further understanding of the cloud system component dependencies using time series data analysis. Further information on the implemented solution can be found in Chapter 5.

This thesis focuses on quantitative cost and performance analysis of cloud resource allocation policies. Challenges in extracting costs related to cloud resource allocation policies can be divided into two main categories: one, addressing the issues that industry is facing in the area of efficient cloud resource management; and two, integrating the need between data extraction and resource management tools. The elicitation techniques used in the CACTOS project allowed for extraction of all the necessary information from the consortium partners needed for the methodological steps of solution design, model development, model validation and experimentation.

4.4 Conclusions

The methodology chapter describes the steps undertaken in order to address the research aims. These steps include: Problem definition, solution design, development and implementation, experimentation, model validation, and result analysis. Each of these steps was carefully planned and executed during the study and was chosen as the most appropriate methodology for solving the issues at hand.

The CACTOS project and its partners served as a data source by providing access to case studies and tools for the work within this thesis. Project objectives were used to drive data acquisition using an array of elicitation techniques including interviews, brainstorming, sequence diagrams, data flow diagrams, use cases, user stories, KPIs and interface analysis. Use of these techniques offered mechanisms to define the research problem and gap for this thesis. Furthermore, throughout the thesis there was

constant contact with consortium partners which allowed for instant feedback and collaboration in solving some of the challenges encountered.

5 Design and Implementation

5.1 Introduction

As has been presented in Chapter 2, there has been a clear growth in the popularity of optimisation policies available for cloud data centre resource management in recent years. However, it is difficult to fully understand the impact proposed optimisation policies can have on a cloud data centre costs and QoS metrics. Every cloud data centre is unique in its hardware configurations, software setup and user resource demand requirements, hence, the same optimisation policy can deliver sub-optimal results if it is implemented across different data centres. In addition, the optimisation goals can be different for different business models of cloud data centres. For example, one might strive to save costs and energy by taking a hit on service performance speed, while another may wish to operate under strict SLAs whereby service performance speed is a crucial metric. Taking elicited requirements into account, the solution presented in this chapter aids in the understanding of optimisation policy effects on existing and newly proposed hardware configurations for an individual cloud computing data centre system. It takes the form of a discrete event simulation framework that integrates directly with the CACTOS cloud optimisation and cloud data collection frameworks (as presented in Chapter 3) enabling the simulation model build process to be automated as well as taking into account optimisation policies during the execution of a simulation run.

Following the data collection approach using methods and techniques described in Chapter 4, requirements were elicited, analysed and specified through interaction with key stakeholders. These requirements, presented in Section 4.1, were then used as the main input parameters into the overall simulation framework solution design decisions. Referring to Figure 5-1, the key simulation framework components presented in this chapter are the system models, the simulation engine and the simulation results. The system models represent the cloud data centre hardware and software configurations captured within the infrastructure models, where application models depict resource demand constraints for different operations within a running application (service). The workload model captures the user arrival rates and interaction patterns with the applications running inside a cloud data centre. Integration with the data collection framework makes it possible to automatically build a cloud data centre system model

using real, current, and historical data saving effort when modelling a large system. With this model in place, alternative scenarios can be modelled and what if analysis can be performed based on different decisions that require support.

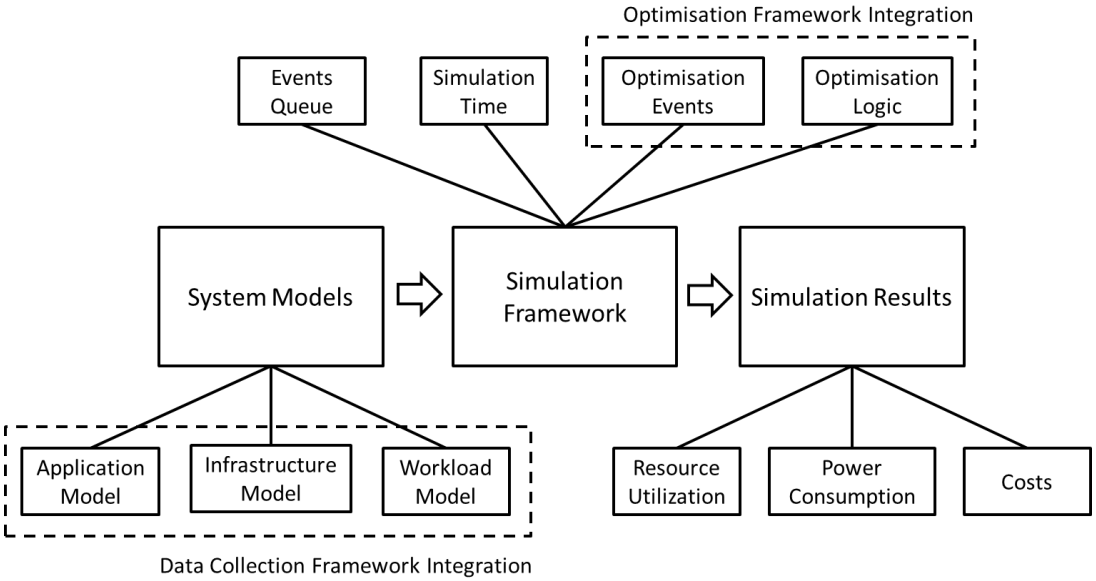


Figure 5-1: Simulation components

The discrete event simulation engine is responsible for organising and deploying different events during the simulation experiment. The main part of the simulation engine is an event queue which represents events within a simulated experiment, with different events being triggered at specific points in time. Within a simulation run, the time frame is not related to the wall clock time as it is a simulation of the time progression. Depending on the number of occurring events (the scale of the model), the difficulty of calculations and the hardware where simulation is executed, simulation time might be faster or slower than wall-clock time. During a simulation run (as the simulation is integrated with the optimisation framework) optimisation decisions based on simulated data can be included in a simulated scenario. In this case optimisation decisions are executed as events within the simulation framework as part of the running experiment. When an event is triggered, its impact on the whole system is calculated and later outputted to the simulation results. Various types of simulation events trigger different calculations in order to estimate key cloud data centre operation metrics such as resource utilisation, power consumption and costs.

Through the use of an integrated simulation framework, the impact of resource management optimisation policies can be evaluated at the design time without the need to deploy alternative configuration options into the real system. Such an approach can save time and allow for head-to-head evaluation of multiple optimisation strategies on the design of the specific system. What-if analysis can be used on an existing system or on systems that are yet to be built, allowing for the support of some of the design and configuration decisions around available hardware and software options. The cost calculation component allows for direct translation of IT metrics into business operation terms allowing for better, more intuitive, understanding of decision impact.

The following sections describe the requirements, the simulation framework design, its methods, system architecture and the implementation decisions. They also outline the information format and direction flow between components as defined within the proposed architecture design. These designs have been realised in a prototype implementation of the software which is also described at both a component level as well as at the user interface level in the form of a structured walkthrough. Finally, conclusions are presented with a lead-in to the following chapter (experimentation and analysis) which utilises this implementation.

5.2 Requirements

After the initial data collection phase which used elicitation techniques with key stakeholders as described in Chapter 4, a number of requirements were identified and specified as given in Table 5-1. These requirements were used as input parameters into the overall simulation framework solution design decisions.

Number	Type	Requirement name
F. 1.0	Functional	Ability to model data centre entity
F. 1.1	Functional	Ability to model a cluster
F. 1.2	Functional	Ability to model a rack
F. 1.3	Functional	Ability to model a node
F. 1.4	Functional	Ability to model physical resources i.e. CPU, RAM, I/O, Network, Power
F. 1.5	Functional	Ability to model a VM
F. 1.6	Functional	Ability to model a hypervisor
F. 1.7	Functional	Ability to model virtual resources i.e. CPU, RAM, I/O, Network, Power
F. 1.8	Functional	Ability to model a cloud application
F. 1.9	Functional	Ability to model a cloud user
F. 1.10	Functional	Ability to model workload induced on cloud application by a user
F. 2.0	Functional	Ability to integrate with runtime cloud resource management framework
F. 2.1	Functional	Ability to simulate self-organising resource management policies

F. 2.2	Functional	Ability to deploy additional VMs at any time during a simulation experiment
F. 2.3	Functional	Ability to migrate VMs to different nodes during a simulation experiment
F. 2.4	Functional	Ability to simulate node power management i.e. shutdown, suspend, sleep
F. 3.0	Functional	Ability to integrate with data collection framework
F. 3.1	Functional	Ability to acquire cloud virtual and physical simulation models based on historical data
F. 3.2	Functional	Ability to acquire cloud application models based on historical data
F. 4.0	Functional	Ability to produce simulation results
F. 4.1	Functional	Ability to calculate cloud resource consumption while processing user requests
F. 4.2	Functional	Ability to calculate TCO of cloud data centre
F. 5.0	Functional	Ability for user to interact with the system
F. 5.1	Functional	Ability to create a simulation experiment
F. 5.2	Functional	Ability to start or stop a simulation experiment
F. 5.3	Functional	Ability to view simulation results
NF.1	Non-Functional	Reliability: stable simulation framework performance able to produce results within the expected system constraints.
NF.2	Non-Functional	Performance: non-exponential resource demand which is increased as expected according to the simulation model size

Table 5-1: List of functional and non-functional requirements for simulation framework

The functional requirements describe technical features of the software and non-functional requirements describe secondary design requirements. Collectively these requirements serve as a feature checklist for the software product. Referring to Table 5-1, the functional requirements of group F.1 reflect the need for the simulation framework to include a model that describes the cloud computing data centre system. As mentioned in Chapter 2 the cloud data centre consists of physical hardware and virtualised resources. The physical components such as CPU, memory, disks and network interface cards (NIC) are located within nodes that are stacked in racks, divided in clusters and placed in the server room within data centre building. The simulation model must be able to capture hardware components characteristics, layout, and network topology, which is noted under F.1.1, F.1.2, F.1.3 and F.1.4 entries.

Given the large scale of cloud data centre operations, manual resource management is deemed by stakeholders not to be a feasible solution. In order to simulate cloud data centre system behaviour, resource management system decisions need to be reflected within the simulation experiment (F2.1) to more accurately determine the impact of an optimisation policy on system performance and costs. Direct integration with an operational optimisation framework would allow for a simulation to make use of already existing resource management policies. In turn, the decisions based on optimisation outputs enacted on a real system should also be realised in the simulation

environment. Such decisions include intelligent deployment of VMs (F2.2) during the simulation experiment, whereby decisions regarding VM deployment location should be made depending on the ongoing simulation state thus enabling the mimicking of real system behaviour, such as the consolidation of VMs onto fewer physical hosts potentially affecting QoS, energy and costs. In a real system, the resource management framework also monitors the system performance periodically and in the case of unsatisfactory performance, results can take corrective actions by migrating VMs and/or adjusting power states of the hardware. Requirements F2.3 and F2.4 reflect the need for the simulation framework to have the ability to model both of these aspects.

As stated previously, a cloud computing data centre is a large-scale system with complex interdependencies between components and modelling such a system by hand is a prohibitively time consuming and complex task. Requirement F3.0 reflects this through specifying the need to integrate with the data collection framework thus significantly reducing the required effort through automation. The simulation framework should acquire models of physical and virtual infrastructure based on the data from the data collection framework (F3.1). Services or applications that are running within a data centre are unique in their resource demands and their user interaction patterns, hence in order to simulate their behaviour, an application model needs to be derived based on historical data where available. The simulation is required to also be capable of retrieving application models in programmatic manner from the data collection framework (F2.2).

After the simulation of an experiment has been executed, results based on the simulated system behaviour need to be produced (F4.0). The outputs deemed important to track relate to CPU, memory, storage and network consumption over time. As such the simulation framework should have the capability to provide time series resource consumption data based as outputs (F4.1). Due to the elicited focus on holistic data centre costs as a key output metric, an additional requirement is the ability to calculate overall data centre costs relating to resource consumption as given in F4.3.

The simulation framework users (humans and software components) should be able to interact with the framework (F5.0). This should allow for simulation experiment interaction (F5.1) which entails the system model creation, modification, storage and viewing. The simulation experiment itself requires parameters such as duration, granularity, optimisation intervals and optimisation policy selections. All of these settings also should be accessible for the user. Once all of the desired experiment parameters

are set the user should be able to start the simulation, monitor the simulation progress and stop the simulation (F5.2). After the simulation has completed running, the simulation framework user should be able to view and compare simulation results in a visual manner (F5.3) through the use of charts, thus enabling decisions to be taken on the outputs of the simulation.

The delivered software should also possess certain non-functional qualities that ensures adequate reliability and performance when used. The simulation platform should have the capability to reliably execute experiments without simulation results being lost or corrupted (NF.1). The performance of the simulation framework (i.e. experiment execution speed and compute resource demand) depends largely on the size of the simulated system model (NF.2). The resource demand should increase gradually with the increase in the number of elements in the model as opposed to, for example, exponential increases in the resource demand which is a cause of errors (memory leaks). In addition, the performance should be comparable with other cloud simulation frameworks identified in Chapter 2.

Taking into account these functional and non-functional requirements, the following section details the architectural design for the software. The software architecture depicts the logical layout of the software routines through breaking them down into smaller components. Each component is responsible for performing specific tasks and together these interconnected logical components deliver the identified requirements to the end-user.

5.3 Software architecture design

The software architecture design presented in this section describes a high-level depiction of functional parts of software. Such design allows for the logical separation of concerns between different software components and gives the breakdown of tasks each component can perform. This architectural design is broken down further into class and flow diagrams containing a description of methods and datatypes used to directly feed into the development of the software.

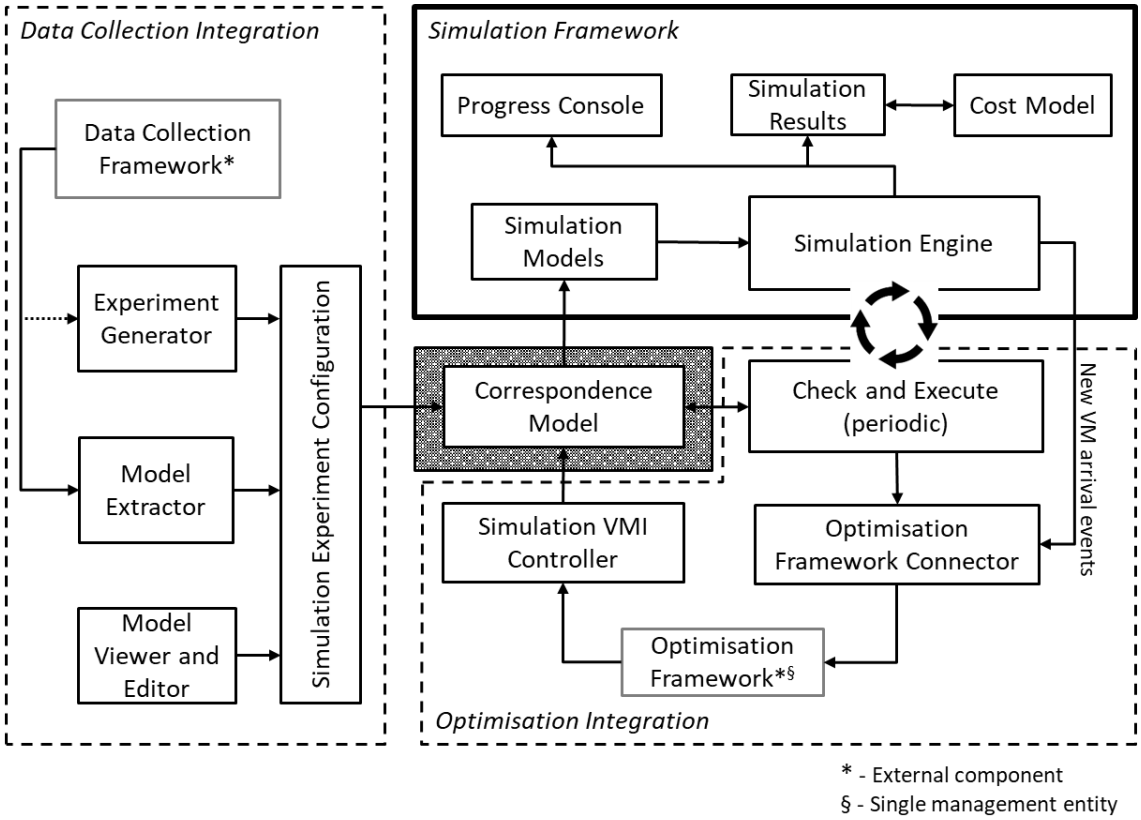


Figure 5-2: Architectural diagram of the simulation platform

The simulation framework architectural design given in Figure 5-2 is based on requirements outlined in the Table 5-1. This design complies to the general simulation components diagram flow presented earlier in Figure 5-1 and therefore the architecture components can be also divided into 3 groups i.e. “Data Collection Integration”, “Simulation Framework” and “Optimisation Integration”. Referring to Figure 5-2, the highlighted component *Correspondence Model* provides a central integration point between the cloud system models used by data collection, simulation and optimisation frameworks.

5.3.1 Data collection framework integration design

Referring to Figure 5-2, The ***Data Collection Framework*** component refers to an external software solution that continuously collects and stores monitoring data from the whole data centre. The relevant monitored data is composed of time series resource utilisation metrics of the physical hardware and virtual machines as well as topology, hardware and software specifications. It is assumed that the collected data is available

to be accessed directly via either a network API specific to the data collection framework or via a database where data being stored. Based on this data that captures data centre state over time, simulation models are automatically built.

The ***Model Extractor*** (shown in Figure 5-2) is a function with a graphical user interface component that enables users to connect the historical database provided by the data collection framework and compose models that describe cloud data centre behaviour based on the data within selected time period. These models include application models, power models and runtime infrastructure models. An application model describes cloud deployed application behaviour through resource demand patterns. A runtime model includes current hardware and virtual layer configuration, including a CPU, memory, storage and network description for physical and virtual hosts. Large scale data centres consist of thousands of nodes therefore automatic model creation can significantly reduce modelling effort through the use of the model extractor.

The ***Model Viewer and Editor*** component provides a GUI for viewing and editing the simulation models. Representing a model graphically aids in the understanding of system design and provides a mechanism for visually checking for inconsistencies. While model creation from scratch is not recommended due to the typically prohibitively large size of a model in this domain, the editing function can aid in testing what-if scenarios that require small scale changes to be made in the models. Apart from the system model, the simulation platform itself requires auxiliary parameters such as the simulation experiment duration time and random seeds to be defined.

The ***Simulation Experiment Configuration*** is a GUI based component that provides a joint configuration point for the simulation engine. It specifies the location of models, simulation runtime configuration parameters and the location for the serialised simulation output results.

The ***Experiment Generator*** component is used to generate experiment scenarios that are either fully synthetic or which are partially based on the real data from a data collection framework. The use of fully synthetic models can aid in the theoretical exploration of resource provision strategies under alternative anticipated scenarios. This approach is beneficial at an early prototyping stage where real data might not be available. Mixed types of experiment scenarios can also be created where (for example) the hardware configuration models are taken from a real system, but user interactions

are synthetically generated to fit a specific pattern of higher demand. Such functionality allows for the simulation of resource allocation policy behaviour in situations that can occur but have not been captured yet by the data collection framework. Finally, the experiment generator can be used to extrapolate models and simulate large scale behaviour based on the smaller sized models. This functionality is particularly of use in cases whereby a resource allocation policy is tested on a smaller set of nodes (i.e. in a testbed), but it is not clear how it will perform in an actual size data centre. In this case a simulation experiment can be created using extrapolated system behaviour models based on the results gained from the initial testbed deployments, in order to test such a policy at a larger scale.

5.3.2 Simulation framework design

The design of simulation framework in Figure 5-2 is presented as five abstract components: ***Simulation Models, Simulation Engine, Progress Console, Simulation Results Output*** and ***Cost Model.***

The ***Simulation Models*** component represents models that capture cloud data centre resource configuration and resource demand relationships in a programmatic way. This component is comprised of an infrastructure model, workload model and application model. The infrastructure model describes the configuration and topology of resources like CPU, memory, storage and network within the cloud data centre. The workload model defines the amount and type of requests that are being submitted by users to the applications (services) which are running inside the cloud. The application model is a link between compute resources and user requests. It translates user requests into resource demands based on the type of request that is being made. The designed application model captures resource demand behaviour of VM as a whole, hence the simulation model assumes one-to-one mapping between the application model and VM. Such a modelling approach allows for aggregated behaviour representation of the resource utilisation based on type, amount and frequency rate of user requests.

The ***Simulation Engine*** component is a discrete event simulation (DES) engine which manages the creation and timely execution of events that represent behaviour of the cloud data centre. The DES engine is comprised of standard components: a state engine, a system clock and an event queue. The state engine manages the state of variables within the simulation model during the simulation execution. The system clock

keeps track of simulation time needed for simulation event coordination. The event queue holds a list of all of the main events that will be executed during the simulation run. The events are arranged by their execution time and are required to proceed in the exact specified order. The simulation engine embodies the “brain” of the simulation framework whereby all of the event-based system behaviour calculations are made.

The ***Progress Console*** diagram component represents the command line prompt type of user interface whereby the information about simulation progress is written. This information includes the stage of the simulation process, simulation time, and any errors or warnings that might occur during simulation execution run. The information in the console serves as a feedback mechanism to the user which aids in both understanding the progress of the simulation run and in debugging any errors that might occur.

The ***Simulation Results*** component represents the final output of the cloud data centre behaviour calculations made during a simulation experiment execution. The calculations are stored in the format of time series data for each metric. It also provides a User Interface (UI) for human readable viewing and analysis.

The ***Cost Model*** component provides a cost calculation method and a model implementation that 1) stores data centre cost related data, 2) is able to read time series simulation results based on which costs are calculated, and 3) generates a cost report based on the combined cost data (see Chapter 5.3.5 for more details).

The format of data collected from the monitoring framework differs from the models used by the simulation engine. The ***Correspondence Model*** contains reference bindings between the simulation engine models and the monitored data models (Stier and Groenda, 2016). With the help of the correspondence model a translation of monitored data can be made to create simulation input, and in the same way simulated data can be translated back into the monitored data format. Such an operation is necessary in order to integrate the simulation framework with both the data collection framework and the optimisation framework.

5.3.3 Optimisation framework integration design

In a real cloud system an optimisation framework is responsible for placement decisions of newly arriving VMs, and continuous VM consolidation through migration

from one host to another. When a user submits a request to admit a VM to a cloud management platform (e.g. OpenStack), the optimisation framework is called for the decision on which hypervisor in a data centre the VM should be placed. In addition, the optimisation framework polls the cloud system monitoring data periodically to check whether any VM migration actions are needed according to the workload. In order to reflect the effects of such a runtime resource management platform within a simulation experiment, the simulation framework is required to be integrated with the cloud optimisation platform.

Referring to Figure 5-2, since the simulation framework does not operate in real time, a ***Check and Execute*** routine is needed to call the optimisation framework via the Optimisation Framework Connector. This routine is executed periodically using the simulated time frame, pausing the simulation for the duration of the optimisation framework execution. Outside of the periodic checks, ***New VM Arrival Events*** are executed at a particular time specified in an input model. The rest of the VM admission procedure remains similar - the simulation process is paused, and the VM provisioning configuration is sent together with the simulated infrastructure load models. Next, the optimisation framework based on the given data generates an optimisation plan that contains VM placement instructions.

The ***Optimisation Framework*** component represents an external resource management framework (described in Chapter 3.5). The Optimisation Framework component represents an external resource management framework (as described in Chapter 3.5). The framework is a single management entity that is deployed in a data centre as an additional service which makes decisions on VM placement, scaling and migration. In a runtime mode it makes decisions based on the incoming data from the data collection and monitoring framework. However, while in the offline mode, the monitored data is substituted by the simulation outputs. The external framework is required to have an API that can be called via the ***Optimisation Framework Connector*** specifically designed for such integration.

The ***Simulation VMI Controller*** component is another optimisation framework integration auxiliary component that is able to parse optimisation instructions and transform the correspondence model based on these instructions. For example, if instructions received from the resource management framework stated that a VM needs

to be migrated from one hypervisor to another, the VMI controller should apply these suggestions to an existing model.

Note that in the live runtime environment the resource management framework works in an asynchronous way with the cloud management system, since both systems run in real time. However, when coupled with the simulation platform the simulation has its own time counting mechanism whereas the optimisation framework runs on wall clock time, therefore the simulation needs to be paused until the response from the optimisation framework is received.

5.3.4 Runtime model design

Creating a common cloud data centre system description format is vital for integration between data collection, optimisation and simulation frameworks. Having standard system models both reduces the effort of integration and increases the reliability of each individual component. By having an agreed central model data format, the system monitoring framework “knows” what data to collect, the resource management framework “knows” what data is available to use by the optimisation algorithms and the simulation framework “knows” what data is available for building simulation models.

Entity	Attributes
Data Centre	Name, ID, List of Clusters
Cluster	Name, ID, List of Racks
Rack	Name, ID, List of Nodes, List of PSUs
Node	Name, ID, List of CPUs, List of Memory, List of Storage, List of Network Switches
CPU	Name, ID, Frequency, List of Cores
CPU Core	Name, ID
Memory	Name, ID, Capacity

Storage	Name, ID, Capacity, Read/Write Speed
Network Switch	Name, ID, Bandwidth, Connected Entities
Power Supply Unit (PSU)	Name, ID, Maximum Wattage, Connected Entities

Table 5-2: List of components of physical data centre model components

Referring to the list of requirements in Table 5-1, a hardware infrastructure model should include the following components: Data Centre, Cluster, Rack, Node, CPU, Memory, Storage, Network and Power. Each of these components has a list of attributes describing the configuration and topology compiled in Table 5-2. Each of the components has to have a unique identification assigned for programmatic processing and a Name attribute for human readability. The hardware model is hierarchical in nature – a data centre includes a list of clusters, a cluster can include a number of racks and each rack is comprised of a number of nodes and power supply units. For each node a hardware configuration is captured by describing available resources i.e. CPU, memory, storage, network and power. A CPU is described as a unit that operates at a certain frequency and has a certain number of available cores. A memory module has a certain capacity that is available for provisioning. A storage unit is described as an entity which has an available capacity and reading and writing speeds. A network switch is described as an entity in which the network connections are made and has network bandwidth and list of connected entities as attributes.

Entity	**Attributes**
Hypervisor	Name, ID, Node ID, Type, List of VMs, List of Network Switches
Virtual Machine (VM)	Name, ID, List of CPUs, List of Memory, List of Storage
vCPU	Name, ID, Number of Cores
vMemory	Name, ID, Capacity
vStorage	Name, ID, Capacity

vSwitch	Name, ID, Bandwidth, Connected Entities

Table 5-3: List of components of virtual data centre model components

The hardware layer is then complimented with virtual layer entities such as Hypervisor, Virtual Machine, vCPU, vMemory, vStorage and vSwitch. The purpose of the model is to capture the configuration for each virtual machine in terms of required resources, model entities and attributes are shown in Table 5-3. The hypervisor entity provides a link between physical and virtual resources with direct link to a specific hardware node in the data centre. Since a VM in reality creates virtual environment which emulates real hardware, the resource configuration remains similar to the one shown in the hardware model.

Entity	Attributes
CPU Core	Name, ID, Utilisation level
Memory	Name, ID, Utilisation level
Storage	Name, ID, Utilisation level
Network Switch	Name, ID, Utilisation level
Power Supply Unit (PSU)	Name, ID, Utilisation level

Table 5-4: Resource utilisation model components

The infrastructure model can be separated into two parts – the hardware layer and the virtual layer. The hardware model describes a configuration of a physical infrastructure while a virtualised layer captures resources assigned to VMs. The resource demand data within a cloud data centre is dynamic and changes with the fluctuation of user demand, therefore the utilisation data should be captured using a separate set of models that contain only utilisation rates for hardware and virtual resource models. The utilisation model design, shown in Table 5-4, focuses on including only resources and their utilisation data. Each resource is then linked by a unique ID to the hardware and virtual model components. Separation of utilisation data from rarely changing configuration data makes models lighter and therefore faster to generate, store and pass around the services.

A workload model describes the amount and frequency of user requests sent to a cloud application and can be seen as part of the application model itself. The assumption is that each application has a usage pattern that reflects user demand for a particular simulation experiment. Such workload information can be captured by the data collection framework from the live system or generated by the Experiment Generator component for describing against corner cases that occur rarely but need to be anticipated.

An application model defines the resource demand of an application deployed in a cloud data centre. Depending on the nature of the application, its behaviour will exhibit different resource demand patterns. Application types can range from a simple single VM deployment to a more complex distributed multi-tier software design spanning across multiple VMs. Different user interactions can trigger different application components resulting in a different resource demands depending on the user request path through the application components. An application model in simulation creates a link between user requests and cloud resource demand requirements.

Entity	Attributes
Application Component	Name, ID, Execution probability
CPU demand	Million Instructions (MI)
Storage demand	Amount of data read/written
Inputs	List of application components
Outputs	List of application components

Table 5-5: Application model components

Each application can be broken down into different interconnected components, the design shown in Table 5-5 describes these. Each application component has an execution probability attribute which determines how likely it is that this component will take part in processing a user request. To process a user request within a component, a certain amount of resources is required shown in the attributes of CPU and storage demand. A CPU demand is expressed in Million Instructions (MI) and storage demand is expressed in the amount of data that require to be written or read. The last two entities

(input and output) contain a list of other application components that are connected to this component forming a path that a user request can traverse.

5.3.5 Cost model design

By simulating the impact of resource provisioning policies on cloud system behaviour, better power consumption demands and equipment utilisation levels taking into account performance variability can be estimated. Power consumption is a direct operational metric that data centre operators typically try to reduce in order to save on costs and become more eco-efficient. From a cost perspective, underutilised compute resources are also considered to be wasteful through the loss of value from depreciation over time.

The cost models described here are aimed at providing data centre Total Cost of Ownership (TCO) calculations based on a combination of data obtained from a cloud data centre hardware infrastructure configuration, overtime resource utilisation data and financial data on the cost of the equipment, energy, property, labour and other expenses. The hardware infrastructure and time series utilisation models (shown in table Table 5-2, and Table 5-4) provide the foundation for cost calculations. Additional data relating to cost may typically be obtained from the accounts payable department, financial reports extracted from an Enterprise Resource Planning (ERP) system or inferred from market research in cases where no real cost data is available.

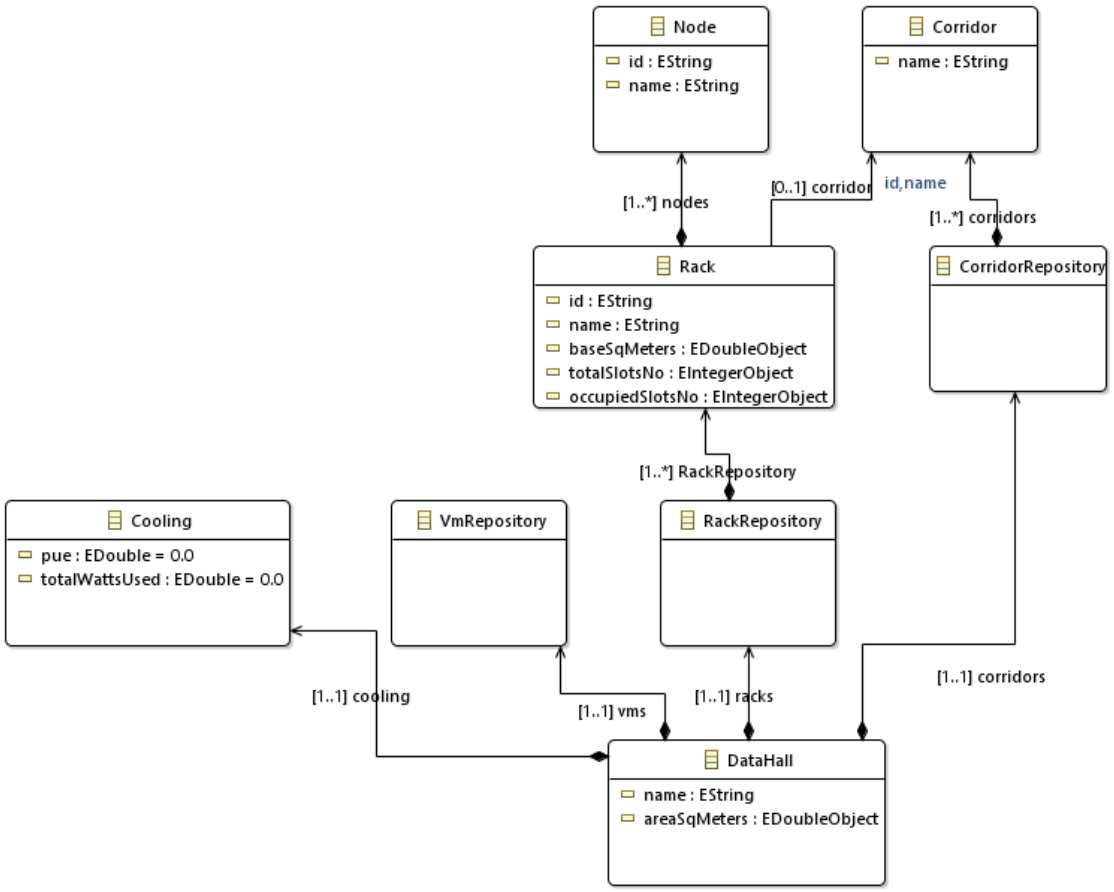


Figure 5-3: Cloud data centre cost model design

The cloud data centre cost model given in Figure 5-3 is designed by mapping overlapping features of TCO models (presented in the literature review in Chapter 2) on top of hardware infrastructure entities (presented in Table 5-2). Referring to Figure 5-3, the cost model separates the data centre as a whole physical location and the data hall where the compute nodes and subsequently virtualised resources are located. The data centre represents a whole physical space that is taken not just by a server room, but also other auxiliary utility space needed to host data processing and hosting equipment. For example, personnel who works on site require additional facility space such as office room, a canteen, meeting rooms and parking lots. Also, additional equipment such as cooling solutions, emergency electricity generators and firefighting gear all requires additional space. A data hall is an area where compute nodes and network switches are placed inside racks. Apart from space, equipment in a data hall also requires technical staff, power and cooling to operate. The cost model enables the calculation of TCO

based on the direct expenses of operating a data hall and the secondary expenses associated with running a whole data centre premises.

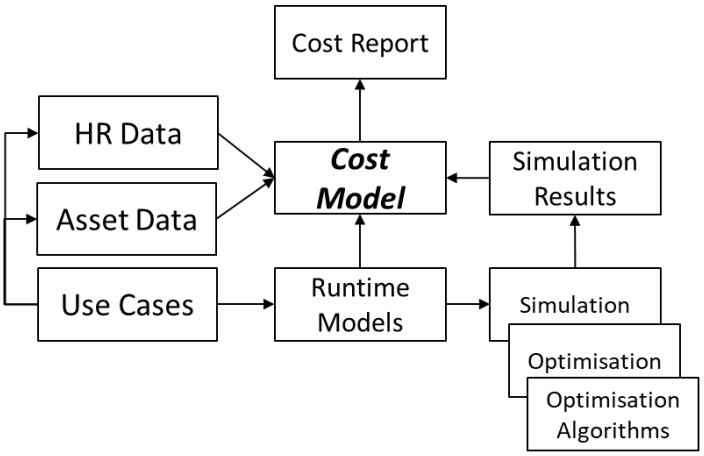


Figure 5-4: Cost model data flow diagram

The data flow diagram shown in Figure 5-4 illustrates the inputs and output components together with the direction of data exchange. Based on the use case (experiment) to be simulated, data from human resources (HR) personnel cost records and asset data such as asset age and cost is being collected into the cost model. Within the cost model HR and asset data is mapped onto runtime models that describe an internal configuration of hardware. In the next step the simulation is executed based on the cloud data centre runtime models and selected resource optimisation strategy. Simulation results also are then parsed inside the cost model alongside the runtime model assets. Once all of the data has been obtained the cost model can be used to produce a cost report containing: TCO, Servers cost, Server Power cost (Power consumption), Cooling power cost, Personnel cost and Maintenance cost.

5.3.6 TCO calculation

To calculate TCO using information provided within a cost model, specific mathematical formulas are required to transition from individual expense items and compute resource utilisation data to a TCO figure. The TCO calculation formulas presented in this section are based on a modified version of the work of Simonet, Lebre, *et al.*, (2016) which calculates cloud data centre TCO in a clear and logical manner.

$$Cost_{total} = Cost_{facilities} + Cost_{servers} + Cost_{maintenance}$$

$$+Cost_{network} + Cost_{energy} \quad (1)$$

As shown in Equation 1 the authors define TCO as a sum of cost of facilities, servers, maintenance, network and energy. However, in the course of this research some of the underlying formulas were modified to incorporate simulation results and different methods of labour and facilities cost estimations. New calculation methods proposed in this thesis enable the use of simulation derived energy consumption data to be included, in turn providing more precise energy consumption estimation within TCO. Formulas for calculating the cost of labour and facilities were altered include cost information in relation of size of the physical site of a data centre. Standard amortization constants, shown in the Table 5-6, were used to account for resource depreciation over time, however these can be adjusted depending on the best practices of an individual enterprise.

Constant	Type	Value
A_s	Servers, routers and switches	5 years
A_m	Racks and cables	10 years
A_l	Buildings	20 years

Table 5-6: Amortization constants (Simonet, Lebre, *et al.*, 2016)

It is worth noting that the Time Value of Money (TVM) concept can be considered when performing amortization calculations. The TVM calculates the interest that can be earned over the period of time. For example, if money is lodged into a savings bank account that earns interest, the sooner the lodgement occurs the more interest value can be earned. The fundamental TVM formula is:

$$\mathrm{FV} = \mathrm{PV} \times (1 + \frac{i}{n})^{(n \times t)}$$

where *FV* - is future value of money, *PV* - present value of money, *i* - interest rate, *n* - number of compounding periods per year and *t* – number of years (Investopedia, 2019). While the TVM can be used to increase the precision of cost calculation, it is not used within the presented work to be consistent with the existing TCO models.

Facilities

Facilities are all of the auxiliary premises and equipment outside of a data hall needed to keep servers running. Examples of premises can include staff offices, canteens, parking or any other functional space. Auxiliary equipment can include voltage

transformers, diesel electricity generators, cooling equipment and fire supressing system. The proposed TCO model estimates the overhead cost of facilities per number of racks hosted inside a data hall. In such a way an estimate of cost can be obtained for a part of the data centre without the need to model and simulate the full infrastructure stack inside the data hall.

$$Cost_{facilities_{sqm}} = \frac{Cost_{building_{facilities}}}{Site_{area}} \quad (2)$$

$$Datahall_{area} = Rack_{area_{sqm}} \times N_{rack} \times K_{space} \quad (3)$$

The connection between the cost of facilities and the number of racks is made through the area of space the rack occupies within the data centre. First the cost of whole building facilities is divided by the total site area where the data centre is built.

$$Cost_{facilities} = Cost_{facilities_{sqm}} \times Datahall_{area} \times A_t \quad (4)$$

Second, the rack base area is multiplied by number of racks and then by the square meter facilities cost. However, in the data hall racks are spaced out to form corridors for access and air flow, therefore a space coefficient (K_{space}) is added to the multiplication to account for additional space requirements beyond the base of rack area. Finally, the amortisation multiplier is added to establish the yearly cost according to predicted depreciation.

Support and Maintenance

The function of support and maintenance is performed by employees that are involved in running the data centre. On-site job positions typically include the facilities technicians, hardware operations engineers and mechanical engineers. These roles would also range in seniority from an assistant to a managerial level. The number of employees would vary from site to site depending primarily on the size of the site.

$$N_{employees_{sqm}} = \frac{N_{employees}}{Site_{area}} \quad (5)$$

To get an estimation of the labour cost for a fracture of a data hall area, firstly the number of employees per square meter is determined by dividing the total number of data centre employees and the area of the data centre.

$$Cost_{maintenance} = Datahall_{area} \times N_{employees_{sqm}} \times Salary_{avg} \quad (6)$$

Secondly, the annual cost of support and maintenance ($Cost_{maintenance}$) is estimated by calculating the data hall area ($Datahall_{area}$) multiplied by the number of employees per square meter ($N_{employees\_sqm}$) and multiplied by the average salary of all of the employees ($Salary_{avg}$).

Servers

Racks contain server nodes and switches that compose the core function of a cloud data centre. The controller nodes are responsible for hosting VM management software, compute nodes are responsible for hosting VMs and providing computing power in form of CPU and memory, whereas storage nodes host a multitude of solid state or hard drives to provide reliable storage that can be attached (mounted) to the compute nodes.

$$Cost_{servers} = Cost_{contr} + Cost_{comp} + Cost_{storage} \quad (7)$$

The total cost of servers ($Cost_{servers}$) hosted in the data hall can be expressed by a total cost sum of controller($Cost_{contr}$), compute ($Cost_{comp}$) and storage ($Cost_{storage}$) nodes.

$$Cost_{contr} = N_{contr} \times P_{contr} \times A_s \quad (8)$$

$$Cost_{comp} = N_{comp} \times P_{comp} \times A_s \quad (9)$$

$$Cost_{storage} = N_{storage} \times P_{storage} \times A_s \quad (10)$$

The yearly total cost for each node type can be calculated by multiplying the number of nodes by price per node and amortisation value.

Network

The switches are responsible for network traffic routing between server nodes and are calculated separately as part of network equipment.

$$Cost_{intranet} = N_{switch} \times P_{switch} \times A_s \quad (11)$$

The total costs per year of network equipment ($Cost_{intranet}$) inside data centre racks are calculated by multiplying the number of switches (N_{switch}) by the price of each switch (P_{switch}) and equipment amortisation (A_s).

$$Cost_{cables} = N_{switch} \times P_{cables} \times A_m \quad (12)$$

The total yearly network cable costs are calculated by multiplying the number of switches (N_{switch}), price of cables (P_{cables}) per switch and the appropriate amortisation value (A_m) which is different to the equipment amortisation A_s.

Power

The cost of power consumed by the compute nodes in a cloud data centre is calculated by the use of the simulation framework. The employed linear power model (Fan, Weber and Barroso, 2007) relies on the measured minimum and maximum node power consumption values and the CPU utilisation. Through the use of a linear formula the power consumption *P(u)* can be obtained by adding a gradient difference in power consumption calculated using the CPU utilisation factor *u*.

$$P(u) = P_{idle} + (P_{busy} - P_{idle}) \times u \tag{13}$$

The P_{idle} value corresponds to a measured power draw of an idle system and P_{busy} value corresponds to the measured maximum power consumption at 100% of CPU utilisation.

$$P_{com_{node}} = \sum_{t=t_{int}}^{sim_{end}} P(u_t) * t_{int} \tag{14}$$

Each P(u) calculation is made at a certain time interval depending on the simulation granularity setting. The total power draw for the whole compute node within the duration of a simulation experiment would be the sum of all of the measured intervals with the assumption that the power draw remains constant for the duration of the whole interval until the next measurement takes place. The calculated value of P_{comp_node} measures power consumption in Watts per second or Joules.

$$E_{comp_{node}} = \frac{P_{comp_{node}}}{1000 \times 60^2} \tag{15}$$

To convert Joules into more a domestically common kWh value, P_{comp_node} is divided by the number of seconds in an hour, 3600, and 1000. The value E_{comp_node} represents the cumulative energy consumption of a single compute node for the duration of a simulation experiment measured in kWh.

$$E_{comp} = \sum_{x=1}^{Ncomp} E(x)_{comp_{node}} \tag{16}$$

Total energy consumption is expressed by the E_{comp} value which is the sum of cumulative energy consumption E_{comp_node} for every compute node in the cloud data centre.

$$Cost_{comp} = E_{comp} \times P_{kWh} \quad (17)$$

To translate the energy into cost, the total energy E_{comp} is multiplied by the kWh cost obtained from the local energy provider. The $Cost_{comp}$ value is the total monetary value of the energy consumption for the all compute nodes in the cloud data centre within the duration of simulation experiment.

Aside from the compute nodes the power in cloud data centre racks is also consumed by the switches, controllers and storage nodes. Simulation of the behaviour and power consumption of mentioned auxiliary entities is outside the scope of the research in this thesis.

$$E_{static} = E_{switch} \times N_{switch} + E_{contr} \times N_{contr} + E_{storage} \times N_{storage} \quad (18)$$

However, it would be impractical to completely dismiss other node energy demands, therefore these calculations are estimated b static power consumption data defined in the equipment specification. The formula shows E_{static} being calculated multiplying number of nodes by type and summation of all of the results.

$$Cost_{static} = \frac{E_{static}}{1{,}000} \times Hours_{sim} \times P_{kWh} \quad (19)$$

The conversion of all static energy estimation into costs is expressed by $Cost_{static}$ value where total estimated static power consumption is converted to kWh energy value and multiplied by PkWh price per kilowatt hour.

The cost of cooling is modelled by using a PUE measure and equals the multiplication of total energy consumption by the PUE value.

$$Cost_{cooling} = Cost_{energy} \times PUE \quad (20)$$

The PUE value shows the relative energy needed to cool nodes inside a rack inside a data hall area. The detailed cooling energy estimation is outside the scope of this work and PUE value is used as an indicator.

$$Cost_{energy} = Cost_{compsim} + Cost_{static} + Cost_{cooling} \quad (21)$$

Finally, the total compute node simulated costs are added to the static estimated costs to form the total figure of the energy draw from the operational racks in a cloud data centre. The $Cost_{energy}$ value represents a monetary cost of energy consumed within the data hall area of cloud data centre.

The described new TCO calculation methods provide new features through the use of power measurement time series data for calculating data hall energy costs and estimates of maintenance and facilities costs based on the data hall general area size. New features enable the use of simulated time series data directly with the TCO calculation making it easy to use simulation generated data together with costs. Data hall size bound maintenance and facilities cost estimations allow for the generation of approximations based on cost per square metre. These approximations can be used when granular cost data is not available for a particular data centre but is available for an alternative one that can be mapped via a size conversion.

5.4 Implementation

The implementation of the simulation framework inclusive of integration with the data collection framework and optimisation framework was accomplished via the CACTOS FP7 project, presented in Chapter 3. The three tools developed during the project (CactoScale, CactoOpt and CactoSim) serve as data collection, resource provisioning and simulation frameworks respectively. This section re-iterates some of the implementation decisions made during project across all the toolkits, with a specific focus on the simulation framework (CactoSim) implementation.

5.4.1 Simulation framework implementation

The CactoSim simulation platform developed and used for this thesis is based on the Palladio simulator (Rathfelder and Klatt, 2011). The Palladio simulator is purposely designed to support improvement in software quality at design stage, which it accomplishes through simulating the behaviour of software component-based architectures. The simulator is written in the Java language as a series of plugins for the Eclipse Integrated Development Environment (IDE) (The Eclipse Foundation, 2018b). Palladio features are based on the Eclipse Modelling Tools (The Eclipse Foundation, 2018c) collection as a UI platform for aiding in model development, visual model design, simulation experiment configuration and simulation result visualisation. An actual Eclipse

based Palladio UI example screenshot is given in Figure 5-5. The left-hand side panel represents a list of projects containing a list of files that contain models and their visual representation. The middle panel shows an interactive model visualisation, and the right-hand side panel shows available elements that can be used with the particular model type.

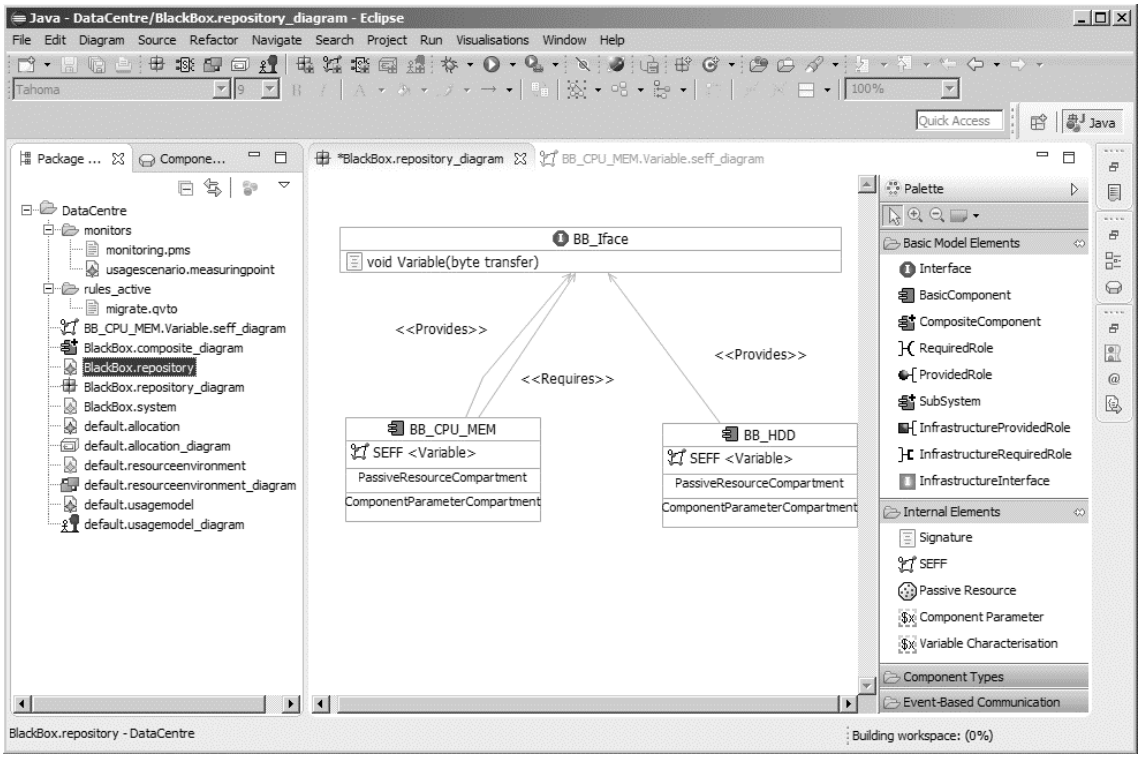


Figure 5-5: Palladio UI example running on Eclipse IDE

As mentioned in Chapter 3, at the core of the Palladio simulation framework is Palladio Component Model (PCM) which consists of specific sub-models: component model, composition model, resource environment model, deployment model, and usage model. The PCM is implemented using Eclipse Modelling Framework (EMF) which allows for the separation of the meta-model design and the model code implementation through automatic code generation. For event orchestration uses the SimuCom simulation engine which is based on Desmo-J, a general purpose DES engine (Lechler and Page, 1999).

Similar to Palladio the CactoSim simulation framework is written also as a collection of plugins for an Eclipse IDE using Eclipse libraries for creating UI components for experiment handling operations such as project creation, model creation, serialisation

and maintenance (see Chapter 5.5). CactoSim extends existing Palladio functionalities by introducing cloud data centre specific runtime models, integration with a data collection framework and integration with the resource management framework.

5.4.2 Runtime model implementation

While PCM has a notion of nested hardware components within the Resource Environment model, it lacks model entities and attributes that specifically describe compute hardware configuration (e.g. CPU frequency, memory capacity, I/O speed) and virtual layer of cloud computing (e.g. Hypervisor, VM). To address that, a lists of components describing the cloud data centre system presented in Table 5-2, Table 5-3 and Table 5-4 were implemented as the following models: Physical Data Centre Model (PDCM), Logical Data Centre Model (LDCM), Physical Load Model (PLM) and Logical Load Model (LLM). As the names suggests, PDCM describes hardware configuration and topology, LDCM describes VM configuration and placements, PLM captures utilisation of physical hardware and LLM captures utilisation of virtual hardware monitored from inside of each VM (See Chapter 3 for more detail). These models were created by using a Model-Driven Software Development (MDSD) process which consists of a meta-model definition stage which formalises model entities, their attributes and dependencies, and an actual model source code generation based on mentioned meta-model. The MDSD model is essentially a generated software code that can be used to programmatically create model instances as programming class objects, their methods and their attributes. The MDSD technique is used to create a central model definition instance that enables the avoidance of any discrepancy between an abstract model design and an actual implementation. The Java language-based Eclipse Modelling Framework (EMF) was chosen as an implementation medium that supports the MDSD approach, and also due to compatibility with PCM. EMF consists of three main components: a meta model (*Ecore*), *EMF.Edit* and EMF.Codegen. Ecore provides a core framework functionality for defining a meta model description, model persistence through XMI serialisation and reflective API for managing EMF objects. The *EMF.Edit* is a framework that enables integration through reusable classes with the editors of EMF models. Finally, the *EMF.Codelgen* represents a code generation facility that creates the *Ecore* metamodel automatic translation into the source code (Eclipse Foundation, 2018).

Due to its intentional component design the CACTOS runtime model serves as a standard data exchange format across all of the tools. CactoScale gathers data from the live system in accordance with the requirements of a runtime model, CactoOpt implements resource optimisation policies using the data in the runtime model and CactoSim bases the simulation experiments on the same data. Since the data format is shared amongst all of the CACTOS tools, the implementation is made in a form that can be used by all the tools, supporting the functional requirements of the project as a whole.

5.4.3 Data collection framework integration implementation

CactoScale was used as a data collection framework to create runtime models and effectively simulation models based on both immediate and historical data. The implementation of CactoScale was based on polling data from Chuckwa agents into a distributed Hadoop Distributed File System (HDFS) placed on dedicated data storage and processing nodes.

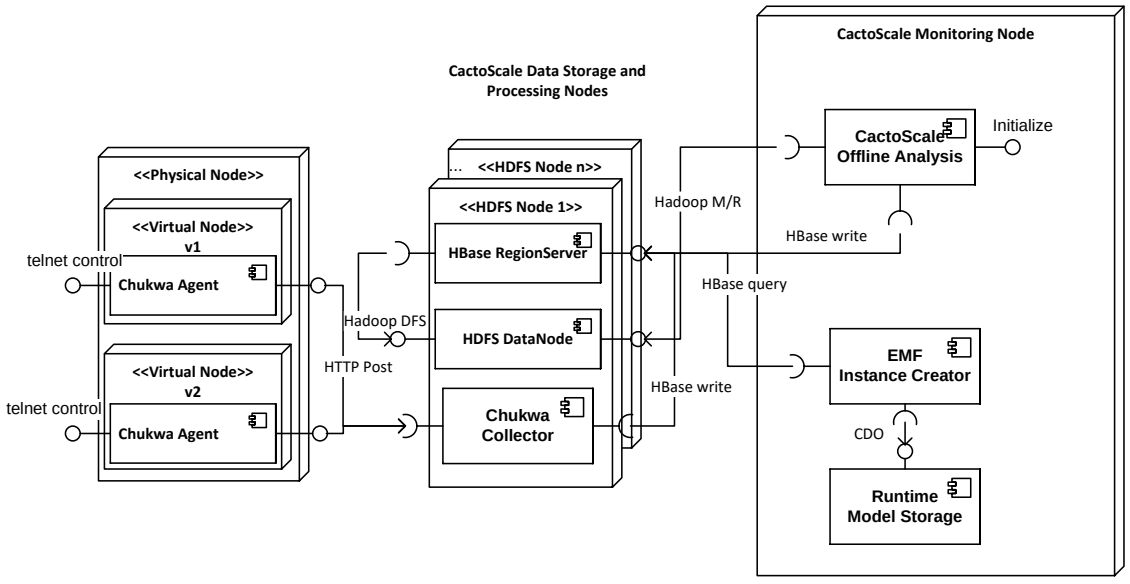


Figure 5-6:CactoScale architecture

As shown in Figure 5-6, the data flows from agents collecting hardware, VM configuration and topology information to HDFS nodes, where further it is queried by the *EMF Instance Creator* component and pushed into the *CDO Runtime Model Storage*. The Eclipse CDO (Connected Data Objects) Model Repository allows for “on-the-fly” EMF model creation and persistence by providing 3-tier architecture that includes support for EMF client, model storage and pluggable backend database solutions (The

Eclipse Foundation, 2018a). The CloudScale HDFS data collection plays a role of the backend data base which is parsed through CDO and exposed in the format of CACTOS runtime models (discussed in Section 3.6 and Section 5.4.2).

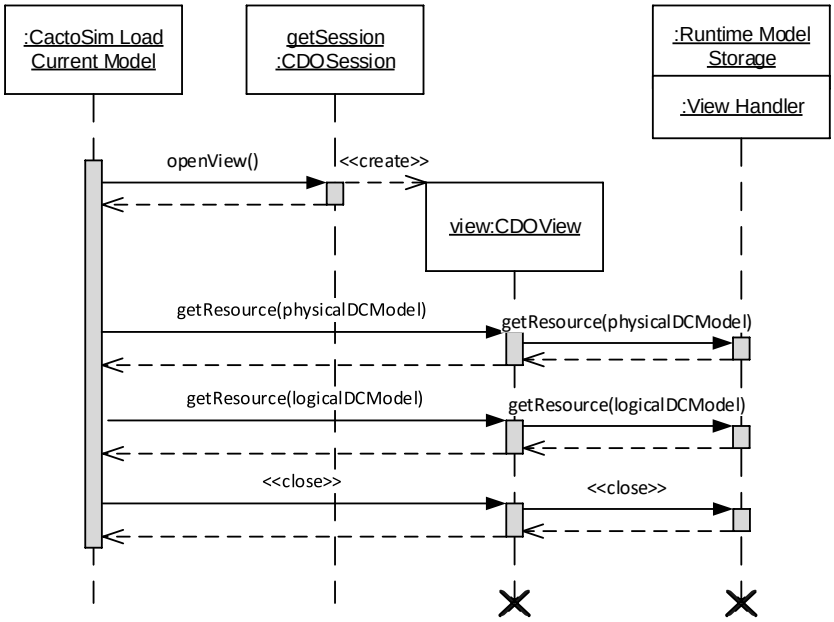


Figure 5-7: Sequence diagram of interaction between CactoSim and CDO runtime model storage

In order to obtain these CDO based EMF runtime models, CloudSim implements a CDO client whose sequence diagram is given in Figure 5-7. The CactoSim CDO client is first required to establish a connection through providing a CDO server address and credentials. Once the connection is established the CDO View object is returned, using this object the client can get both Physical and Logical data centre models. Once the model retrieval operation is finished the connection is closed and the models are serialised inside the Eclipse project workspace. A full UI walkthrough on how to retrieve runtime models from a CDO repository using CactoSim can be found in the User interface Section 5.5.

5.4.4 Optimisation framework integration implementation

The CactoSim simulation framework is integrated with the optimisation framework, CactoOpt, so as to simulate the self-managing properties of an autonomous cloud system. CactoOpt is a resource management framework that uses optimisation techniques in order to achieve certain objectives such as cost, energy efficiency and

performance given certain QoS constrains. In addition, the CactoOpt framework can be seen as a wrapper around optimisation functions that are used for the VM placement and continuous VM location optimisation at real time inside real system deployment. For example, when a user creates a VM, CactoOpt decides to which node in a cloud data centre this VM will be assigned. The continuous optimisation polls the cloud system state from the data collection framework and decides if any VMs need to be migrated to other nodes in order to assure optimal system performance, within the set optimisation goals.

```
01. public interface IOptimisationAlgorithm {
02.
03.     public OptimisationPlan generateOptimizationPlan(
04.                                         PhysicalDCModel pdcm,
05.                                         LogicalDCModel ldcm,
06.                                         PhysicalLoadModel plm,
07.                                         LogicalLoadModel llm);
08. }
```

Figure 5-8: Continuous optimisation interface API

The simulation framework can mimic the continuous optimisation method by calling the optimisation framework and specifying an optimisation policy and the set time interval within the simulation experiment configuration. In Figure 5-8 a Java code interface snippet is shown which is extended by every continuous optimisation algorithm to make a call standard across every algorithm implementation. The interface expects to receive runtime system models consisting of a physical data centre model, a logical model and the load of physical and logical hardware at the time of the call. These models are populated with the simulated data using the same format as the real system.

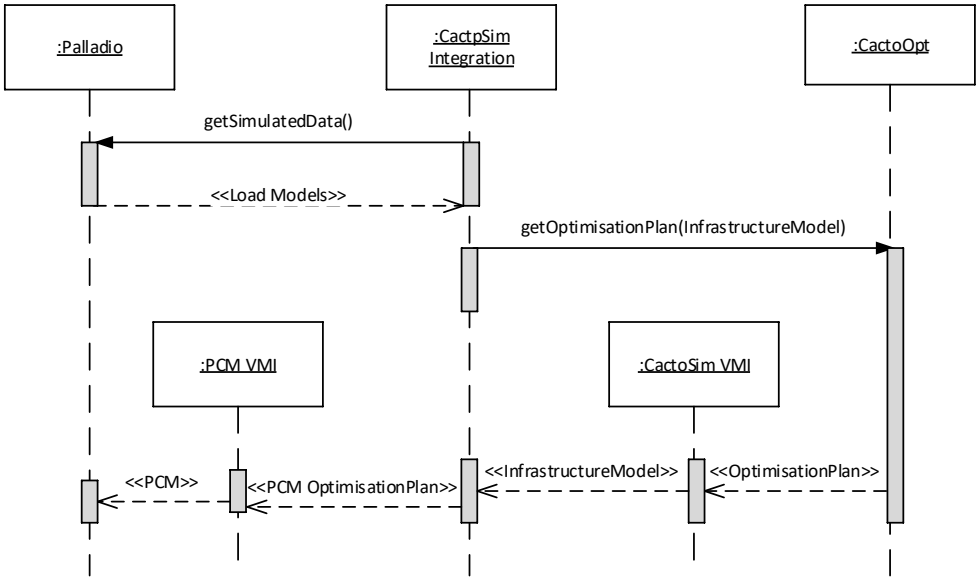


Figure 5-9: Sequence diagram of interaction between CactoSim and resource management framework CactoOpt

The sequence diagram shown in Figure 5-9 shows the calls stack occurring inside the integration logic between CactoSim and CactoOpt. Once the periodic stop time occurs the simulation framework is paused, and the simulated load data is extracted from PCM and populated as CACTOS runtime models instances. Then the optimisation algorithm is evoked using the interface data structure presented in Figure 5-8. Once the request is processed the optimisation plan is returned back to the CactoSim and is implemented in both CACTOS runtime models and PCM by using the CactoSim VMI and PCM VM components respectively. After the optimisation plan is enacted onto both models the simulation process is resumed using the updated models.

```
01. public interface InitialPlacementAlgorithm {
02.
03.     public OptimisationPlan generateOptimizationPlan(
04.                                     PhysicalDCModel pdcm,
05.                                     LogicalDCModel ldcm,
06.                                     PhysicalLoadModel plm,
07.                                     LogicalLoadModel llm,
08.                                     List<VirtualMachine> vmsToPlace);
09. }
```

Figure 5-10: CactoOpt VM placement API

During the regular day to day cloud data centre operation the new and existing users typically create new VMs and close running VMs. In reality, during these new VM

admission requests the cloud management system (such as OpenStack) would send a request to CactoOpt to ask for decision support as to where the new VMs need to be admitted. To reflect this in the simulation environment the model of new VMs is submitted to CactoSim containing new VM configurations and starting times. The VM placement functionality of CactoOpt is then triggered using the VM placement API shown in Figure 5-11. During the placement call the simulation data is converted into the CACTOS runtime model. This is sent to a selected placement policy together with the list of VMs. Once the placement policy has processed the request, an optimisation plan is sent back with the placement decisions for the new VMs.

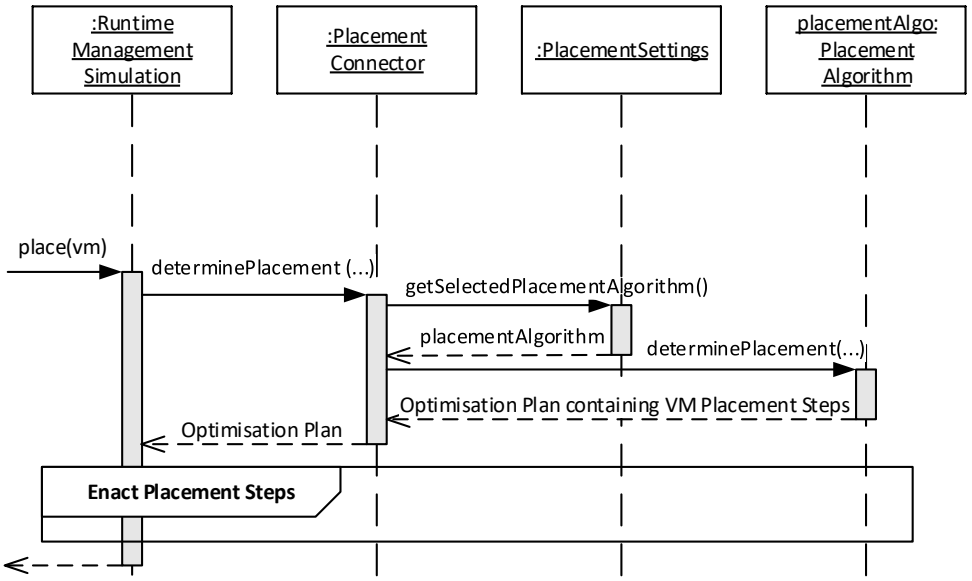


Figure 5-11: Sequence diagram of placement algorithm execution

The sequence of steps shown in Figure 5-11 includes calls within CactoSim to the placement connector to determine the placement algorithm to use based on the information specified by the user in the simulation experiment. The optimisation plan is then pushed to the CACTOS runtime models and PCM using the VMI as shown in Figure 5-9.

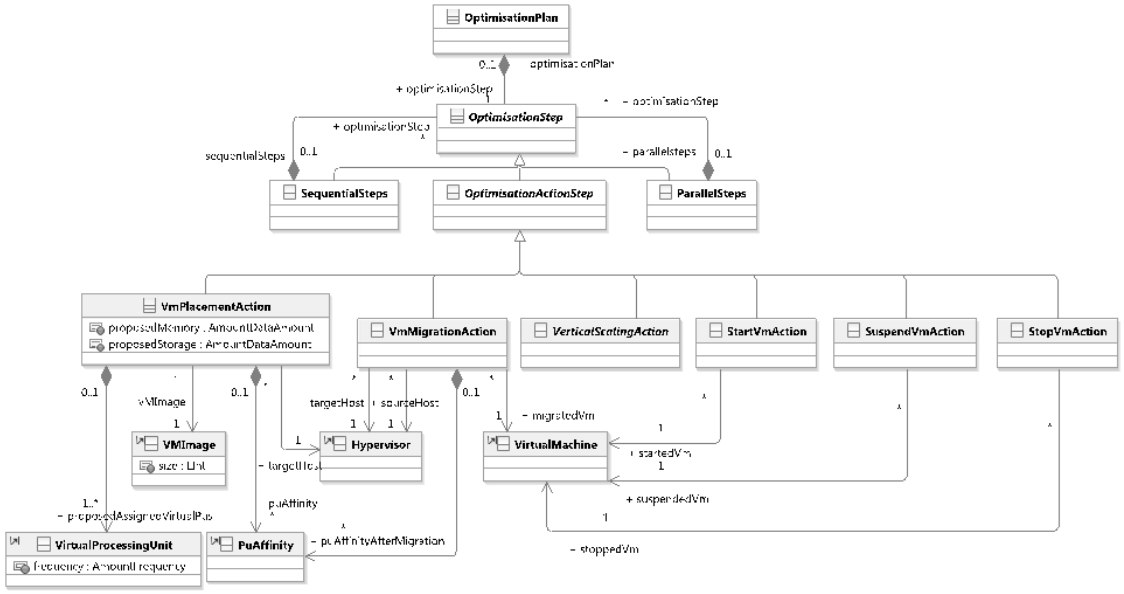


Figure 5-12: Optimisation Plan model (Krzywda, Ali-Eldin, *et al.*, 2014)

The Optimisation Plan model is also implemented using EMF technology which helps to enforce data standardisation and integrates with other EMF based CACTOS runtime models. As given in Figure 5-12 the optimisation plan consists of optimisation steps that can be executed sequentially or in parallel with no particular order. Each optimisation step can be expressed as VM placement, VM migration, VM vertical scaling and VM start/stop/suspend action. The VM placement action contains information on VM resource configurations such as vCPU, storage, memory and information about the hypervisor where the VM should be placed. The VM migration action contains a source host where a VM is currently located and a target host where a VM should be moved. The remaining actions contain the attribute updates for a VM object that need to be enacted in the system.

5.4.5 Cost model implementation

The cost model is implemented in conjunction with the CACTOS runtime model to get the list of the equipment in the data centre data hall area and the utilisation time series data. For consistency and ease of integration the EMF technique was used to produce an Ecore based meta model.

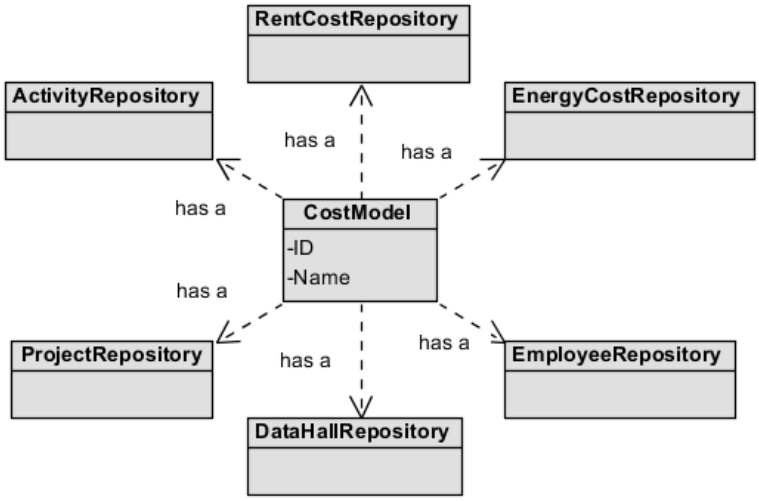


Figure 5-13: Cost model root element class diagram

Referring to Figure 5-13, the root model element CostModel has two attributes: a unique, generated identifier, and a human readable name. In addition, the model includes six repository elements: ActivityRepository, RentCostRepository, EnergyCostRepository, EmployeeRepository, DataHallRepository and ProjectRepository. Each "repository" element represent a collection of elements and their attributes that used within the cost model.

The employee records their working time spent on different tasks in the data centre. Time entries would be then recorded against an ongoing project denominator. By using this arrangement, costs are separated by type of task and by the project allowing for more granular reporting and cost analysis. This data structure was chosen to fit the commonly followed ERP timesheet recording pattern, to allow for easy mapping with the cost model elements.

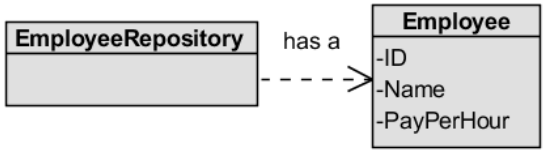


Figure 5-14: Employee repository class diagram

The *EmployeeRepository*, shown in Figure 5-14, contains a list of people that are working on the tasks related to data centre operations. Each *Employee* object contains a unique ID number, name and their pay per hour rate.

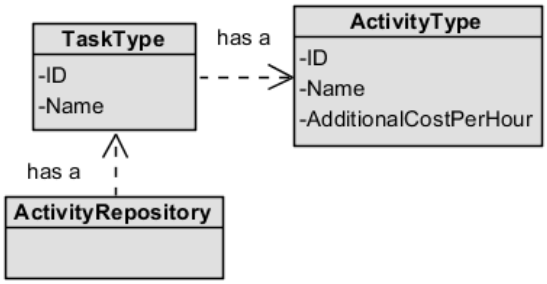


Figure 5-15: Activity repository class diagram

The ActivityRepository object, shown in Figure 5-15, contains the list of tasks that an employee can perform. The task type contains ID attribute and a Name attribute and are further branched into more detail activity types. The ActivityType object contains also an ID and a Name attributes plus an additional attribute of additional cost per hour. The additional cost is available for accounting for activities that result in complimentary cost compensation to an employee, such as night shift, overtime or working during holidays.

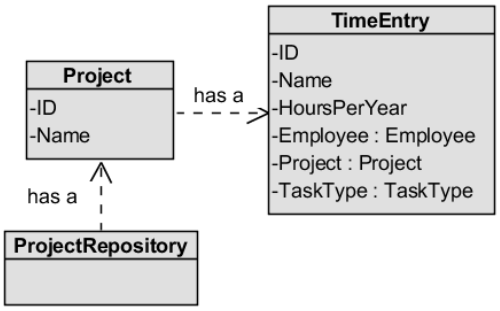


Figure 5-16: Project repository class diagram

The ProjectRepository object, shown in Figure 5-16, contains the list of ongoing projects in context of operations inside a cloud data centre. Each Project object inside the repository has an ID and Name attribute and also a time spent by employees working on the project recorded by a list of TimeEntry objects. The TimeEntry contains information on how much time each employee spent working on a particular project performing what task. Each TimeEntry provides a link between Employee, Project and TaskType enabling labour cost to be broken down to report on a project and task type level.

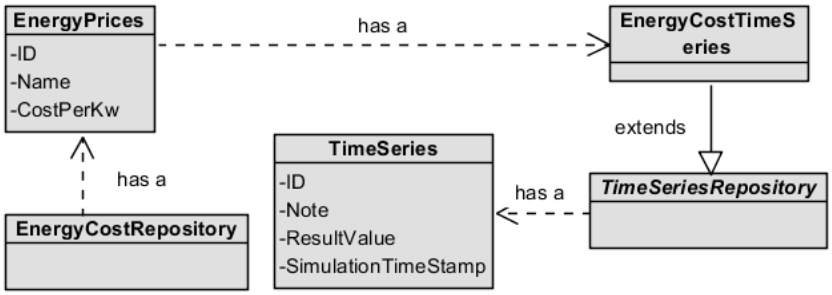


Figure 5-17: Energy cost repository class diagram

Apart from labour, the cost of hardware must be accounted for by taking into account the costs of energy and property. To capture a range of prices and price plans offered by a number of suppliers the EnergyCostRepository object, shown in Figure 5-17, enables the maintaining of a list of electricity providers by hosting a collection of EnergyPrices objects that has ID, Name and CostPerKw attributes. In addition, to account for overtime price fluctuations, the EnergyCostTimeSeries can capture price variations over the time of a single energy price plan and incorporate that with the simulation model.

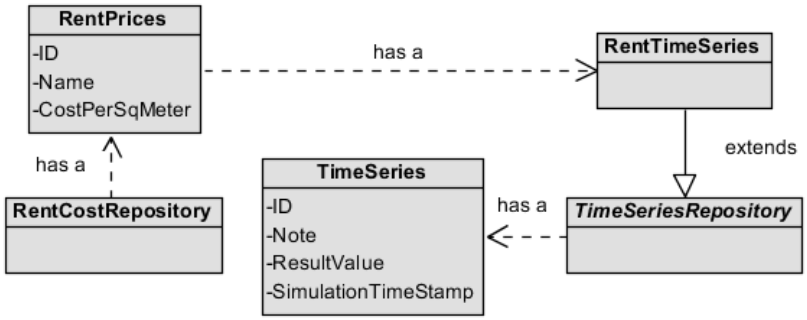


Figure 5-18: Rent cost repository class diagram

Similar to the energy cost repository, the RentCostRepository captures a list of rent prices per square meter. Each entity of RentPrices can reflect the property price for a specific geographical area or a quote received from different lenders. Also, as rent prices can fluctuate time series data can be created to account for drops and rises in costs.

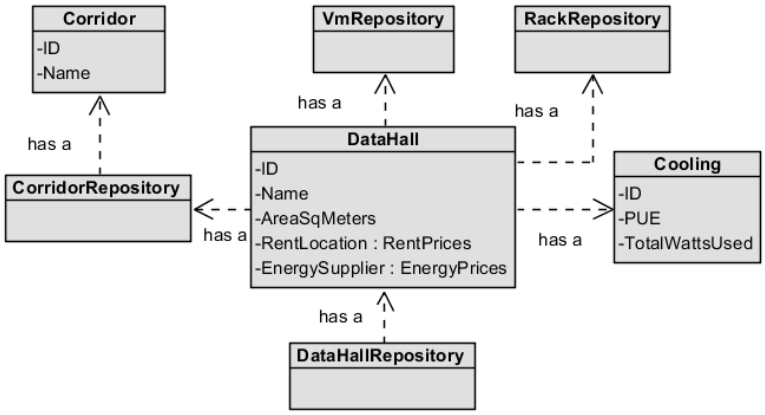


Figure 5-19: Data hall repository class diagram

The DataHallRepository object, shown in Figure 5-19, contains a collection of DataHall objects that represent an area within a data centre premises where the compute resources are situated. The DataHall object contains ID, Name, AreaSqMeters, RentLocation and EnergySupplier attributes. The area estimation is used for the rent cost calculation and the data of an energy supplier is used for electricity bill calculations. The cooling equipment energy consumption is not part of simulation, but estimated costs can still be accounted for by using Power Usage Effectiveness (PUE) ratio coefficient or if the cooling system or a direct measurement of power consumption for a specific period of time. The cost model provides a Cooling object that contain both PUE and TotalWattsUsed attributes to hold such data if it is available.

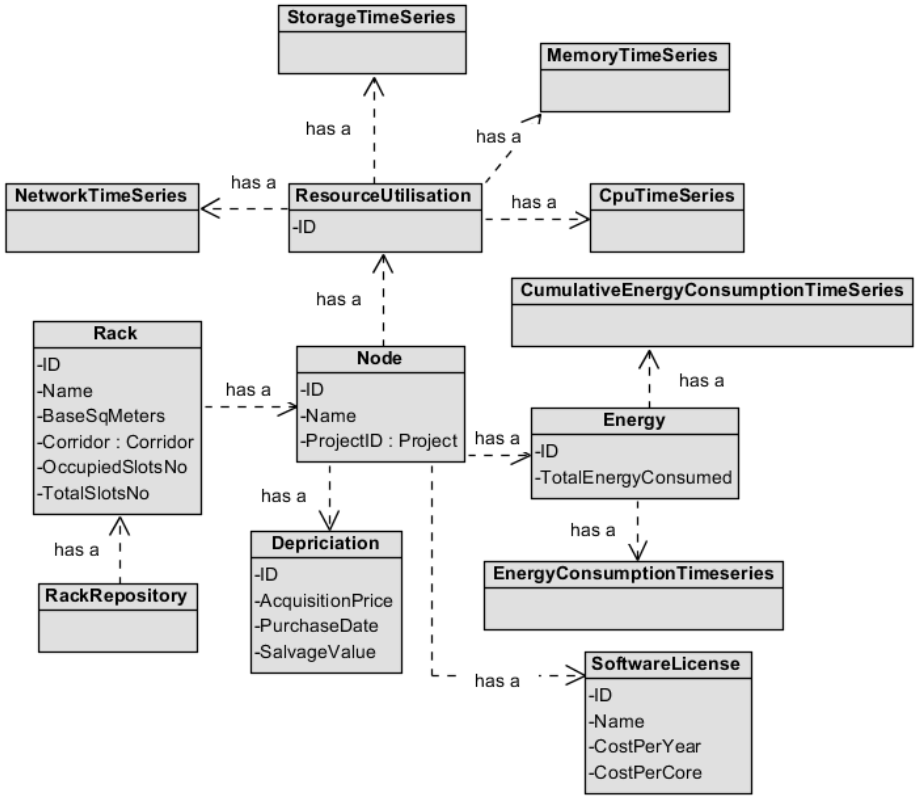


Figure 5-20: Rack repository class diagram

The RackRepository object, shown in Figure 5-20, is also a part of the DataHall object. It contains information on the number of physical racks and the number of physical nodes within each rack. Each Rack element has an ID and Name, base area measurements, corridor assignment, a total of slot number, an occupied slot number and a collection of nodes. The base area measurement, corridor assignment, and slot capacity information is available for spatial layout planning variations where nodes information is used for hardware acquisition and exploitation cost estimation. The Node model object contains the cost information for depreciation, energy, resource utilisation and software licence calculations. The energy and resource utilisation metrics are contained in a form of time series data that is taken from the CACTOS runtime models.

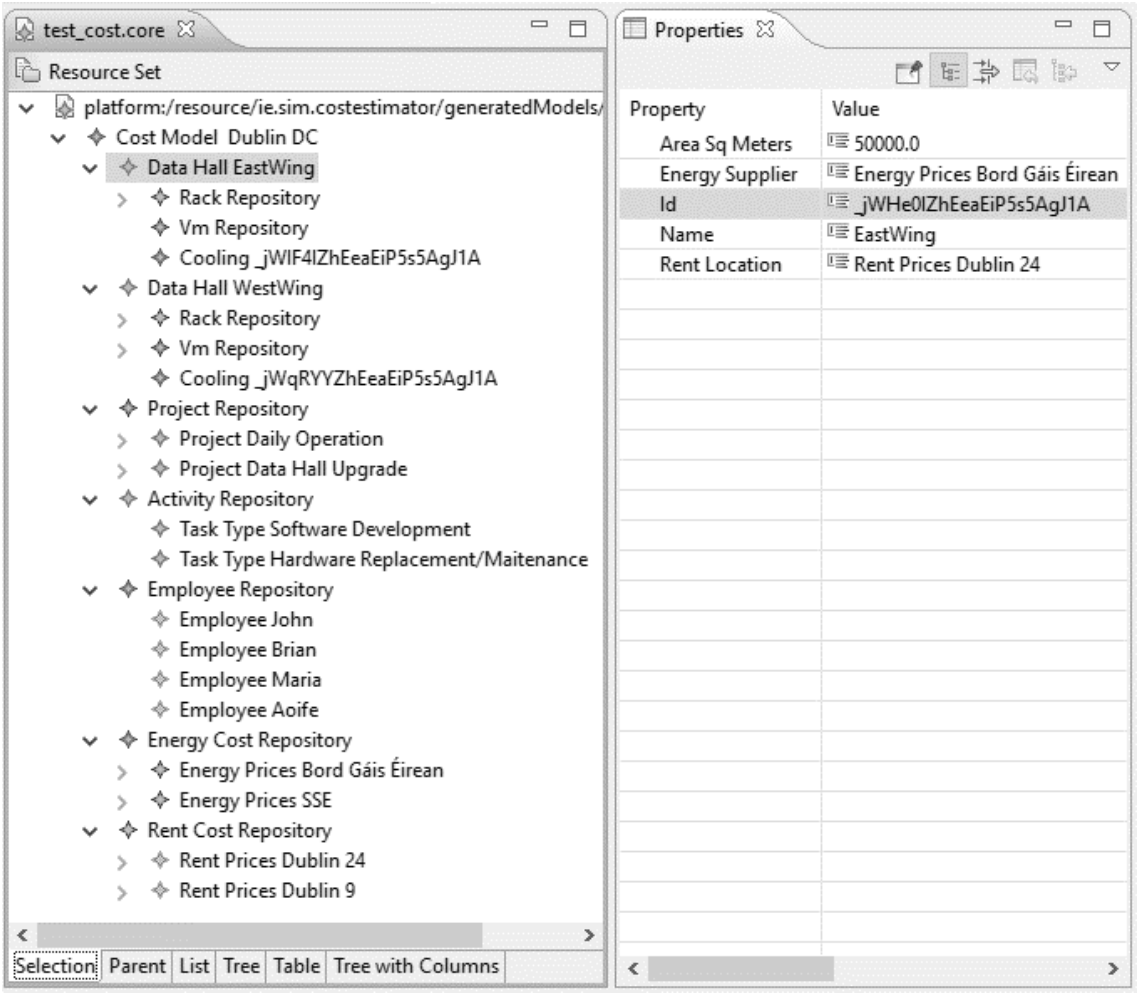


Figure 5-21: Cost model EMF based implementation UI screenshot

The implemented cost model also is available to view through a standard EMF graphic user interface. An example of populated cost model is shown in Figure 5-21, which shows a hypothetical Dublin based data centre instance. The “Dublin DC” consists of two data halls “EastWing” and “WestWing” each having its set of racks, VMs and cooling configuration. Each data hall can be assigned to an electricity provider and can be placed in “Dublin 24” or “Dublin 9” district rent price range. There are also four employees “John”, “Brian”, “Maria” and “Aoife” that can perform tasks of “Software Engineering” or “Hardware Replacement/Maintenance” for either “Daily Operation” project or the “Data Hall Upgrade” project.

5.5 User interface

The CactoSim simulation platform is designed for offline user interaction in the form of model acquisition, model design/modification, simulation experiment parameterisation and result analysis. The User Interface (UI) plays an important role in providing access to all of the previously mentioned functionality as well as to overall user experience and simulation framework adoption. The following section provides an illustrated step-by-step structured walkthrough describing the core UI functionality of the CactoSim implementation.

CACTOS Runtime model acquisition

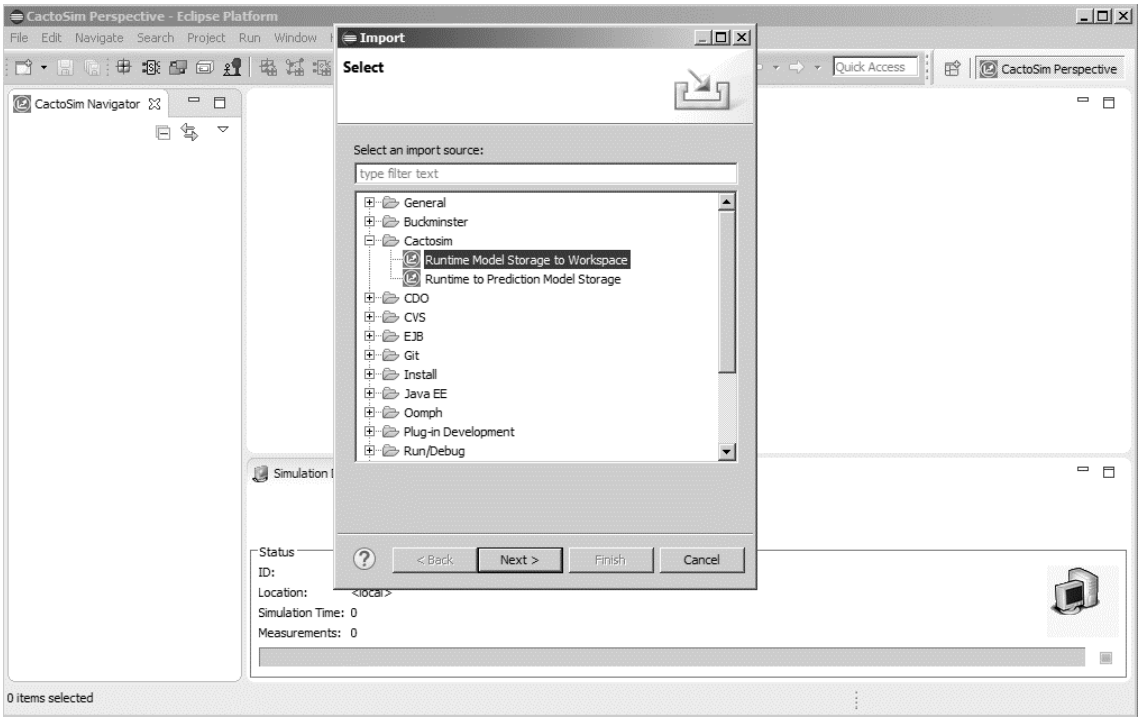


Figure 5-22: Acquire runtime model - import

The model is acquired from the data collection framework (CactoScale) through a CDO powered API using the standard Eclipse project import functionality. To access this option in the UI, a user must navigate to the “File-Import” menu and from the “Import” dialog window select the CactoSim import function section and select “Runtime Model Storage to Workspace” option and click “Next”, as shown in Figure 5-22.

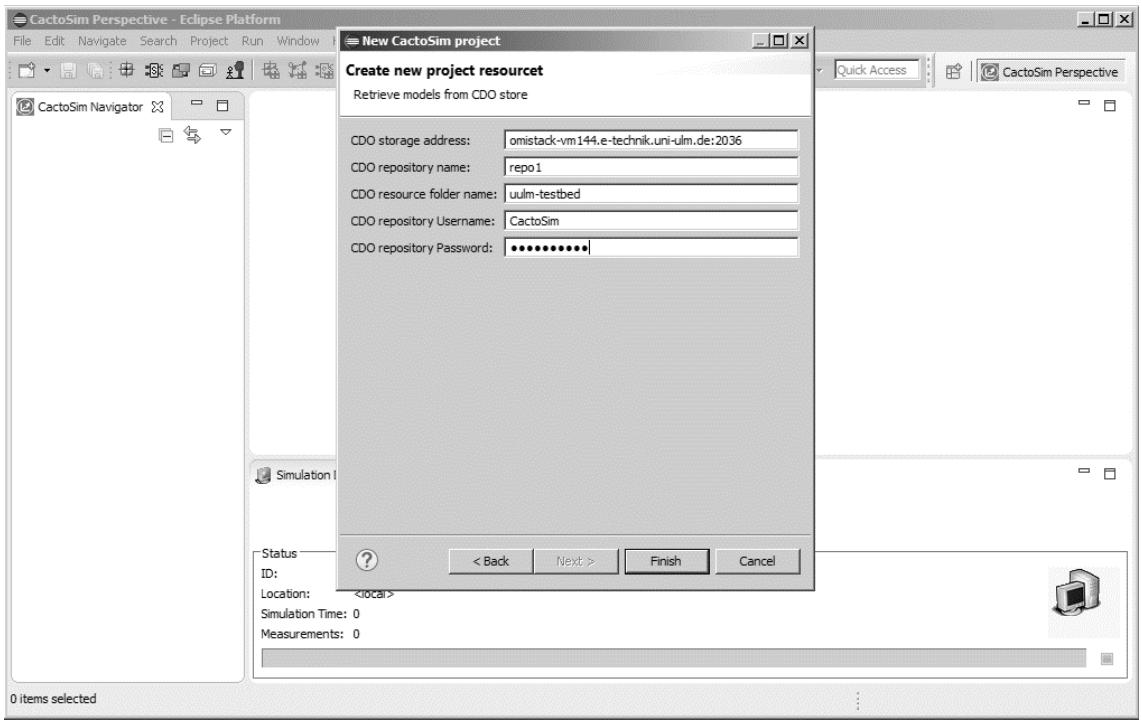


Figure 5-23: Acquire runtime model – CDO connection

In the next screen, shown in Figure 5-23, the user is prompted to supply the address and access credentials required to establish a connection with CDO server. The connection details include the address name and the port number of CDO server, name of CDO repository and name of the folder on the server where the models are located. The credentials consist of username and password. Both credentials and CDO repository configuration details must be provided by the CDO server administrator with a minimum of a Read access level.

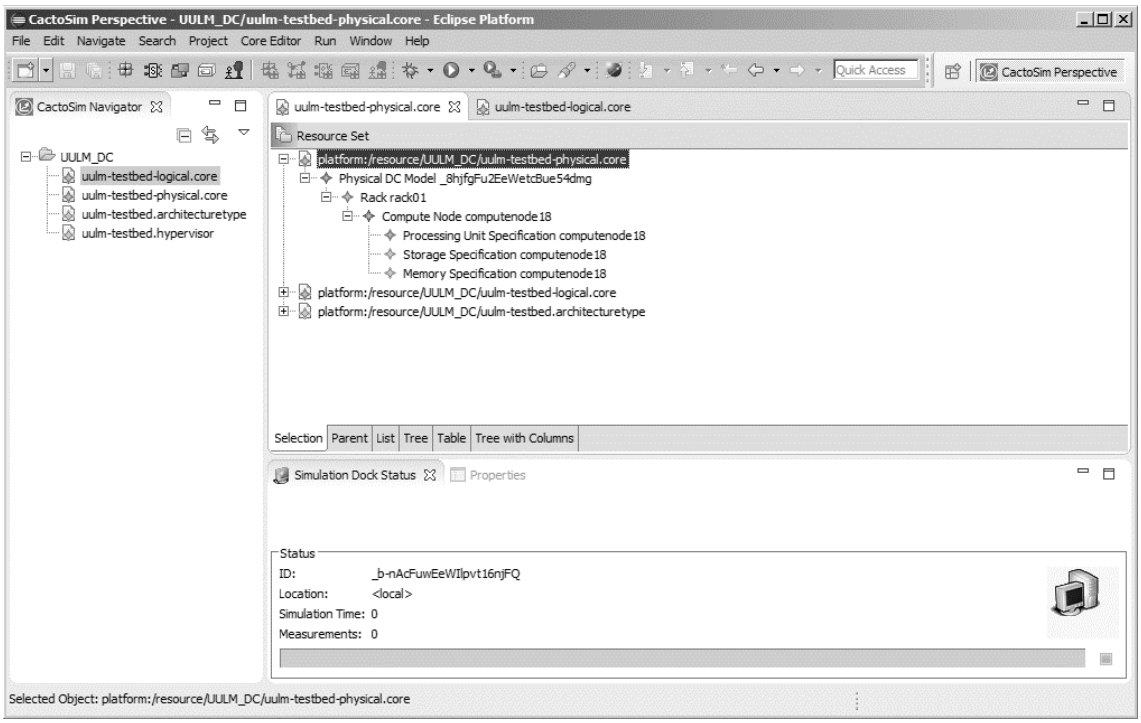


Figure 5-24: Acquire runtime model – final result

If the configuration details and credentials were correct the session will be established with the CDO server and models will be serialised in form of EMF core files to the newly created Eclipse project. Each model can then be viewed and modified if needed or used directly “as-is” to perform simulation experiments. Figure 5-24 shows the final result of model acquisition from the CactoScale framework running inside of a live testbed environment.

Creating model elements

Each model can be modified or created completely from scratch using the UI. To add a model element the user can right click on an element and add a new child element or add a new sibling element.

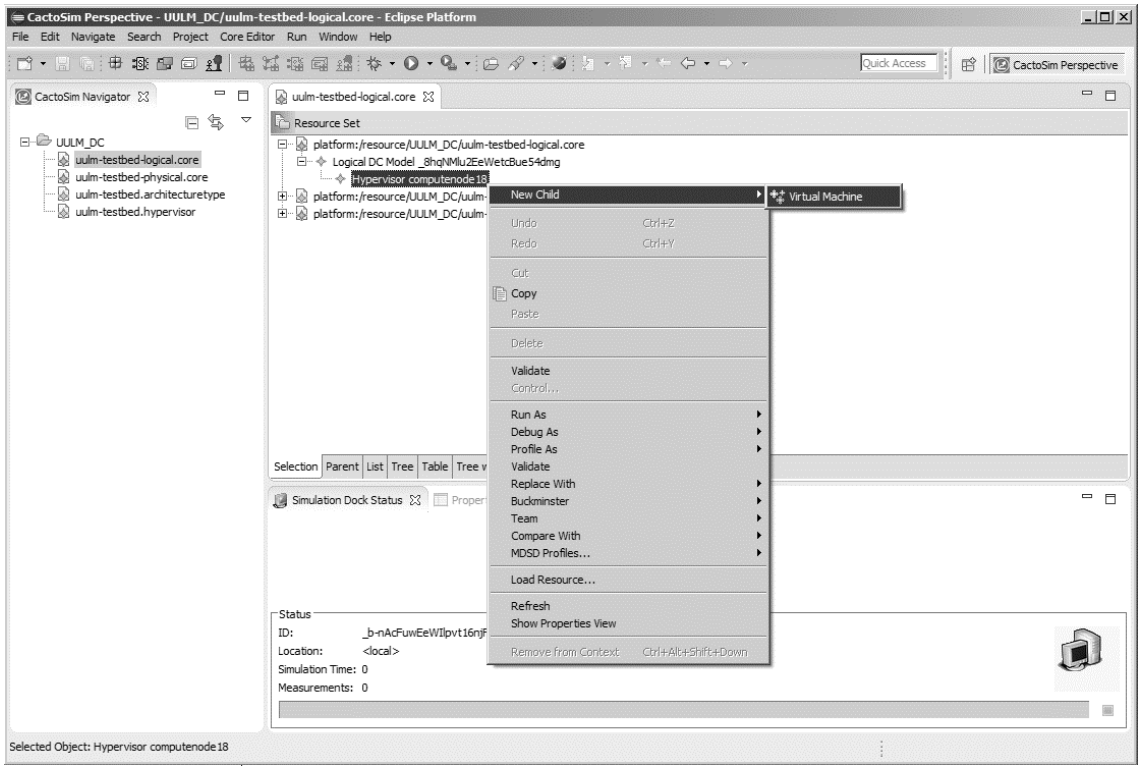


Figure 5-25: Adding new VM element to the runtime model

The screenshot shown in Figure 5-25 illustrates the means of adding an additional VM to a Hypervisor object within the Logical DC model. Once the Virtual Machine element is added it also requires parameterisation of other composite elements within the VM object such as application model, image instance, memory and processing unit elements.

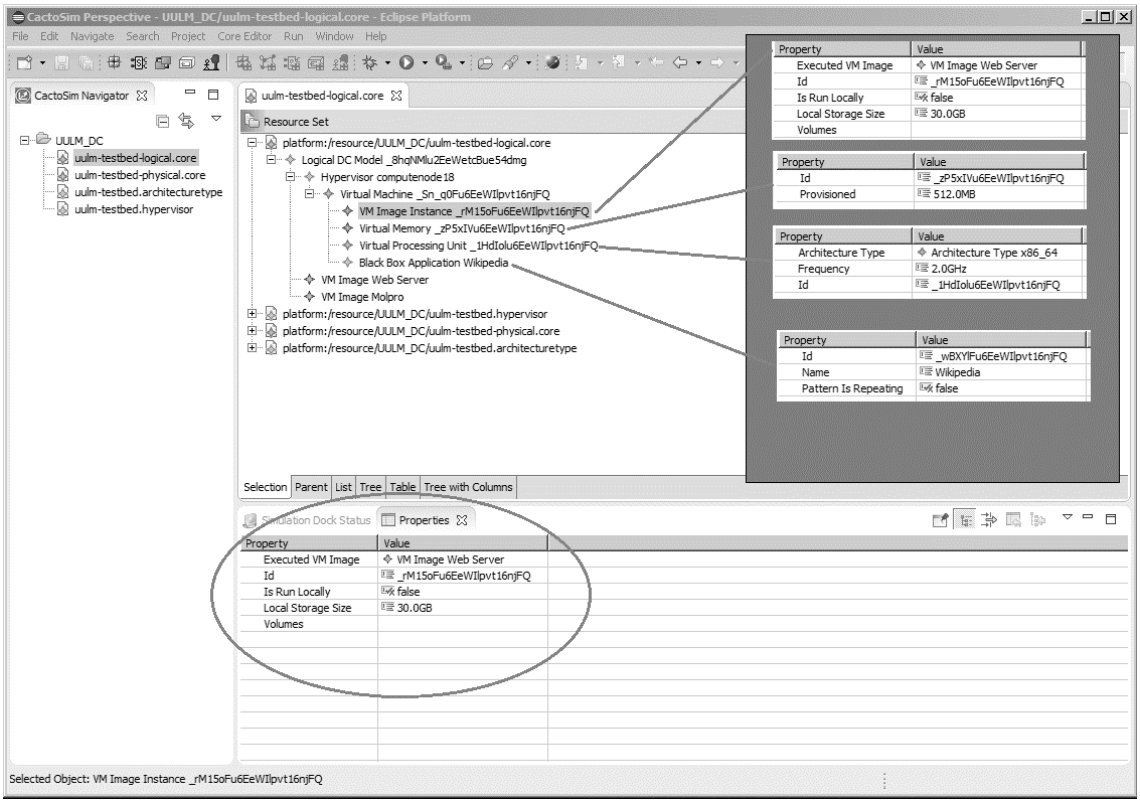


Figure 5-26: Logical data centre model VM parameters

Figure 5-26 illustrates an implementation example for the VM model instance content. Each element's attributes are displayed in the "Properties" UI panel and can be modified at the same location. In the example the added VM configuration consists of a local storage allocation of 30 GB, 512 MB memory, 2 GHz CPU and is mapped to a "Wikipedia" type of application model.

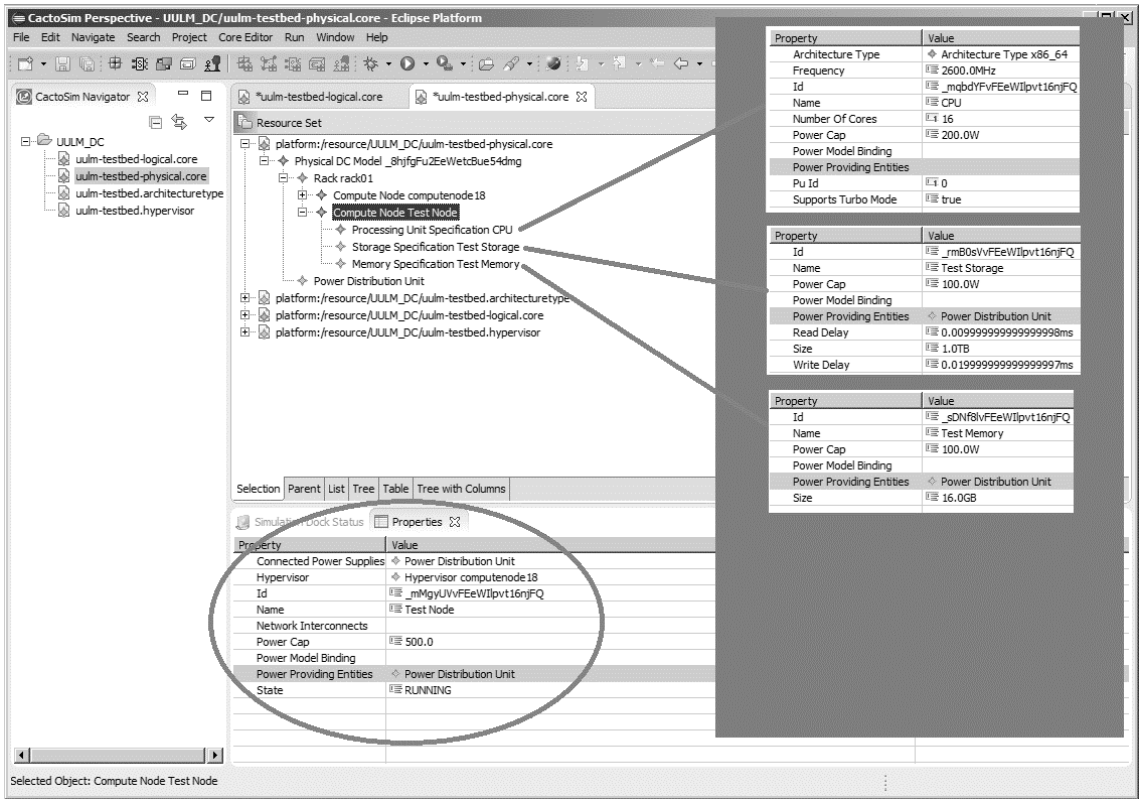


Figure 5-27: Physical data centre model node parameters

Similar to a VM, a compute node also can be added or modified within the model. The example shown in Figure 5-27 shows a new node entity being added to the CACTOS physical data centre model implementation. The added node has a hardware configuration provided which describes available storage, memory, and CPU capabilities. Since the hardware node energy consumption can be measured the energy bindings of hardware resources are also captured within the model.

Simulation Engine Parameters

After the cloud data centre models are created the simulation experiment can be executed. In addition to the simulation models, a simulation engine also requires additional configuration parameters for running the simulation experiment.

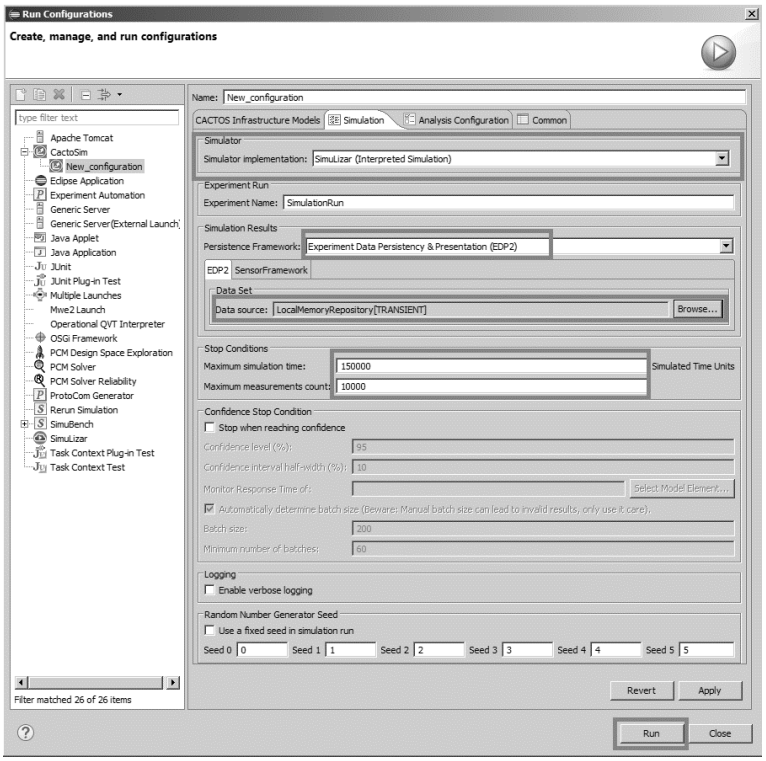


Figure 5-28: Simulation engine configuration

The Simulation engine parameters for CactoSim were implemented using the standard Eclipse UI library for creating a run configuration. The user is able to access this functionality from the top menu option by clicking on “Run -> Run Configurations”, right click on the “CactoSim” menu entry and selecting the “New” option. As shown in the Figure 5-28 a user is required to specify the type of simulation engine, the persistency framework for results storage, the storage location and simulation termination conditions. “SimuLizar” and “EDP2” are the only options available for CactoSim for the simulator and the persistence framework respectively. However, the result location can be saved in a specific file on disk in the Eclipse workspace or held temporarily in Random Access Memory (RAM). There are two available simulation termination conditions, the amount of measurement or the simulation time limit. If the user only requires one of these to be used, the other field must be set to “-1” to be ignored by the simulator.

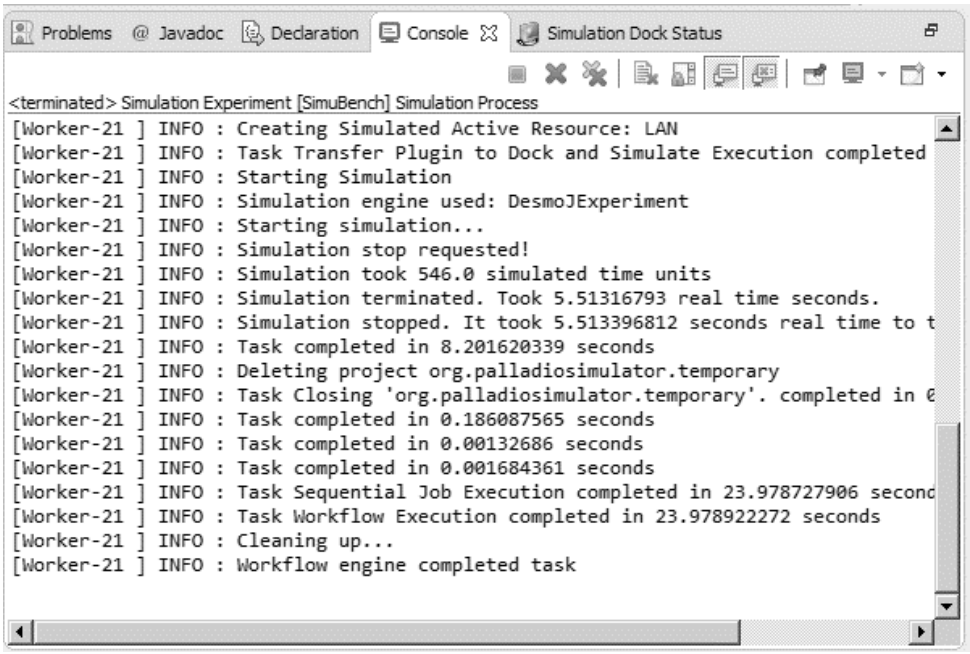


Figure 5-29: Simulation information console

During the simulation process execution, the Eclipse console functionality is implemented to provide user information about the progress of the simulation experiment. As shown in Figure 5-29, the user is provided with the information on model creation, simulation duration, task execution and memory management.

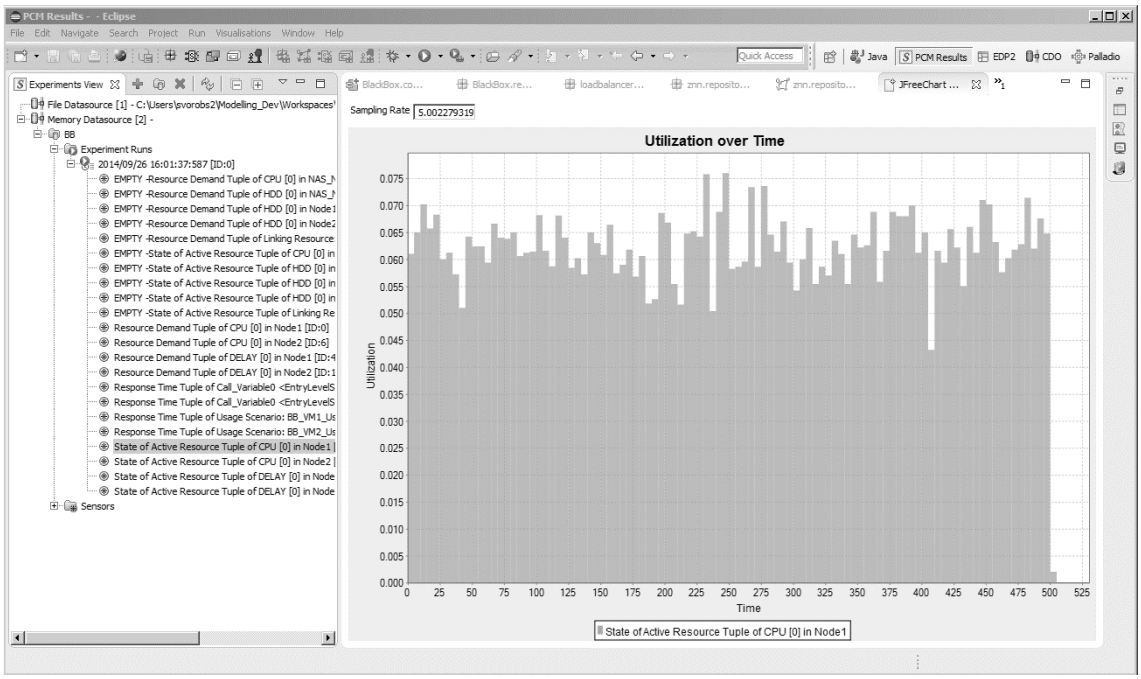


Figure 5-30: Simulation result analysis

After the simulation experiment run completes the simulation results are available under the PCM Results view which is part of Palladio and implemented using Eclipse UI libraries. The experiment results are available as a list of nested resource utilisation data over simulation time. The example provided in Figure 5-30 shows a CPU utilisation rate over time until the end of simulation experiment. Each graph can be also exported as a CSV file for visualisation and analysis using third party tools.

5.6 Conclusions

The chapter provides an overview of the design and implementation process, from the initial simulation system functional requirements specification through to a structured walkthrough of working implementation. The existing Eclipse IDE based simulation platform (Palladio) has been extended to form a new cloud data centre simulation framework (CactoSim). Eclipse IDE UI framework elements have been used and built upon to create the new CactoSim user control elements such as cloud data centre model retrieval, creation, and modification through the integration with the CactoScale CDO powered storage and EMF. The EMF also provides a mechanism for meta-modelling and automatic code creation for the CACTOS runtime models, optimisation plan models and cost models. The Palladio simulator functionality provides

access to the PCM models, the simulation engine and the simulation result viewer implementation through code integration and model transformations between PCM and CACTOS model components. As a result, the CactoSim simulation framework contains a foundation for an established simulation platform extended to fit cloud data centre simulation requirements.

The following chapter uses this CactoSim implementation to provide a range of experiments and demonstrates the tool working in an applied research cloud data centre scenario in order to provide enhanced cost estimations.

6 Experimentation and Analysis

6.1 Introduction

The previous chapter described the design, implementation and integration of the data collection, optimisation and simulation frameworks. These integrated components form a “simulation-driven toolkit” which provides decision-making support for data centre operators. The design of this toolkit is focused towards enhancing cost estimation for cloud data centres. It does this by providing the user with a deeper understanding of complex system behaviour under various experimental scenarios with a focus on the trade-offs of cost and QoS. As part of the solution, a user interface allows for the alteration of simulation models and new scenario development to assess different data centre design options and resource demand conditions. Such experimentation allows for resource distribution to be modelled under different usage scenarios, which aids in the selection of the closest-fitting hardware, equipment and resource management policies that satisfy QoS requirements and cost trade-offs for an enterprise. In order to assess the designed and implemented solutions capability for modelling such scenarios, the following chapter presents a range of experiments to demonstrate the validity of results in comparison to the real-world systems. Further, cost analysis experimentations demonstrating the methodology in an applied research scenario are described. In carrying out this demonstration, case scenarios from two real case companies were used Company X and Company Y.

Company X is a communication and information centre of a large educational provider, which delivers their IT function to external clients as well as supporting the organisation internally. Company X provides a catalogue of IT-related services such as document storage, backup and delivery, IT security, network and connectivity, and resources for scientific experimentation. For the purposes of experimentation, Company *X* provided access to their cloud data centre as well as related financial data, and a range of experiments were executed on this platform. *Company Y* is a software SME which provides proprietary cloud computing management and orchestration software, essentially enabling hosting companies to become cloud providers. The company provides some hosting internally to their clients for both live deployments and for testing purposes via the data centre based on site. Company Y provided access to an isolated

part of the data centre and range of cloud applications to use for validation scenario execution. Validation scenarios from both companies were used to build simulation experiments and compare simulation results against the real system results.

Two sets of experimentation are described in this chapter. The first set of experiments (*Section 6.2*) provides simulation model development and validation against real data. The purpose of these experiments is for model content verification, to ensure accurate model descriptions have been created, followed by simulated and measured results comparison to assess the precision and reliability of the simulation outputs. The second set of experiments (*Section 6.3*) uses these validated models to provide further experimentation and cost analysis in order to offer a more detailed understanding of how such models can be applied in the making of real-world decisions.

A day-to-day task of a cloud infrastructure provider in the IaaS scenario is to give access to compute resources requested by users. The users specify a number of CPU cores, memory and storage needed to be assigned to the VM which will host the cloud application. Different resource combinations are also pre-defined by the cloud resource provider as VM “flavours”. Once VM configuration is selected, the VM is then deployed onto virtualised hardware by a cloud resource management framework. It is then up to the resource management framework to identify which hardware is suitable to use for hosting new VMs. Resources allocated to the VM serve as maximum capacity indicators that are managed by the hypervisor. However, resource utilisation levels depend on different cloud applications and application usage patterns. For example, compute resource demand for downloading a web page will be different if it contains just plain text or returns a large database query. In the same way resource demand will also differ depending on the number of concurrent users of a cloud application. Considering application resource utilisation patterns and available virtual resource distributions across the data centre, the resource management framework can also migrate existing running VMs e.g. to minimise resource contention or improve resource consolidation. Resource management frameworks employ resource optimisation policies for VM admission and migration to effectively improve QoS, save costs and maximise profit. However, different optimisation policies can yield different results depending on individual data centre configuration and differences in the cloud applications resource demands. In *Section 6.2* two such cloud data centre management scenarios are

presented using *Company X* and *Company Y* test beds. The real data was extracted from a controlled environment created within the cloud data centres of *Company X* and *Company Y*, in which a range of system test scenarios were executed. Company X scenarios were based on the quantum chemistry software "*Molpro*" (Werner, Knowles, *et al.*, 2012) taking the role of a cloud application in line with the "*CACTOS scientific computing use case*" (*Section 3.2*). *Company Y* experiments were based on a variety of web applications in accordance with the "*CACTOS business analytics use case*" (*Section 3.2*). Based on the analysis, simulation models were developed using real cloud data centre hardware implementations including resource demand patterns and resource management policies. These cloud data centre models were then used in simulation experiments and simulation results were validated via statistical and visual comparison techniques against real system scenarios measurements. The validation work presented in *Section 6.2*, firstly, demonstrates the accuracy and reliability of the implemented framework through the use of real field studies and secondly, demonstrates applicability to the wider array of scenarios by experimenting with distinctively different case studies from two companies.

Equipment acquisition is a common task which is at the core of a cloud data centre business model. Cloud data centre operators are responsible for maintaining hardware in running order and compliant with modern user demands. New equipment can be bought as replacements for old equipment or as part of a data centre capacity expansion plan. Hardware equipment depreciation time can vary from 3 to 5 years depending on the type of hardware and its function. At the end of the depreciation period the old equipment is sold (if possible) and the new equipment is acquired. Brand new equipment can offer better processing capacity, greater energy efficiency and smaller form size. The processing capacity allows for faster user workload processing which can directly improve QoS and the number of users that can be served. Better energy efficiency allows for reductions in energy costs and while being more environmentally friendly through lower CO_2 footprint. Smaller equipment form size can lead to space savings, increasing the density of cloud data centres without the need to invest in building expansions to increase compute resource capacity.

However, there are multiple equipment vendors offering a multitude of hardware configurations at different price points. To select the right configuration of new equipment one must consider how it will impact existing users; if existing resource management

policies function as intended with the new equipment and if the cost of new equipment will be recovered, justifying the investment in the first place. *Section 6.3* describes experiments focused on the use of the proposed methodology for a scenario evaluation relating to the upgrade of data centre hardware in an existing data centre of *Company X*. In this scenario, the effects of a partial data centre equipment upgrade on QoS and costs under different resource management strategies were explored. Simulation models that were previously developed during the modelling and validation stage given in *Section 6.2* are used as the basis of this experimentation in order to build a realistic system representation in response to real-world cost-related decisions being taken.

The aim of the experimentation presented in this chapter is to cross theory into practice through focusing on an actual usage of the developed components together in a single methodology using real-world case study data. This is carried out firstly, to obtain and build cloud data centre system models using real data in an automated manner; secondly, to simulate and validate system behaviour including decisions relating to optimisation; and thirdly, to support cost-aware decision making through results gained from further what-if analysis. The outcome of this chapter is a number of analysed empirical results that provide evidence of the applicability and value of the use of the proposed methodology as part of a cloud data centre management approach.

6.2 Validation

Simulation model validity and credibility was undertaken throughout the proposed methodology with an emphasis placed on model content relevance in relation to representing real system attributes and behavioural traits. The model was designed and developed via a continuous collaboration cycle with the CACTOS projects subject matter experts, in this instance Company X and Y stakeholders. From the problem definition stage through to result analysis, simulation model components and outputs were verified on a continuous basis within the consortium and critically reviewed by a panel of experts appointed to the CACTOS project by the EU commission. “The most definitive test of a simulation model's validity is to establish that its output data closely resembles the output data that would be expected from the actual (proposed) system” (Kelton and Law, 2000).

In carrying out the validation process, use case analysis scenarios were designed and executed on the real cloud data centre testbeds of Company X and Y. Each scenario input consisted of a script that defined VM lifecycles and the resource management

optimisation configuration settings (shown in Figure 6-1). The VM lifecycle script imitates VM creation and termination requests including the information of VM resource and application configurations. Optimisation configuration settings define the choice for VM admission and migration policies used within the test scenario. In summary, the test scenarios were designed to show system resource management capabilities under different optimisation policies, handling VM admission to the data centre and migration decisions.

During the real system scenario, execution data on system configuration and behaviour was automatically collected, and simulation models based on this data were created automatically using the modelling tools developed within the simulation toolkit. Simulation models include the physical infrastructure, logical infrastructure and experiment scenario models. The physical infrastructure model contains a description of the data centre compute hardware and its specifications such as number of nodes, CPU cores, memory and storage. The logical infrastructure model contains workload information for executed applications, VM configurations and hypervisor relations to hardware components. The experiment scenario models contain information on the VM admission and termination times having the same function as real system VM lifecycle execution scripts. The models obtained are an exact replica of the system and user resource demand patterns and are used without modification for validation. To increase confidence in the simulation model two different cloud data centre systems were modelled and simulation outputs compared side by side.

The presented research looks at the problem of resource provisioning and cost estimations from the IaaS provider viewpoint. The infrastructure is provided on-demand and can be used for any application and workload processing depending on the need of a customer. Such business models lend themselves towards the infrastructure provider being agnostic as to what cloud application is deployed in a VM or how it might be connected to other VMs in the data centre. To model IaaS resource demand a Black-Box application modelling technique developed as part of the CACTOS project was used in the validation exercise (Groenda and Stier, 2015). The Black Box model captures individual VM resource demand patterns and then allows for a “replay” of these demands in the simulation, replicating resource demand over time for each admitted VM.

As shown in Figure 6-1 the real system scenarios and simulation experiments were both executed using the same inputs of VM lifecycle and optimisation

configurations. The automated simulation models describing testbeds were created to represent the real system environments. In addition, during the real scenarios and simulated experiments the same resource management policies were used via the resource management framework. This setup allowed for the creation of a mirror image of the real system and associated scenarios in the simulated environment, meaning that the outputs of the simulated and real systems if designed correctly should be comparable given that the experiments represent the same real system scenarios. Hence, the simulation results can be directly compared to the real system measurements in order to validate the simulation models accuracy and reliability. It is generally understood that simulation data will not be exactly equal to the real-world measurements due to high abstraction levels. In recognition of this and based on past experience the project consortium agreed that the simulation results will be successfully validated if the average error rate is equal to or lower than 10%.

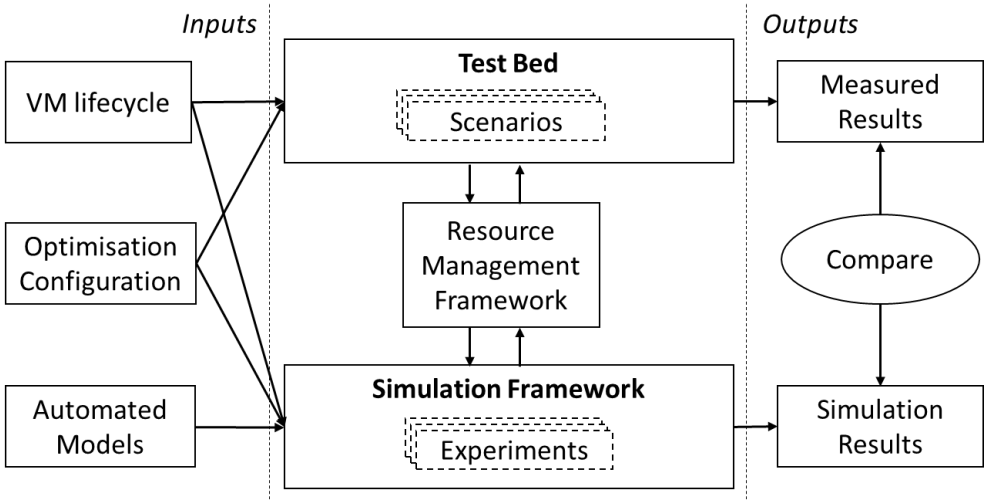


Figure 6-1:Validation flow

The validation exercise was performed to ensure that the developed methodology and toolkit operated as intended and produced results that can be trusted to support decision making. In order to ensure that the simulations experimental results are both accurate and reliable, visual and statistical validation was carried out where a comparison was made between the simulation results and the outputs from two field test experiments executed in the real case environments. To distinguish between real and simulated environments, the tests executed in the real system are called “scenarios” and

tests in completed using the simulation models are called “experiments” as also depicted in Figure 6-1.

6.2.1 Company X

In the initial step of validation, system testing scenarios are designed and then executed in a real data centre according to the design specification. During the execution, dedicated hardware remains isolated from other data centre users to create a controlled environment reducing any external factor influence on the experiment readings. During the scenario execution phase, data related to VM configuration, VM lifecycle and resource demand was being recorded by the data collection framework for later use in the modelling and experimentation stages.

ID	CPU			RAM	Disk
	Make	***Cores***	***Frequency***		
Cloud Controller	2x Intel Xeon 6-Core Westmere	12	2920 MHz	48 GB	2x 1TB HDD
Network Controller	2x Intel Xeon 6-Core Sandy Bridge	12	2000 MHz	64 GB	2x 1TB SATA HDD
NAS Node	2x Intel Xeon 6-Core Sandy Bridge	12	2000 MHz	64 GB	2x 500GB HDD 6x 2TB HDD
computenode02	2x Intel 8-Core Sandy Bridge	32	2600 MHz	128 GB	190 GB
computenode05	2x Intel 8-Core Sandy Bridge	32	2600 MHz	64 GB	190 GB
computenode08	2x Intel 8-Core Sandy Bridge	32	2600 MHz	64 GB	No local storage
computenode11	2x Intel 8-Core Sandy Bridge	32	2600 MHz	128 GB	190 GB
computenode12	2x Intel 8-Core Sandy Bridge	32	2600 MHz	128 GB	866 GB
computenode14	2x Intel Haswell 8-Core	32	2400 MHz	128 GB	No local storage
computenode15	2x Intel Haswell 8-Core	32	2400 MHz	128 GB	No local storage

computenode16	2x Intel Haswell 8-Core	32	2400 MHz	128 GB	No local storage

Table 6-1: Allocated experiment testbed infrastructure at Company X

The infrastructure of the *Company X* testbed is shown in Table 6-1, consisting of 11 nodes in total. However, the nodes marked as “Cloud Controller”, “Network Controller” and “Network Attached Storage (NAS)” are not used for VM deployments. The cloud and network controller nodes are used to host OpenStack and CACTOS Runtime components to manage compute resources of the cluster. The NAS node is used for hosting non-volatile storage which can then be mounted over the network to deployed VMs. Compute nodes available in the data centre are equipped with Intel (Intel Corporation, 2018) manufactured CPUs, which come from two different microarchitectures Haswell and Sandy Bridge. While both CPU types have the same number of cores the Newer Haswell CPUs come with a lower nominal frequency of 2400 MHz. Most of the compute nodes are equipped with 128 GB of memory with the exception of “computenode05” and computenode08” which only have 64 GB available. Also, half of the compute nodes have no local disk storage attached, relying on NAS virtual storage containers to be used over the network when provisioning VMs.

Scenario	Placement Policy	Migration Policy
OS-A	System default	none
OS-B	Memory based load balancing	Load balancing
OS-C	Memory based consolidation	Consolidation
OS-D	Consolidation	Constraint programming-based consolidation
OS-E	Fragmentation	Fragmentation
OS-F	Energy efficiency	Energy efficiency

Table 6-2: Experiment list with optimisation policies

During the experimentation phase, a total of six isolated experiments were carried out on the *Company X* testbed using different workload optimisation policies. All six experiments with their associated optimisation policies are shown in the Table 6-2. As can be seen in the table, the first experiment “OpenStack-A” (OS-A) is executed to develop a baseline system performance estimation without introducing any optimisation decisions from the resource optimisation framework - CactoOpt. During the OS-A

experiment the default OpenStack VM admission policy was used which is controlled by the default policy which strives to admit workload using a round robin approach. As no optimisation policies were used this means that the VMs were not migrated, remaining at all times on the node on which they have been initially placed during the experiment run. The remaining five experiments were executed with the CactoOpt optimisation framework fully engaged and taking over VM placement and periodic migrations for the purpose of enhanced resource management. Each tuple of the optimisation policies was selected to work together using the same optimisation goals:

- ***Load balancing*** – aims to spread VMs among all compute nodes taking proportion of their sizes into account.
- ***Consolidation*** – aims to reduce number of compute nodes used by VMs by placing and migrating VMs as compactly as possible.
- ***Fragmentation*** – aims to reduce compute resource wastage caused by division in demand between CPU and memory by grouping VMs with certain flavours together depending on user demand and hardware specification.
- ***Energy efficiency*** – aims to reduce total power consumption for cloud data centre by combining power saving features of hardware and managing VM distribution.

As shown in Table 6-2, VM placement and migration policies were implemented in tuples in order to complement each other and not be counterproductive. The first scenario (OS-A) was executed with OpenStack default configuration which only has a VM admission policy. The default admission policy picks a first node in the rack and places VMs there until all CPU cores are assigned. After that the same procedure is done on the next node in the rack. The second experiment (OS-B) was executed with CactoOpt framework using the memory-based load balancing algorithm for VM placements and load balancing algorithm for VM migrations. The newly created VMs were admitted to the system on the node that has the largest amount of free memory, and if the resource utilisation reached peak measurements, VMs were migrated to the nodes with more free resources available. The third scenario (OS-C) is the complete opposite to the load balancing approach, which means that instead of spreading VMs evenly across all available nodes, the policies are placing VMs on as few nodes as possible. The memory-based consolidation uses memory utilisation measurements to

determine suitable nodes where VM can be placed. The consolidation migration policy ensures that VMs are co-located as tightly as possible and if a VM was shut down, creating a resource gap on the node, the policy will migrate a fitting VM from another node.

Experiment OS-D was executed using constraint programming (Rossi, Van Beek and Walsh, 2006), an optimisation based consolidation approach which is able to take multiple constraint parameters and consider multiple parallel VM migrations at the same time. The approach is using the constraint programming solver which is able to obtain multiple data centre resource provisioning options and evaluate them using a set of global functions. Based on the evaluation results, the best suited option for resource allocation is selected. The resource fragmentation/stranding avoidance algorithm was used for scenario OS-E testing which tries to keep utilisation across CPU and memory at even levels. This approach allows for the avoidance of large "pockets" of resources which cannot be used for VM deployments. For example, when all CPU cores are already assigned to VMs, but large quantities of memory remain unused or vice versa. The energy saving optimisation approach used in the final scenario OS-F estimates energy consumption of the whole data centre and makes the resource distribution decision based on the lowest predicted configuration (Krzywda, Rezaie, *et al.*, 2015).

VM Sizes	CPU Cores	RAM (GB)	HDD (GB)
XSmall	1	8	80
Small	2	8	80
Medium	1	12	168
Large	2	12	168
XLarge	4	12	168

Table 6-3: Admitted VM configurations and quantities for Company X

During the system testing scenarios VMs are deployed to the testbed to recreate an operational users load on the system. The flavours of VMs (shown in Table 6-3) are divided into 5 categories – extra small, small, medium, large and extra large. Each flavour requires a specific amount of resources for successful deployment ranging from 1 CPU core, 8GB of memory and 80GB of storage to 4 CPU cores 12GB of memory and 168GB of storage. The amount of VMs admitted is restricted by the capacity of the testbed infrastructure, for example more extra small VMs can be hosted than extra large. The

VMs are admitted using deployment scripts with five minute intervals in a round robin fashion. Once each VM is deployed, the Molpro application task is executed inducing workload on the assigned resources. During the execution of the test scenarios, resource demand is measured periodically over the duration of each scenario in the form of CPU, memory storage and power consumption. The quantity of provisioned resources is directly proportional to the number and flavours of admitted VM's.

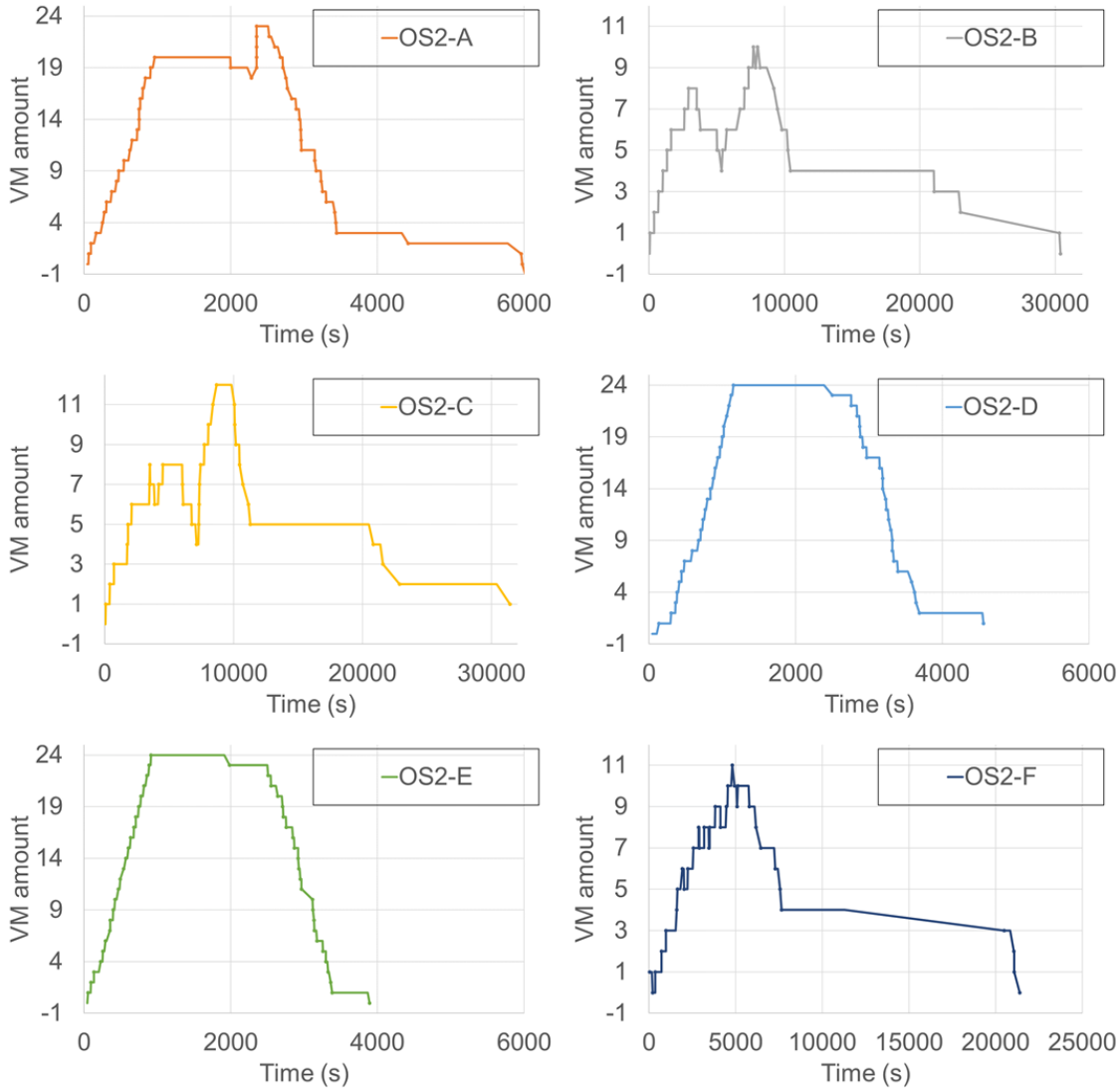


Figure 6-2: Company X testbed VM admission experiment scenario

Figure 6-2 shows the recorded VM admission patterns for all experiments. From the graph it can be seen that the experiments OS-A, OS-D, and OS-E have a similar, steeper, VM admission curve due to the shorter experiment times and a higher amount

of admitted VMs. The experiments OS-B, OS-C and OS-F, on the other hand, have a longer duration and lower total amount of VMs submitted during the execution. Different VM admission patterns were created specifically to investigate different types of scenarios testing system behaviour under demand spikes (e.g. OS-A, OS-D, and OS-E) and steady operation. The experiment OS-A has a value of "-1" for VM admission which means that the VM was admitted outside a monitored time interval.

Name	CPU provisioning (%)		Memory provisioning (%)		Storage provisioning (%)	
	Avg.	Peak	Avg.	Peak	Avg.	Peak
OS-A	8.2	16.8	13.2	30.7	0.1	0.3
OS-B	4	6.6	6.4	10.6	0.5	0.7
OS-C	4.2	7.8	6.8	12.3	0.5	0.8
OS-D	7.4	14.8	12	24	1.4	2.8
OS-E	7.2	14.8	11.7	23.8	1.3	2.8
OS-F	4.3	7.4	6.3	11.4	0.4	0.8

Table 6-4: Experiments resource provisioning summary

In certain cases of resource shortages, compute resource double booking can occur, meaning the same compute resource can be assigned to multiple VMs e.g. CPU core. From the resource provisioning summary presented in the Table 6-4 it can be seen that the dedicated testbed has sufficient capacity to cater for all of the experiment VM admission demands without double booking. The highest rates of demand for the *Molpro* application were observed under the memory component relative to the available resources within compute nodes on the testbed. The highest peak memory amount provisioned was during the OS-A experiment of 30.7% which corresponds with the highest number of VMs admitted when comparing to other experiments. Peak CPU core demand was observed at 16.8% also for the OS-A experiment and peak for storage demand at 2.8% was captured during OS-D and OS-E experiments. The Company X testbed contains a sufficient quantity of nodes to spread the VMs, hence resource shortage is not observed. However, memory demand is noticeably higher than CPU, suggesting that memory shortage prior to CPU shortage should the demand grow in the same manner.

Figure 6-3: Company X testbed measured power consumption

For each real system test scenario data on power consumption was collected by reading power meters which were connected to compute nodes 5,8,11,12,14 PDUs, every 10 seconds. Due to physical constraints compute nodes 2 and 16 had no power meter attached and were therefore excluded from validation comparison datasets, presented later. Figure 6-3 shows the power consumption fluctuation of all experiment results, the magnitude of power draw can be directly correlated with the workload increase created by arriving VMs. As expected, the highest power draw spikes adhere to the experiments with higher admitted VM count i.e. OS-A and OS-E with the exception of the OS-D experiment. This discrepancy for the OS-D experiment arises from the

placement policies extensive use of nodes 2 and 16 which are not power monitored and therefore not included in this analysis.

Scenarios were executed sequentially one-by-one on the assigned test bed hardware. During each execution, resource utilisation was monitored continuously, and polled data was stored within the CACTOS data collection framework. Further, the collected data was then specifically used as a direct input into the simulation model creation. Therefore, each real system scenario, has a corresponding simulation experiment created. The following section describes the simulation experiment models for Company X which are then used to produce simulation output results for overall model validation.

Automated Simulation Model

As described earlier, modern cloud data centres are expected to have a lot of interacting components which when modelled translate to large complex simulation models. In such circumstances manual model building is not a viable option. The presented toolkit described in this thesis is centred on automated model building working in unison with an integrated data collection framework. This integration allows for the simulation toolkit to query the data collection framework for historical data and to use this data to build representative real-world models which can be used for simulation experimentation. During the model build process models capturing physical and logical components of the cloud data centre system are used. Model design details previously presented in Chapter 3.6 are then followed by implementation described in Chapter 5.4.2. This section describes simulation model content that was built using the data collection framework integration method. Details on the data collection framework integration was presented earlier as part of the methodology (Chapter 4), software architecture design (Chapter 5.3) and software implementation (Chapter 5.4) discussions.

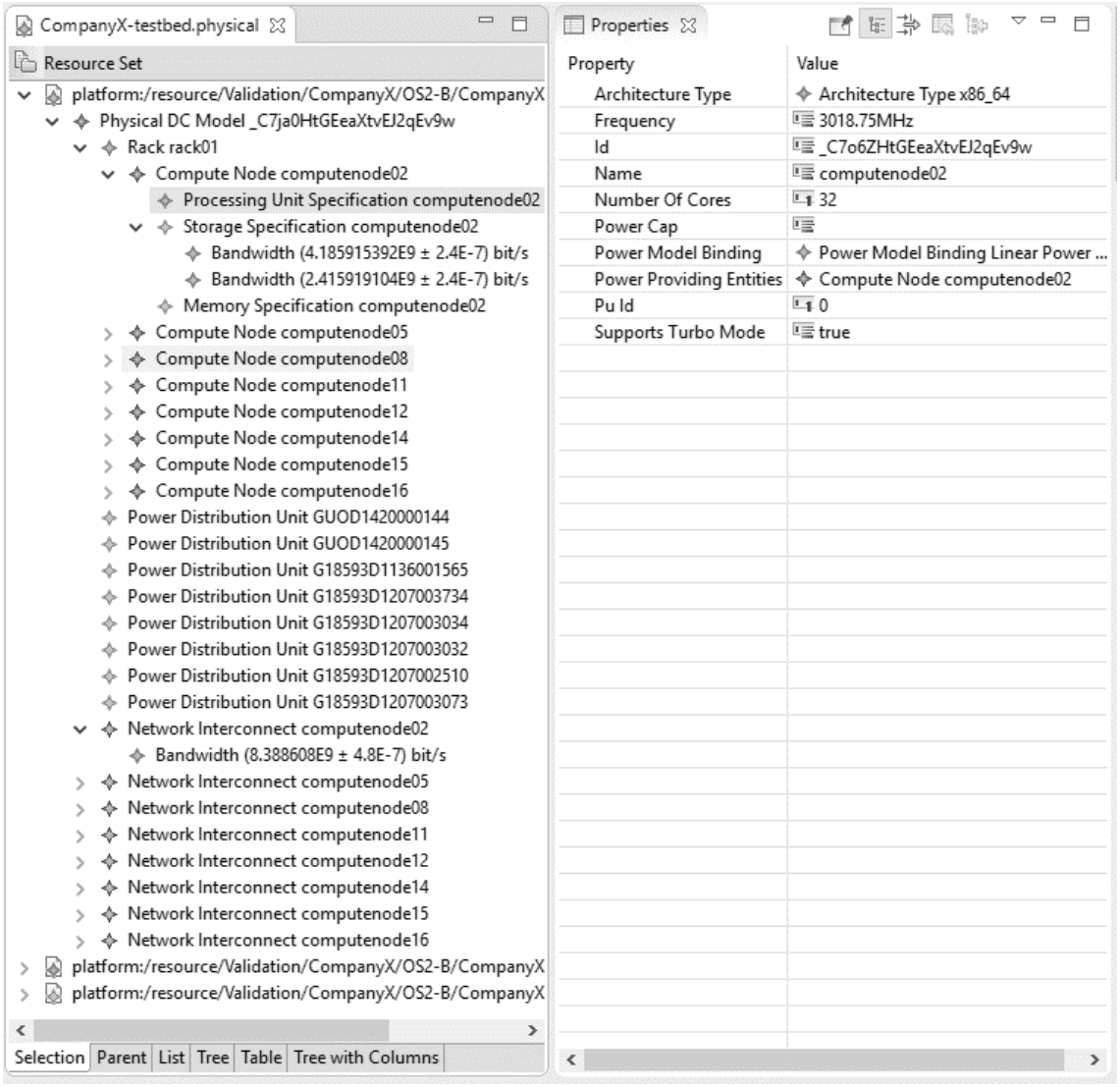


Figure: 6-4: Extracted physical data centre model of Company X

The Physical Data Centre Model (PDCM) was built to represent hardware components inside the data hall in the data centre of Company X. Figure: 6-4 presents a screenshot of the user interface from the model, which includes racks which host compute nodes, power distribution units (PDU) and network interconnects. Each compute node includes processing unit specifications, storage specifications and memory specifications. PDUs contain peak power information and ids of connected nodes. Network interconnects carry information of available bandwidth for each node.

The acquired model data was verified against the hardware inventory list shown in Table 6-1 to ensure the right information was extracted from the data monitoring

system and that the data matches the standards for hardware specifications one would expect to find. However some small deviations are expected from the factory data, for example model data shown in Figure: 6-4 lists the CPU frequency to equal “3018.75 MHz” as opposed to “3000 MHz” listed in the specifications (shown in Table 6-1). Such value fluctuations can be attributed to the dependency on the operational conditions, for example device temperature or motherboard voltage control.

Figure 6-5: Extracted logical data centre model of Company X

The Logical Data Centre Model (LDCM) is designed to capture the cloud data centre virtualisation layer. This includes hypervisor deployments, virtual disk volumes, VM images, application models and VM flavours. The automatically composed model is shown in Figure 6-5. It includes hypervisor elements mapped to every physical compute

node where VMs can be deployed. Every compute node storage is exposed through a virtual interface to be available to be mounted over the network from VMs running on different nodes. VM image instances represent available configured operating system environments for VMs and VM flavours corresponding to the VM resource configurations presented in Table 6-3.

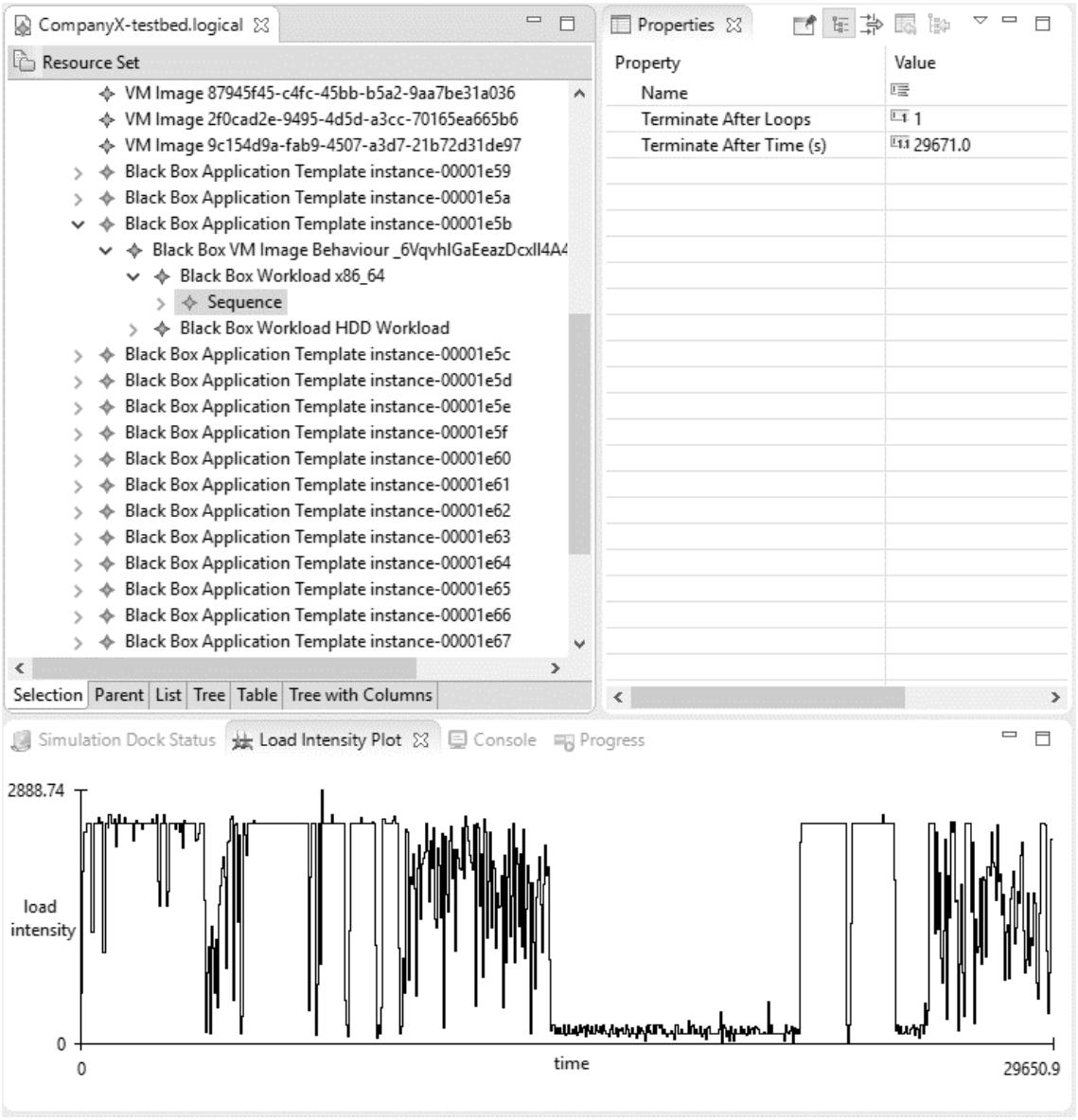


Figure 6-6: Extracted application model of Company X

The captured LDCM also contains “Black Box Application Template” instances where application workload sequence models are stored. The Black Box application template is used because the infrastructure provider is usually not aware of the

application type running inside a VM, having access to just resource demand data per VM. Due to the Molpro application nature, which is running large molecular simulation calculations, it requires a large amount of resources and subsequently long job processing times. The captured CPU sequence shown in Figure 6-6 can be logically divided into 4 different calculation phases executed sequentially. Each phase requires a certain amount of CPU processing power for the duration of an application run. The mentioned sequence of resource demand is started as soon as the VM is deployed to the cloud and terminates after the VM is shutdown or once the sequence time is finished.

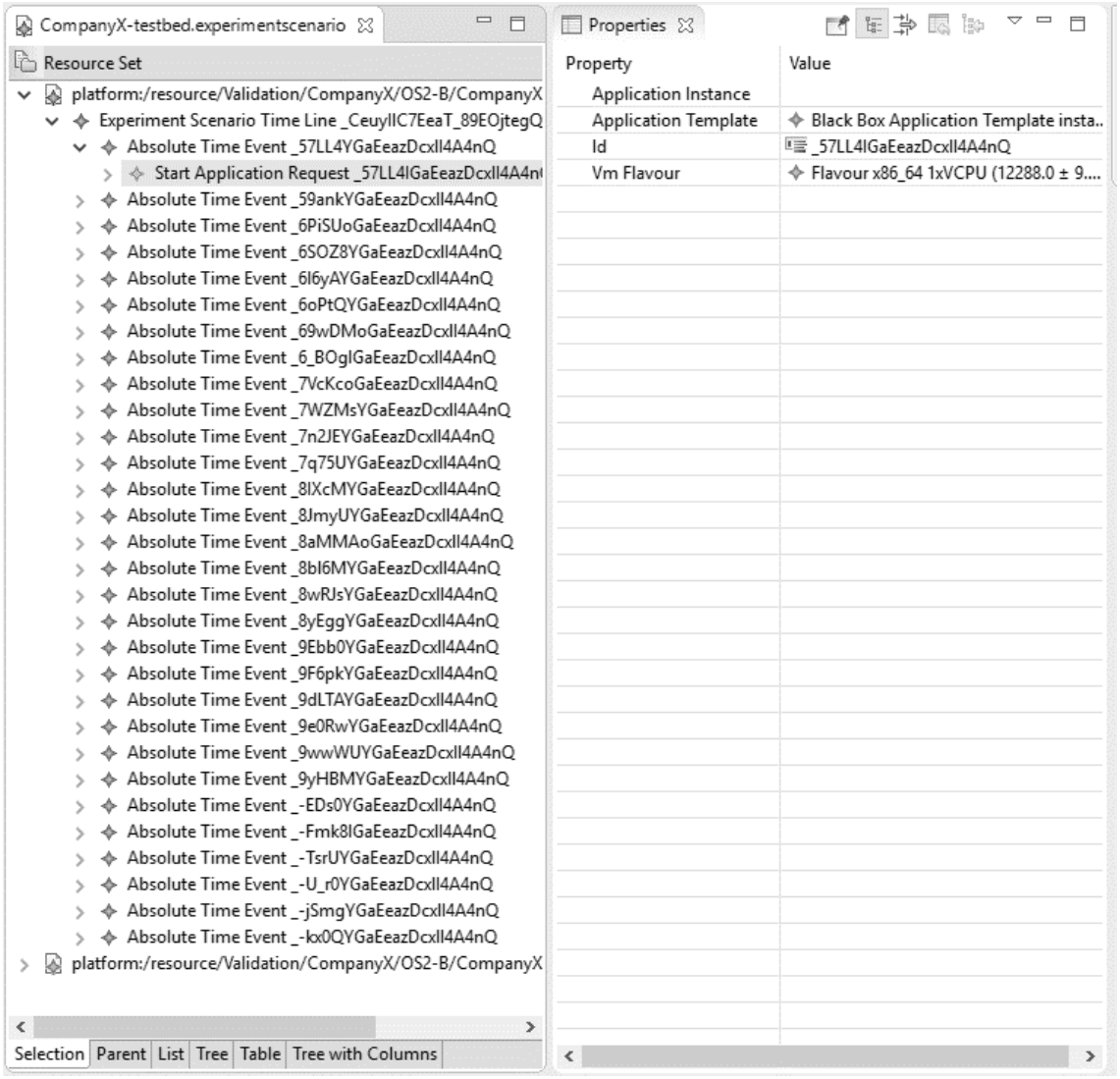


Figure 6-7: Extracted experiment scenario model of Company X

The purpose of the experiment scenario model is to capture the lifecycle of the VMs arriving and departing the cloud data centre shown in Figure 6-2. The experiment

scenario is extracted from the cloud monitoring framework database by specifying its start and end time. The retrieved model will contain a list of the time events defining when the VM is deployed and the time event when the VM should be terminated within the simulation experiment. As shown in Figure 6-7 the VM start request contains a link to the application template and VM flavour to use. The application template (shown in Figure 6-9) is assigned to the deployed VM executing the resource demand sequence defined in the application model. Via the link to VM flavour (shown in Figure 6-5) the starting request identifies the amount of virtual resources needed for the VM to be provisioned. The VM termination request just has the ID of the start request so that the associated running VM can be identified and shut down.

Experiment	VMs admitted	Admission duration (s)	Monitored time (s)
OS-A	25	6015	6360
OS-B	15	30840	30840
OS-C	18	31440	31560
OS-D	24	4556	4739
OS-E	24	3892	3900
OS-F	17	21387	21540

Table 6-5: Experiments workload summary

Table 6-5 compliments the VM admission overview presented in the Figure 6-2 and Figure 6-7 by showing the exact figures of the total number of recorded VM admissions, the duration of VM admission sequences and the monitored time covering each of the experiment. The presented figures confirm experiment equivalences making experiment OS-A the most workload intensive with 25 VM admissions and experiment OS-C the longest duration with a duration of 8.8h (31,560s). The time period for which the data was collected is shown as monitored time and is equal to or longer when compared to the actual VM lifecycle duration. Some slight extensions to the monitored time provides a time buffer allowing for any time synchronisation inconsistencies. The difference in the admission patterns between the experiments does not impact the validation exercise since the simulation experiments are modelled to mirror each real system test scenario.

The following section presents the comparison between the real system test scenario results and corresponding simulated experiment results, thus validating the proposed simulation-based framework output against real system output.

Simulation result validation

This section compares the real system scenario results side by side with the results of the same scenarios replicated in simulation, for the purpose of validation. All six experiments were compared side by side in the same manner using visual comparison and statistical result analysis. Experiment OS-B is used to illustrate the validation process in detail here within the thesis. While the step by step for each of the other five scenarios is not presented in detail in this chapter, each has undergone the same process and the overall results are presented in summary in Table 6-7.

Statistical experiment validation was performed by comparing real system power measurements collected during test scenarios with the simulation experiment results. Both data consisted of time series information, sharing the same time axis. Each real system measurement and simulated counterpart values were treated as a corresponding data sample pair. For experiment OS-B a total of 21,587 data pairs were observed and analysed using Student's two tail paired T-test (STUDENT, 1908). Additionally, the difference between the measured and simulated data was statistically analysed in order to gain insights about data distribution and correlation metrics. From the histogram shown in Figure 6-8 the distribution of difference between measured and simulated results shows data normality with most results gravitating towards the centre. The highest two ranges of the results differences are located on both sides from zero, between -14 and 1 W and between 1 and 16 W. Amalgamation of data points close to zero provides further evidence that the difference between the simulated and measured data are also converging close to zero.

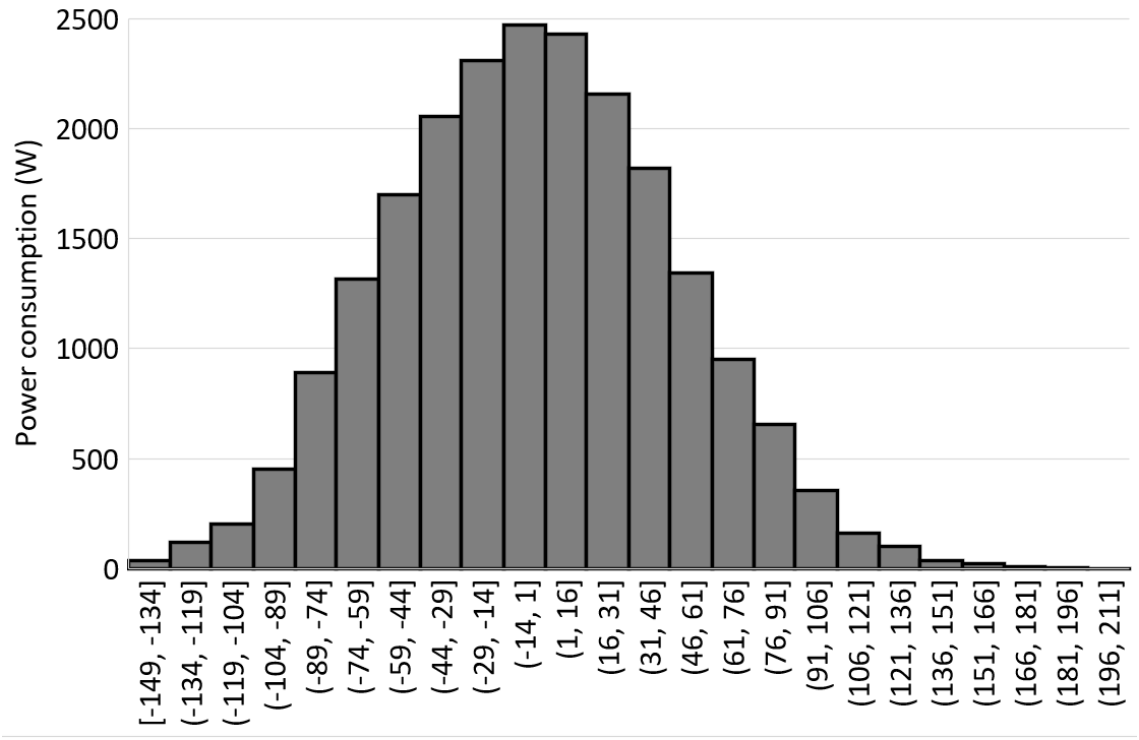


Figure 6-8: Difference spread between simulated and measured data (OS-B)

Statistical analysis results presented in Table 6-6, show that the average (mean), difference between measured and simulated data is 2.73 W and analysis shows 95% confidence of the mean being distributed between -3.4 to -2.06 based on difference analysis. These low values show the close proximity of the average data set values for the whole population, meaning that future experimentation results should also remain within the same rage. This is also confirmed by a high Pearson correlation score of 0.87, meaning that the measured and simulated data pairs follow similar patterns. However, the T-test shows significant differences between the two data sets with a low probability of $1.75*10^{-15}$. According to Kelton and Law (2000) the low probability of simulation results being almost the same as measured results, or in other words, rejection of null hypothesis, is an expected outcome of data analysis as high precision is rarely achieved in simulated systems.

Attribute	Value
Observations No.	21587
Mean	2.73
Confidence level (95.0%) (margin of error)	0.67
Standard deviation	50.34
Pearson correlation	0.87
Lower bound	-3.40
Upper bound	-2.06
P (T-test (two-tail))	$1.75*10^{-15}$

Table 6-6: Statistical difference analysis between measured and simulated data (OS-B)

To compliment this statistical data analysis, a visual data comparison is also performed to compare the two datasets. Figure 6-9 shows both predicted and simulated power consumption measurements plotted on the same graph within the duration of the experiment. It was generated using a moving average of 250 records in order to smooth out the curves. From the figure it can be seen that the power demand peaks at approximately 830 W for all of the monitored compute nodes in the beginning of the experiment and then reduces to 630 W and then towards the nominal 480 W when all of the running VMs are terminated and the workload drops. It is seen that the simulated power curve closely follows the measured power curve, reacting in the same way to the induced workload by drawing more power at the beginning of the experiment and reducing the draw once the workload has been gradually reduced.

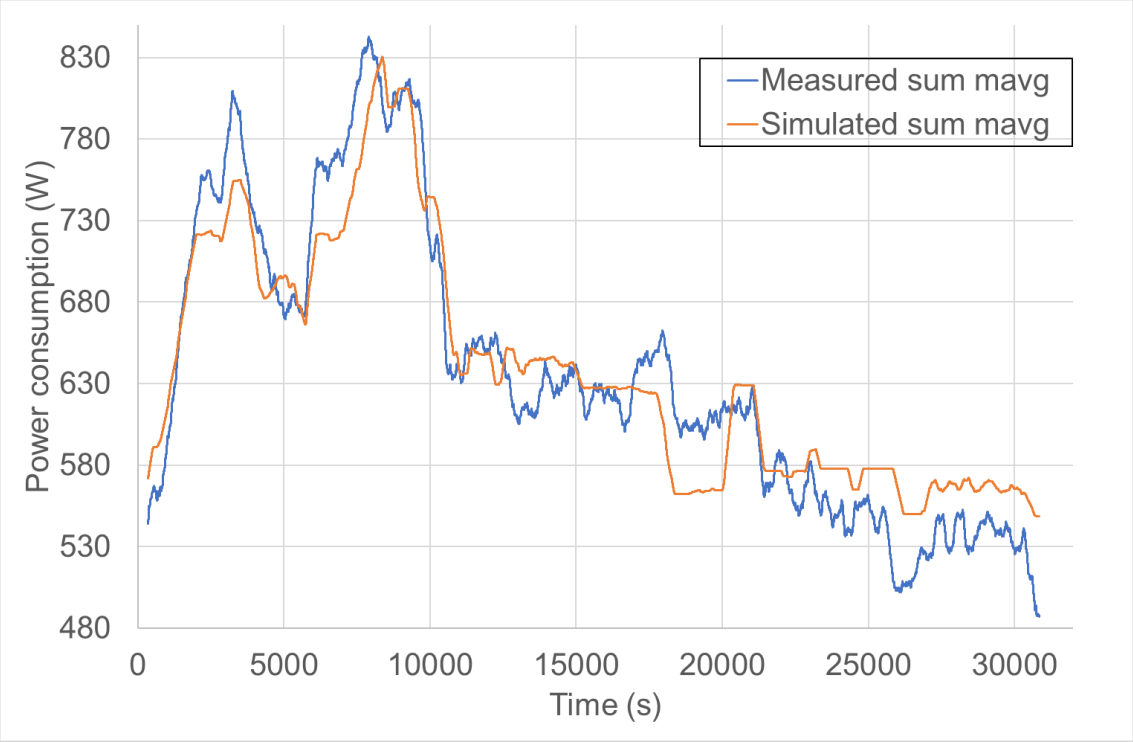


Figure 6-9: Measured and simulated power consumption for OS-B experiment

To compare the deviation between the simulated and the measured data Figure 6-10 shows the time series difference between measurements taken at the corresponding timestamps and converting to percentages. Throughout the experiment the average error rate is 5.6%. This falls within the set 10% average error limit and the results of the model are deemed valid. However, it is important to note that an increasing error rate can also be seen towards the end of the experiment with a spike error of approximately 12%. The spike in the results deviation can be linked to the excluded two nodes due to the absence of power measurement equipment and use of a simplistic linear power estimation model.

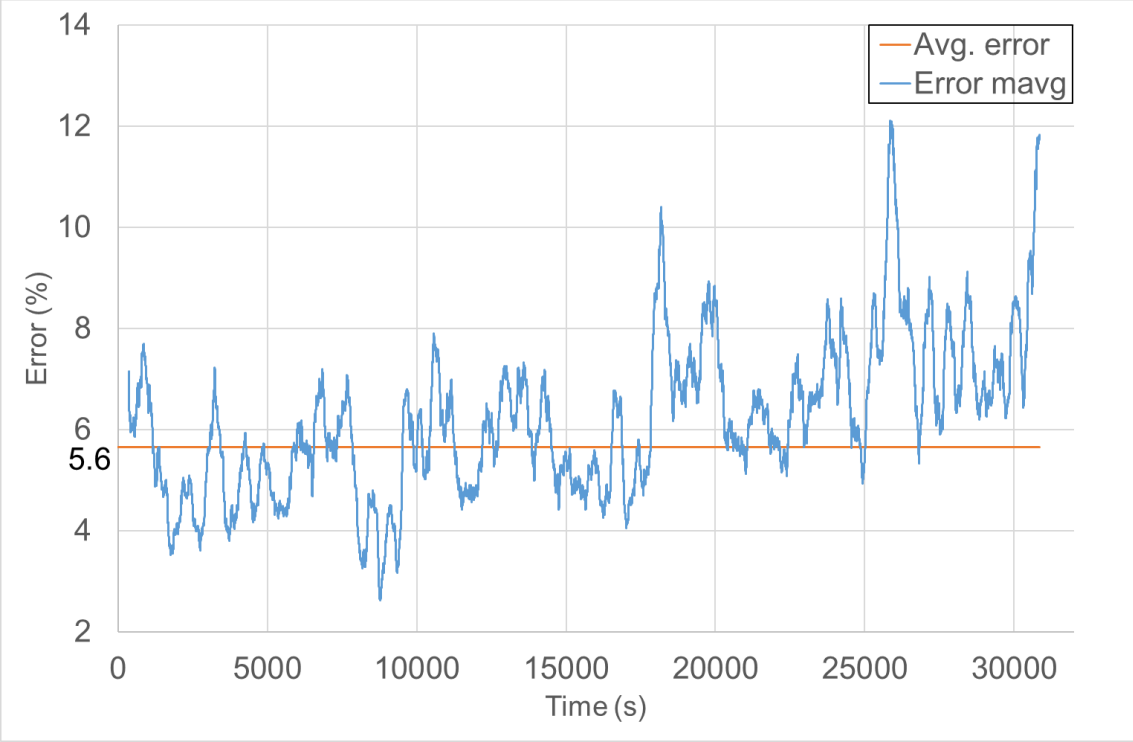


Figure 6-10: Simulation energy cost error for OS-B experiment

Full statistical comparison results for all validation experiments are presented in Table 6-7. The null hypothesis was rejected for all experiments meaning that the measured system values are unlikely to be exactly the same as the simulation results, synchronised at the same points in the experiment time. This is as expected when comparing real world and simulated results. A similar pattern of results as observed in the OS-B results was confirmed across each of the other five experiments. Linear relationships between measured and simulated variables expressed in Pearson correlation shows ascending strong correlation coefficients for all of the experiments with the exception of OS-A. As a reminder, OS-A was executed to develop a baseline system performance estimation without introducing any optimisation decisions from the resource optimisation framework - CactoOpt. During the OS-A experiment the default OpenStack VM admission policy was used which is controlled by the default policy which strives to admit workload using a round robin approach. As no optimisation policies were used this means that the VMs were not migrated, remaining at all times on the node on which they have been initially placed during the experiment run. Since OS-A is an experiment which was executed with a set of mismatched resource management policies for the real and simulated environments it demonstrates an anticipated behaviour by having a poorer

correlation in output results. The corresponding experiment correlation levels for all others range from 0.75 in OS-D to 0.93 in OS-E which means that simulation generated results are reflecting resource management decisions made by the optimisation framework. However, experiment OS-E while having the highest correlation rate, also has the highest average difference of 159.36 W which alludes to a discrepancy in the model data for that experiment.

To put the validation results into perspective the average error percentage column shows the ratio of inaccuracy of simulation results to the measured readings. From the results Table 6-7, it can be concluded that the average simulation error rate for OS-B to OS-F is less than or equal to 6.9 for all five experiments. Each of these five experiments were run with a matching optimisation policy being used in both the testbed and simulated scenarios. These all fall below the set 10% average error limit and the results are all deemed valid. The mismatched optimisation policy experiment OS-A shows an average error rate of 20.5%. Experiment OS-A highlights the negative influence of a lack of detailed resource management policies on system outputs and the value of developing simulation-based models which explicitly includes such policies.

Name	Observations	Mean	Standard deviation	Pearson correlation	P (T-test)	Average error (%)
OS-A	*4,417*	*85.48*	*171.22*	*~0*	*$9.72*10^{-216}$*	*20.5*
OS-B	21,587	-2.73	50.34	0.87	$1.75*10^{-15}$	5.6
OS-C	22,077	-55.76	68.1	0.81	0	5.6
OS-D	3,316	-38.18	104.57	0.75	$2.97*10^{-92}$	5.7
OS-E	2,729	159.36	72.25	0.93	0	6.9
OS-F	15,076	-25.69	50.5	0.78	0	4.0

Table 6-7: Measured and simulated energy comparison

Following validation steps outlined in Figure 6-1 the real system scenarios were replicated in simulation experiments and then real system measurements were compared to simulation outputs to see the accuracy of the proposed simulation-based decision support framework. Validation steps were demonstrated by showing a walk-through of OS-B experiment. These steps included a detailed statistical comparison

(Table 6-6), visual power data analysis (Figure 6-9) and visual error rate analysis (Figure 6-10). Finally, a summary of all experiments is presented in Table 6-7. The presented analysis demonstrates the practical application of the cloud simulation framework with variable success rates where the average difference between simulated and measured results fluctuates from -2.73 to 159.36. In conclusion, based on achievement of the targeted average error rate for OS-B to OS-F the simulation model has been deemed valid.

6.2.2 Company Y

In recognition that cloud data centres host a variety of different application types to accomplish different tasks for a variety of different users, this section introduces a second real case scenario, from Company Y. As was already seen in Section 6.2.1, simulation experiments were carried out to mirror real system scenarios comprised of Molpro application deployments which characterize a HPC cloud application. However, a Molpro application workload represents only one type of service running in the cloud. To expand and diversify the validated model pool of cloud applications, further analysis is now presented for the Company Y case study which includes a broader range of cloud applications.

The goal of this additional modelling exercise is to demonstrate the proposed methodology and frameworks usability in a different use case setting. This case study presents the use of the developed automated modelling framework in a different data centre setting, demonstrating the modelling frameworks ability to be applied to different cloud applications seamlessly. The tested cloud application contains a web frontend and data base backend components imitating website setup which is a popular application setup in public clouds.

Validation exercises are presented in this section following the same processes outlined in Figure 6-1:

1. Test scenario execution in the real cloud data centre (Company Y)
2. Scenario measurement data recorded and used to produce simulation models of the cloud data centre infrastructure and workload
3. System models are used to run simulation experiments

4. Simulation results compared with measured real system data for validation purposes

The test scenario described in this section was conducted in the data centre of Company Y using the CACTOS business analytics use case approach (see Chapter 3.2). As with the previous example, scenarios were setup in an isolated controlled environment to avoid any external interference for the duration of the testing phase. Deployed VMs were pre-provisioned with a web-application and immediately after VM deployment user workload was emulated for each VM, creating demand for cloud resources. Data collected during the real system scenario was then used to automatically create and later to validate simulation models via comparison with the simulation output.

Scenario design and execution

The scenario is set in a physical testbed containing a subset of nodes representing the cloud data centre infrastructure. The aim is to recreate a real case scenario where VMs are being admitted to the data centre and workload is being induced on each VM, representing a user workload as in day-to-day operations within an actual large-scale data centre. The test scenario setup can be divided into three parts: 1) infrastructure configuration, 2) admitted VMs and 3) the workload running within these VMs.

The test scenario is setup to represent a public cloud provider set-up where an internal structure and distribution of the web application is unknown to the cloud provider. The cloud infrastructure provider only can monitor each VMs resource demand patterns but is unaware of any application details such as application type, number of requests or the request types the application serves. Such application representation is generally termed a Black Box model and it includes only resource demand (utilisation) data for each VM. The resource utilisation collected from system monitoring data is then translated into resource demand bindings within the simulation models.

ID	CPU			RAM	Disk
	Make	***Cores***	***Frequency***		
10.157.128.30	AMD Opteron 6366 HE	16	1800 MHz	128 GB DDR3	4 x 1TB HDD

10.157.128.31	AMD Opteron 6366 HE	16	1800 MHz	128 GB DDR3	4 x 1TB HDD
10.157.128.32	AMD Opteron 6366 HE	16	1800 MHz	128 GB DDR3	4 x 1TB HDD

Table 6-8: Allocated experiment testbed infrastructure for web services at Company Y data centre

The infrastructure available for the experiment (shown in Table 6-8) consists of three identical compute nodes running an AMD Opteron 6366 HE 16 cores @1800MHz make CPU with 128GB of memory and 4 1TB HDDs as storage. Each node has a hypervisor deployed, ready for VM deployment. These nodes are part of a larger cluster that are interconnected by a switch and managed by a separate dedicated management node in the cluster. The management node is running a proprietary Flexiant Cloud Orchestrator (FCO) implementation for managing cloud computing platform integrated with CactoScale data collection framework and CactoOpt optimisation framework. Since the management node resources are not part of the public cloud infrastructure offered to customers, management node is excluded from the infrastructure model.

During the scenario execution two optimisation policies were used: one for VM placement and another one for VM consolidation. The VM placement policy is responsible for determining hardware resource allocations for the arriving new VMs. When the user defines VM configuration and requests VM creation, a placement algorithm will make a decision based on available resources as to where to place the VM. The VM consolidation policy is deployed on a periodic basis reviewing the resource demand patterns to determine if VM placement could be better adjusted when compared to the initial placement decisions. If an algorithm suggests that VMs should be rearranged the migration command is issued to the management system.

Company Y's testbed is managed by the FCO system which was integrated with the CactoOpt optimisation framework. However, FCO is a proprietary cloud management solution with limited outside access possibilities. Hence, due to the FCO design subtleties VM placement decisions were not managed by CactoOpt despite its integration, but still remained within the FCO platform (see Table 6-9). This mean that when new VM request arrived from a cloud user the FCO used its own proprietary algorithm to determine the resource allocation for the VM. This is typical in many real-world scenarios where proprietary solutions are being used. The consolidation policy

was fully operated by the CactoOpt framework being in control on which policy to use for the scenario. The load balancing algorithm which spreads VMs equally among compute nodes was selected to be used with the test scenario to avoid system resource bottle necks as much as possible.

Scenario	Placement Policy	Migration Policy
FCO-A	FCO Proprietary	Load balancing

Table 6-9: Company Y experiment list with optimisation policies

To emulate user demand for public cloud services an automatic script is developed to provision 57 new VMs within a duration of 3.5 hours. To represent realistic scenarios the VM configurations are divided into three categories Small, Medium and Large. Small VMs require 1 CPU core, 512MB memory and 1 GB storage (shown in Table 6-10). Every VM is pre-provisioned to run a certain popular web service such as web frontend, SQL and an intense calculation backend service. Once a VM is deployed, the service is then put under a load that represents user demand, for the web frontend JPetstore website implementation is used and MySQL database engine provisioned to represent SQL type storage services. Custom scripts are executed to create user requests for accessing web pages and reading and writing database entries. To mimic the calculations of an intensive custom proprietary cloud application a synthetic workload is induced using a Stress framework on the remaining VMs.

VM Sizes	CPU Cores	RAM (GB)	HDD (GB)	SQL Server VMs	Web Sever VMs	Unknown (Stress) VMs
Small	1	0.5	1	15	15	15
Medium	4	4	2	10	10	10
Large	8	8	4	5	5	5

Table 6-10: Admitted VM configurations and quantities

To imitate a steady increase of workload demand in the cloud system VMs, were gradually admitted to the testbed. The admission delay was programmed in the script to be evenly timed between each VM deployment, however in the real system there were some delays (as seen in Figure 6-11) due to the FCO internal processing intricacies. After VMs were admitted, the system was left running for approximately one hour before

terminating all the VMs at once. The test scenario was designed to allow the resource management framework to handle the growing VM admission rate and then give an opportunity for further optimisation via VM migration during an hour of system steady state.

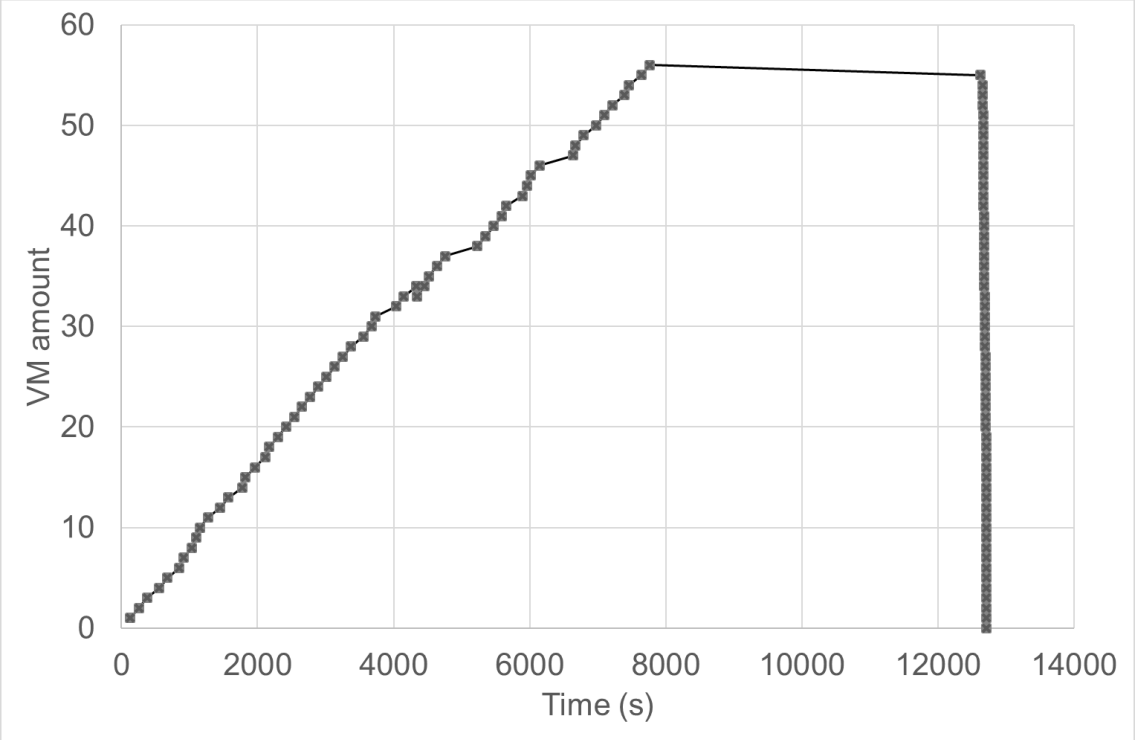


Figure 6-11: Company Y testbed VM admission experiment scenario

To summarise, resource provisioning demand, Table 6-11 presents averages and peak resource assignments during the FCO-A scenario execution. Because there were only 3 compute nodes available within the testbed to host VMs the provisioning for CPU cores goes to 118.8% on average and 255% at the peak of VM saturation, with both values above 100% of system capacity. These figures mean that the deployed VMs had a higher demand for CPU cores than the system can provide which resulted in VMs sharing CPU cores between each other. When it comes to memory the average demand over the experiment was 22.8% with a peak of 54.6%, which indicates sufficient resources to serve demand. The storage was in demand the least with an average of 0.2 % and 0.4% peak respectively comparing to CPU cores and Memory. These results indicate a sharp shortage of CPU cores most likely hindering the performance of the CPU bound workloads. Such situations should be avoided in the production system by expanding hardware capacity.

Name	CPU provisioning (%)		Memory provisioning (%)		Storage provisioning (%)	
	Avg.	Peak	Avg.	Peak	Avg.	Peak
FCO-A	118.8	255	27.8	54.6	0.2	0.4

Table 6-11: Experiment resource provisioning summary

During the test scenario two compute nodes 32 and 30 had a power meter device connected in between the Power Distribution Unit (PDU) and the physical node. This was used to measure the power consumed by each node over the duration of the experiment. Figure 6-12 presents the measured time series graph of power consumption for both nodes. As can be seen, the power demand increases as more VMs are admitted to the system and decreases again once VMs become idle when the generated workload running inside each of these VMs is being completed. It is worth noting that the measured power consumption reflects only the power draw of the compute nodes without taking into account the cooling effort needed for the server room.

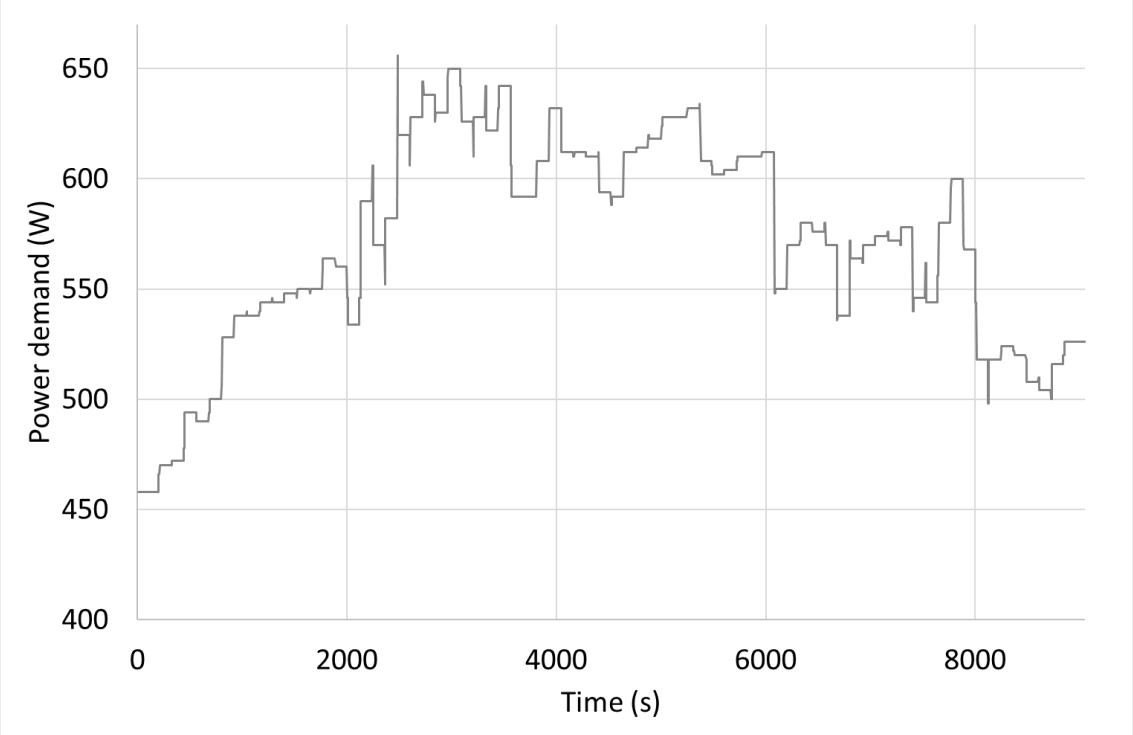


Figure 6-12: Company Y testbed measured power consumption

As shown in Table 6-12, the duration of the scenario was set for 12717 seconds, with system performance being monitored for 9045 seconds which is the cut off point for the load induced by the workload.

Name	VMs admitted	Admission duration (s)	Monitored time (s)
FCO-A	57	12717	9045

Table 6-12: Experiment workload summary

During the test scenario 57 VMs were gradually admitted to the test bed and a synthetic load was induced on the VMs emulating different resource demand patterns. Resource management was shared between the proprietary FCO system and the CACTOS optimisation framework - CactoOpt. Decisions on where to place arriving VMs were handled by the FCO, but VM migration options were executed by CactoOpt. This power sharing agreement between optimisation systems limited the degree of control over resource distribution. After the scenario was executed the measured system data was collected to be used further in simulation experiments.

Automated Simulation Model

The model build process for Company Y follows the same steps as previously described for the build process for Company X. Experiments were executed in the relevant testbed and models were compiled automatically based on the data obtained from the data collection framework.

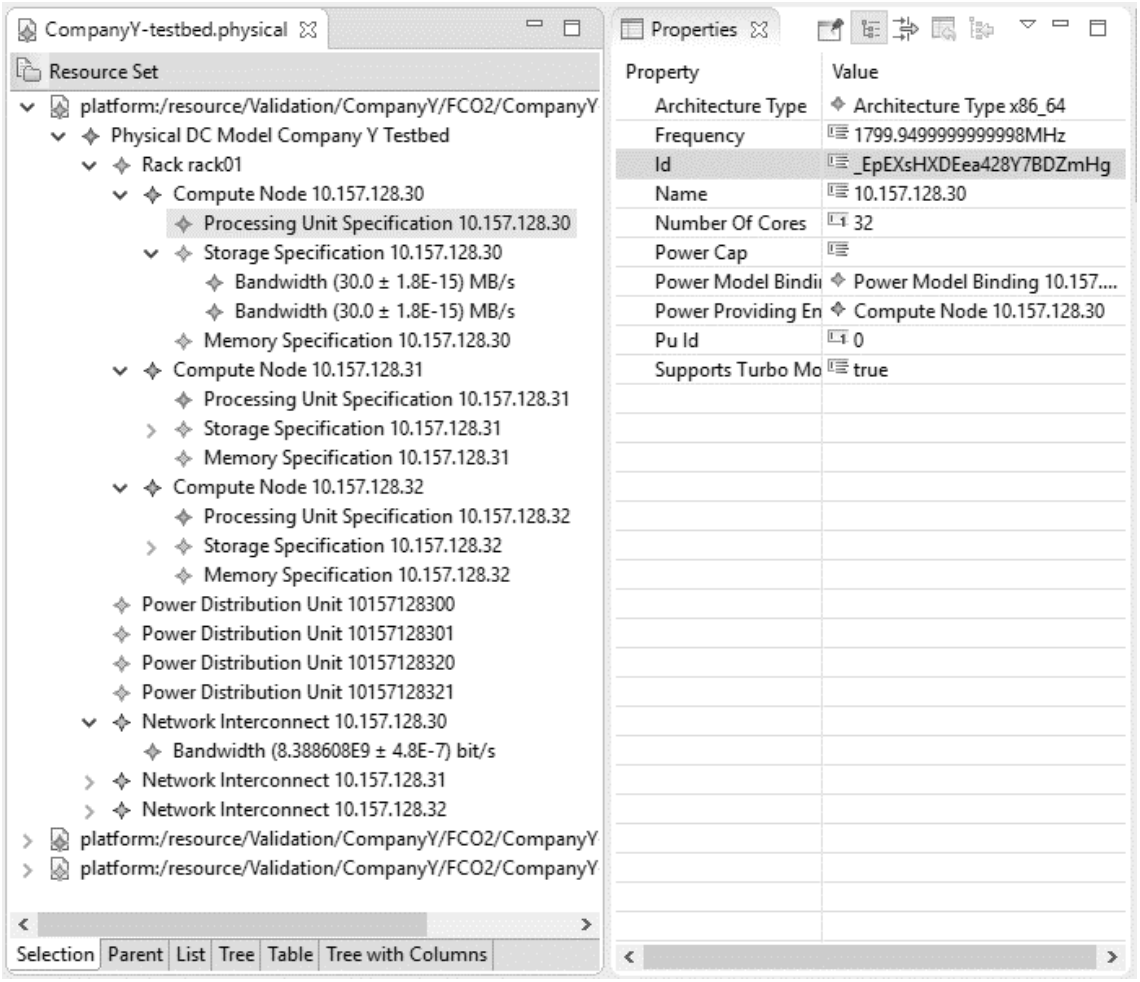


Figure 6-13: Extracted physical data centre model of Company Y

The collected data centre infrastructure model shown in Figure 6-13 consists of a tree element topology. The top node in the topology represents the data centre element named “Company Y Testbed” which consists of one rack element named “rack01”. The rack itself has three nodes named “10.157.128.30”, “10.157.128.31” and “10.157.128.32”. Hardware specification seen in the model was obtained from data collection framework and verified against manufacturers specifications shown in Table 6-8. Apart from the node resources information model contains PDU unit and network interconnect information for each node. Each node is connected to two PDU units for redundancy purposes, but since node “10.157.128.31” does not have a power meter attached there is no power data supplied to the model, hence only four PDUs are shown.

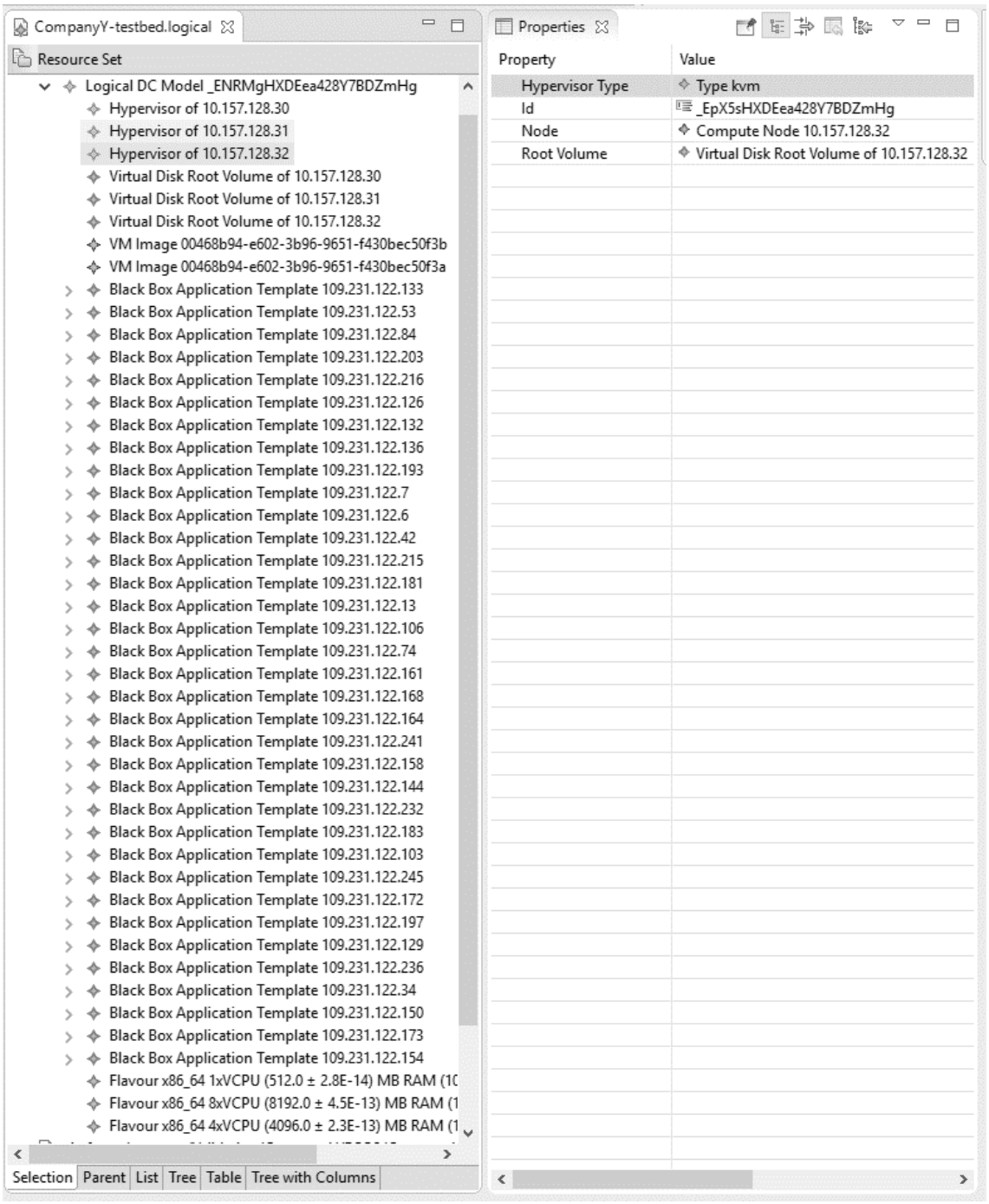


Figure 6-14: Extracted logical data centre model of Company Y

The logical model captures three hypervisor entities each deployed on three compute nodes respectively (Figure 6-14). Three virtual storage devices are listed in the model representing virtualised storage physically located at each node. Two types of VM

images and pre-configured VM flavours corresponding with VM flavours shown in Table 6-10.

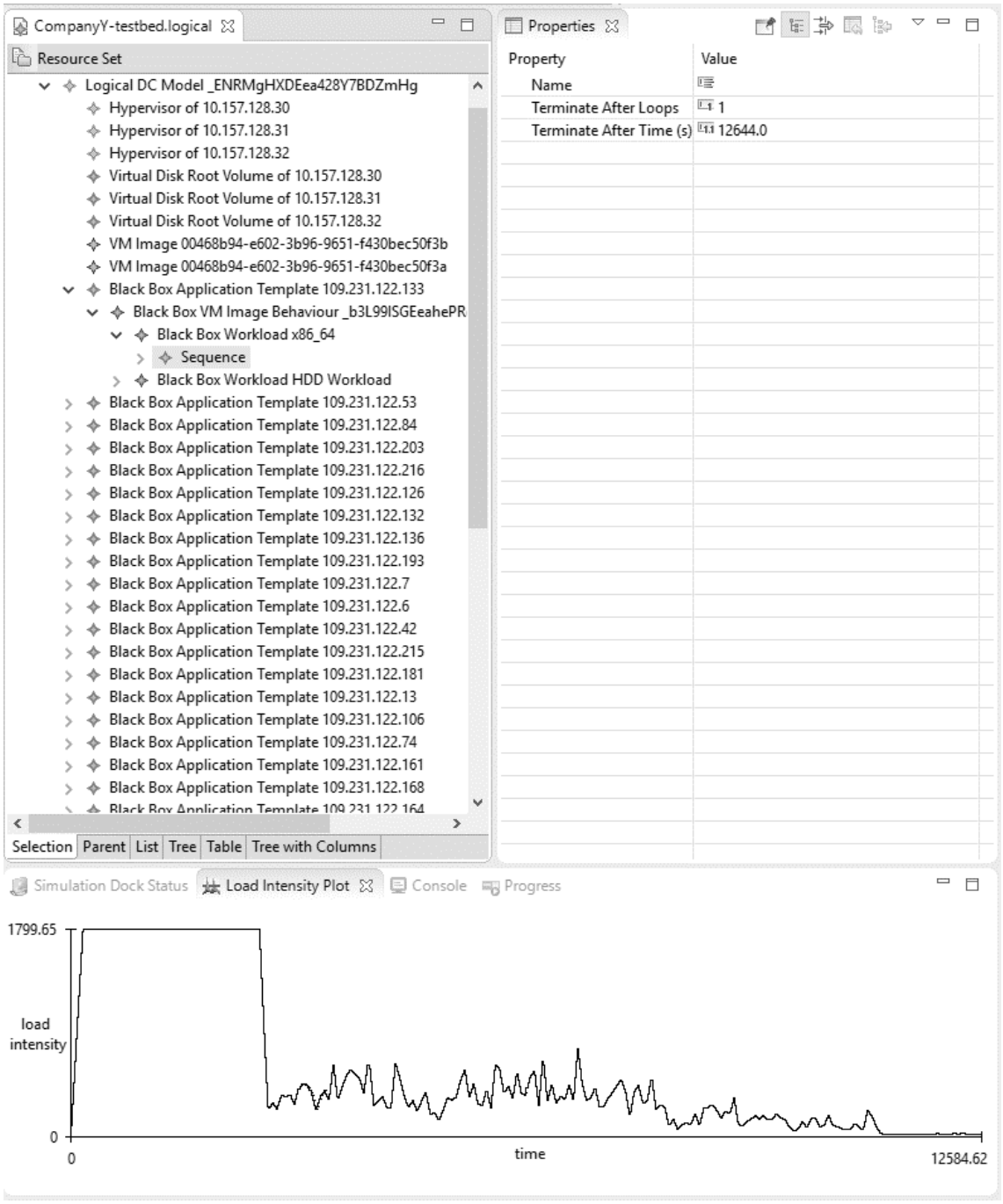


Figure 6-15: Extracted logical data centre model of Company Y

The extracted application models reflect the time series notation of resource demand captured per monitored VM within the duration of the whole experiment. Figure

6-15 is an example of a single VM resource demand. It presents the CPU resource demand over time expressed in Millions of Instructions (MI) and the storage resource demand expressed in Input Output (IO). Such metrics are captured for every deployed VM in the cloud data centre by the CactoScale data collection framework. Within the simulation model, CPU and Storage hardware components have a processing speed defined in Million Instruction Per Second and Input Output (operations) Per Second (IOPS). By combining the application resource demand and hardware processing capacity expressed in such a way, the processing time can be calculated. For example, if the application requires 1,000 MI CPU power to process a user request and CPU core assigned to VM has 2,000 MIPS processing speed then the request will be processed in 0.5 sec. Each of the “Black Box Application Template” elements shown in Figure 6-15 contains a unique resource demand trace that was captured through the data collection framework monitoring capabilities during the execution of the real system experimentation.

Figure 6-16: Extracted experiment scenario data centre model of Company Y

The workload model is captured in terms of VM arrival and termination within the experiment duration. Once a VM has been deployed to the data centre the workload captured in the application model is being executed simulating resource demand of a particular application on the cloud infrastructure. Figure 6-11 shows VM starting times

within the duration of the experiment together with the total amount of active VMs in the system. The extracted data shown in Figure 6-16 is in line with the experiment setup of 3.5 hour VM admission cycle with an average 2.2 min delay between VM deployments up until maximum of total 57 VMs deployed to the system.

Simulation results validation

Once the test scenarios were executed on the real system, next, the simulation models were derived using the monitored experiment data. At this stage no change to the simulation models are applied and the exact scenario setup is transferred to the simulation experiment, executed using the CactoSim simulation framework. This exercise is performed in order to compare simulation results and actual monitored results side by side, thus, validating the simulation platform accuracy.

As described earlier in the validation of the Company X case, simulation result validation is achieved by comparing the difference between monitored real system data against simulated data via statistical and visual analysis. As shown in table Table 6-13 statistical validation was carried out using 1441 tuples of data, each tuple consisting of measured and simulated power values in Watts. On average each simulation data point was 40.42 W higher than measured data with a 95% confidence that mean will not deviate more than 2.7 as a margin of error. This means that based on the data analysis, there is a 95% confidence that the simulation is expected to produce power consumption estimations in the range from 38.35 W to 42.49 W higher than the actual system measurements.

Attribute	Value
Observations No.	1441
Mean	40.42
Confidence level (95.0%) (margin of error)	2.07
Standard deviation	40.19
Pearson correlation	0.71
Lower bound	38.35
Upper bound	42.49
P (T-test (two-tail))	$6.89*10^{-223}$

Table 6-13: Statistical difference analysis between measured and simulated data (FCO-A)

A standard deviation of 40.19 suggests that 68% of the data collected has a difference between the simulated and measured data clustered around ±40.19 W from the average of 40.42. Figure 6-17 presents a more detailed ranges of data comparisons. The conducted T-test shows significant difference in the data with a very low probability that the difference between the simulated and measured data is zero. As discussed earlier, this is in line with expectations. An overall strong correlation coefficient of 0.71 was observed between the simulated and measured data sets meaning that values from both sets are following similar patterns.

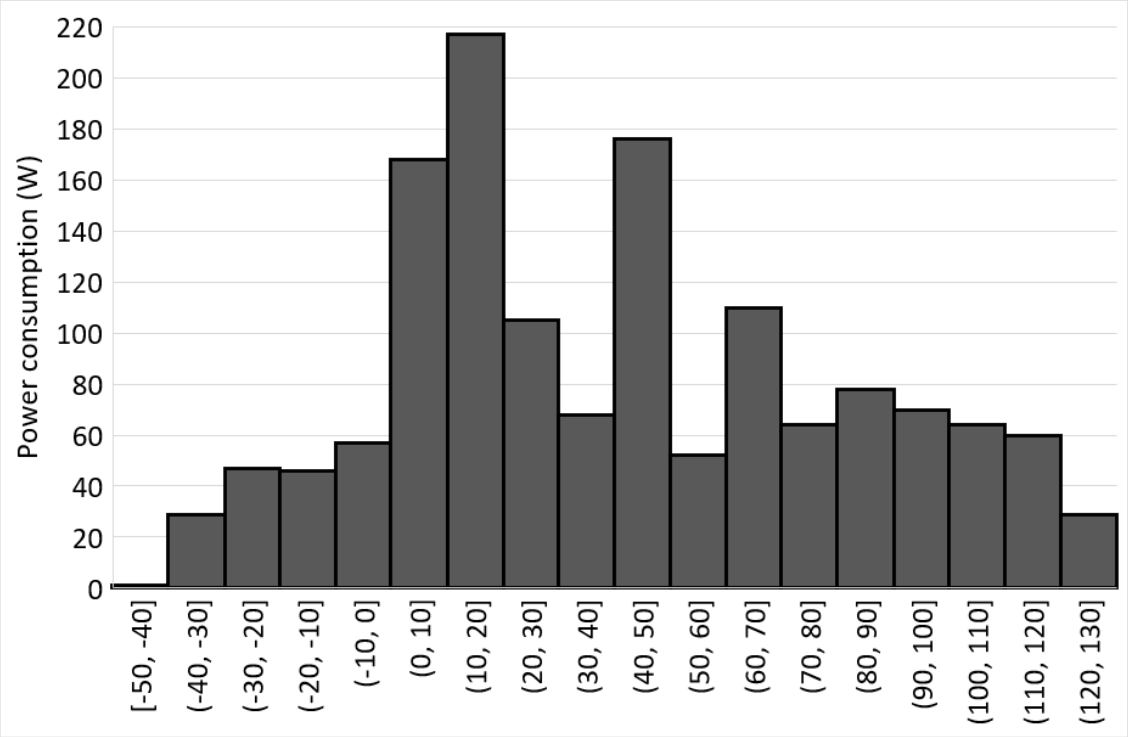


Figure 6-17: Difference spread between simulated and measured data (FCO-A)

A strong correlation can also be observed via the visual comparison shown in Figure 6-18 where both simulated power demand and measured power demands are plotted side by side. The compared power data follows a similar pattern of gradual increase when the VMs are admitted into the system and decrease once the user workload decreases.

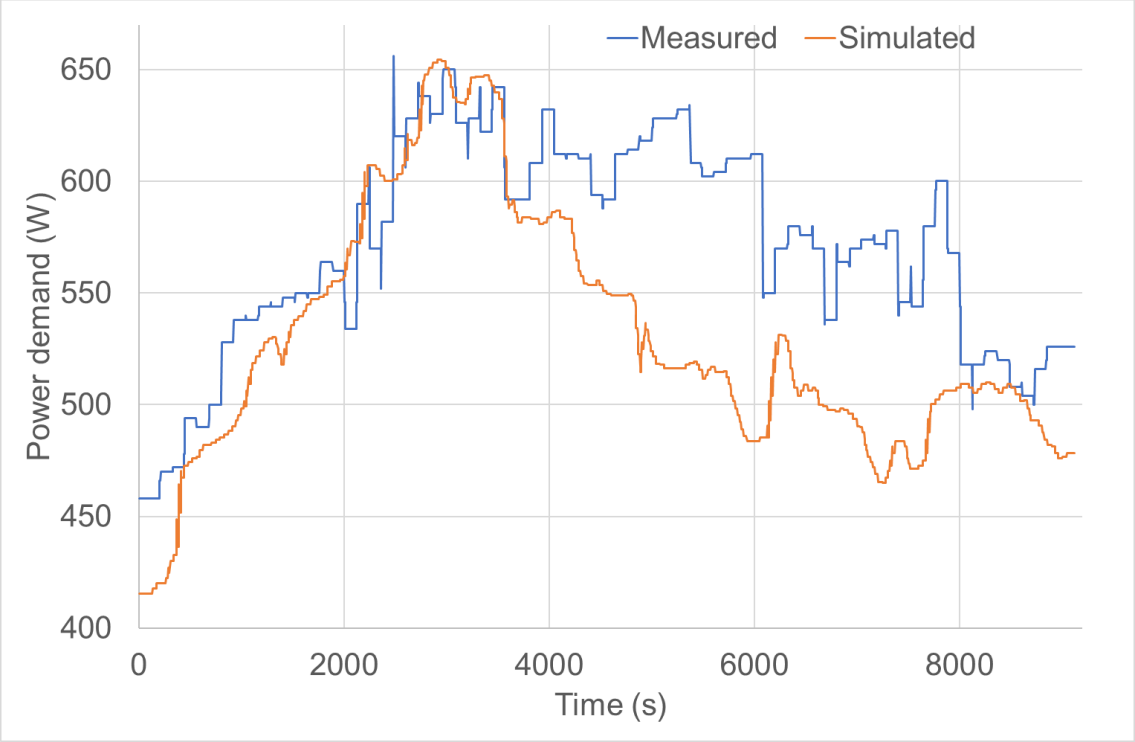


Figure 6-18: Simulation predicted vs measured Company Y testbed power

To facilitate further validation analysis the absolute error percentage between the measured and simulated energy consumption data was calculated and presented in Figure 6-19. The average error rate time series analysis serves as a useful way to visualise data deviation that was presented via the statistical analysis. If data trends shown in Figure 6-18 are compared to the error rate presented in Figure 6-19, both graphs follow a logical trend with a small error percentage rate in the beginning of the experiment and higher error rate towards the end of the experiment when measured data shows higher readings than simulated. As the average error rate (7.57%) for the experiment remains below the set threshold of 10% the results are deemed acceptable and the model considered valid. Again, as before it is important to note that there are some deviations from this average rate with the highest peak reaching a maximum error rate of just over 20%.

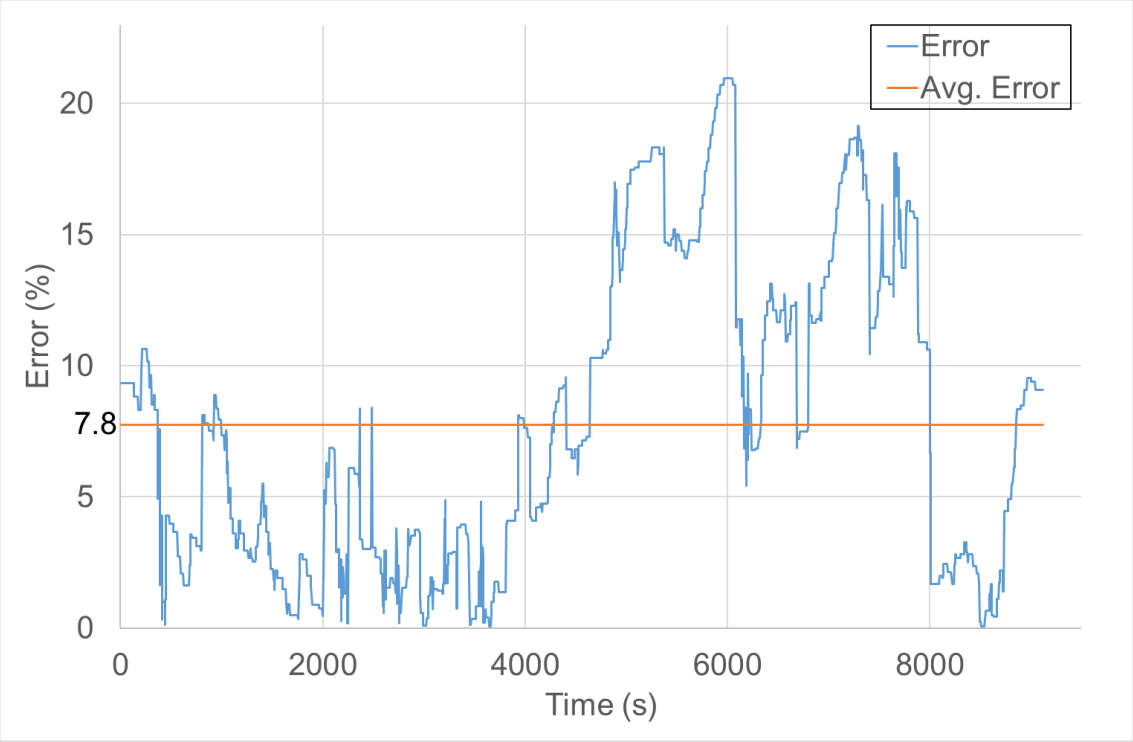


Figure 6-19: Simulation cost error for FCO-A experiment

Based on the analysis the presented validation shows that the simulation framework, CactoSim has been successfully integrated with the optimisation framework CactoOpt and is capable of executing provided placement and consolidation policies in a representative manner. This statement is backed up by the strong Pearson correlation score and visual comparison of both the simulated and measured power results. However, it can be also seen that in the second part of the experiment the deviation is increasing in error rate reaching a peak of just over 20%. Such behaviour is also echoed in the presented statistical data distribution analysis showing a large spread of results on both sides of zero in Figure 6-17. Due to the proprietary nature of the FCO placement policy within the simulated experiment the experiments were only using the placement policies supplied by CactoOpt. Even though a load balancing policy was used for the experiments, it would not guarantee the exact decisions for the VM placement when compared to the real system testbed experiments meaning that the real system setup was not fully replicated into the simulation environment. Hence, the discrepancy between the optimisation placement policies is reflected in the presented simulation results analysis. Again, while highlighting the usefulness of the methodology and framework,

one limitation is the presence of proprietary components being used within data centres. In such instances complete data is not available to the modelling framework.

Web application models from the business analytics use case were used by the simulation toolkit to estimate the cloud data centre power consumption for the FCO-A experiment with an average deviation error of 7.8%. These validated application models are added to the pool of cloud application models together with the Molpro application models obtained in the experimentation process shown earlier in Section 6.2. The application model pool is used to assemble simulation experiments for the hardware acquisition cost analysis exercise demonstrated in Section 6.3.

6.3 Hardware acquisition cost analysis

The core proposition of this thesis is that simulation can be effectively used to aid and improve the TCO estimation process for cloud data centres by calculating the effects of resource management policies on QoS and cost. This section demonstrates the benefits of using the proposed approach for determining typical objectives for cloud data centre planning and operations. The main advantage of simulation is the ability for users to carry out offline experiments with representative system models. The experimentation can include change in model structure or attributes to understand system behaviour and its limits. By combining a cloud simulation framework and TCO models a user is able to conduct “what-if” analyses which can help in finding a desired balance between operational efficiency and a financial value for cloud data centre planning and management. This section provides a detailed, practical example of such an approach for the existing testbed of Company X described in Section 6.2.1 and Table 6-1.

The scenario presented in this chapter is a real hardware upgrade scenario that Company X underwent in early 2018. The case arose that Company X wanted to sell part of their old hardware from 2013 and invest partially recouped money into new, more efficient server hardware. In consultation with Company X they shared a hardware configuration that was under investigation and financial data of the acquisition cost quote. As expected, the proposed acquisition option of new computer equipment had significantly different energy, design and performance properties all of which had unknown implications on QoS and costs of the data centre. To better understand the effect the new hardware upgrade will have on the data centre performance and costs the methodology and modelling techniques proposed in this thesis were used for analysis.

The hardware acquisition analysis logical design is shown in Figure 6-20. The idea is to simulate the same workload and resource management decisions for old and new hardware and then analyse results. Experiment scenario models remain the same for both experiments only the hardware models change. The experiment scenarios consist of application models which were acquired earlier during validation phase (see Section 6.2), a generated VM lifecycle model and range of costs obtained from Company X invoices and public sources.

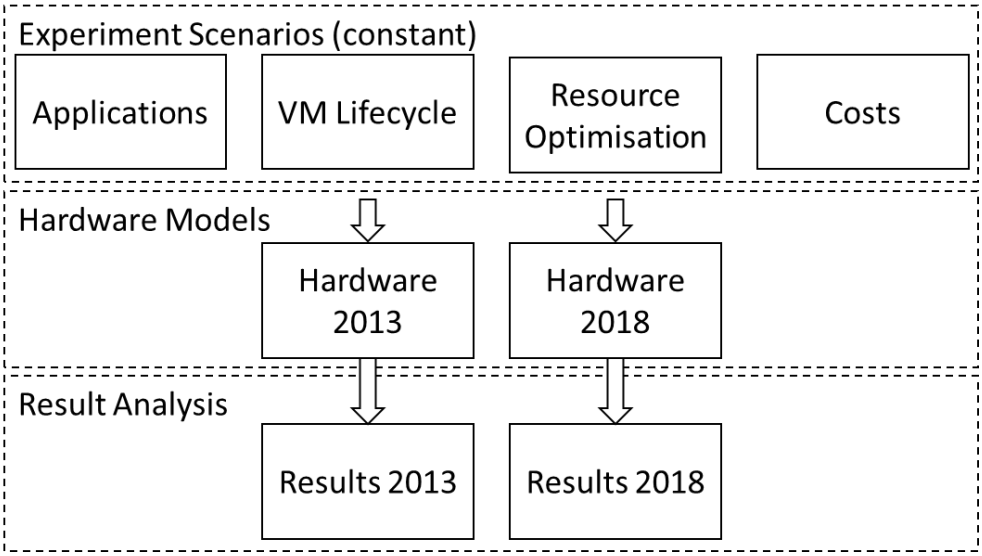


Figure 6-20: "What-if" experiment scenario design

A number of experiments are then executed using different combinations of optimisation policies for new and old hardware models. Results from all simulation experiments are then compared side by side to better understand the potential difference in costs and QoS.

6.3.1 Data centre TCO analysis

As a demonstration for the proposed approach the following work shows the application of simulation for TCO calculation. The supported "what-if" analysis takes the TCO and performance calculations of an existing data centre and evaluates the effects of a hardware upgrade.

The data centre setup shown in Table 6-14 represents an actual cost quote docket from an equipment vendor for Company X's data centre presented earlier in Table 6-1. The whole data centre is mounted in a single rack and consists of a network switch,

three controller nodes, a file server (storage) node, four Intel Haswell CPU based compute nodes and twelve Intel SandyBridge CPU based nodes. The offer came in a bundle at a cost of €100k and was acquired and installed in 2013.

Nr	Qty	Equipment description
1	1	**SDN Switch NEC Programmable Flow UNIVERGE PF5240 (1U)**
2	1	**Controller Server NEC HPC 144Rc-1 (1U), 2x Nodes**: -Open Flow Controller Node - 2x Intel Xeon X5670 6-Core Westmere Processor (2.93 GHz) - 12 GB ECC Reg. DDR3 Memory (6x 2GB) - 2x Gigabit Ethernet Controller, onboard - 2x 250GB SATA HDD, 7.2k rpm, Raid-1 -Cloud Controller Node - 2x Intel Xeon X5670 6-Core Westmere Processor (2.93 GHz) - 48 GB ECC Reg. DDR3 Memory - 2x Gigabit Ethernet Controller, onboard - 2x 1TB SATA HDD, 7.2k rpm, Raid-1
3	1	**Cloud Network Node NEC HPC 124Rd-1 (1U)**: - 2x Intel Xeon E5-2620 6-Core SandyBridge Processor (2.0 GHz) - 64 GB ECC Reg. DDR3 Memory - 2x 1TB SATA HDD, 7.2k rpm, Raid-1
4	1	**Fileserver Node NEC HPC 1212Rd-2 (2U)**: - 2x Intel Xeon E5-2620 6-Core SandyBridge Processor (2.0 GHz) - 64 GB ECC Reg. DDR3 Memory - 2x 500GB SATA HDD, 7.2k rpm, configured as Raid-1 - 6x 2TB SATA HDD, 7.2k rpm
5	1	**Cloud Worker Server NEC HPC 1812Rf-2 (2U)** with 4 Nodes: - 2x Intel Xeon E5-2630v3 8-Core Haswell Processor (2.4 GHz) 2 Nodes equipped with: - 64 GB ECC Reg. DDR4 Memory - 2x 1TB SATA HDD, 7.2k rpm 2 Nodes equipped with: - 128 GB ECC Reg. DDR4 Memory - 2x SSD SATA, 240 GB
6	3	**Cloud Worker Server NEC HPC 1812Rd-2 (2U)** with 4 Nodes: - 2x Intel Xeon E5-2670 8-Core SandyBridge Processor (2.6 GHz) 6 Nodes equipped with: - 64 GB ECC Reg. DDR3 Memory 6 Nodes equipped with:

		- 128 GB ECC Reg. DDR3 Memory 2 Nodes equipped with: - 2x SSD SATA, 240 GB 3 Nodes equipped with: - 2x 1TB SATA HDD, 7.2k rpm
Total Bundle Cost: €100,000		

Table 6-14: Baseline data centre configuration acquired in 2013

To calculate the TCO of the data centre the general formula shown in Equation 1 is used. However, because the installation of the data centre is housed inside the large organisations total premises some of the costs i.e. maintenance, facilities and cooling energy are absorbed by already existing internal amenities and agreements and therefore are not available explicitly to this study. However, in a bid to present the complete view for the TCO calculation method these costs are estimated based on information from secondary sources. In addition, the presented equipment quote, which includes both network and server equipment, reflects a bundle deal meaning that the sum of the $Cost_{servers}$ and $Cost_{network}$ is combined into one variable - $Cost_{racknodes}$.

$$Cost_{racknodes} = \frac{(P_{servers} + P_{network}) - P_{salvage}}{Years} \quad (23)$$

$$Cost_{racknodes} = \frac{100k - 23k}{5} = 15.4k \quad (24)$$

The annual equipment cost is calculated using equation (23) with the standard linear depreciation set over five years with a salvage value of €23k as the four Haswell based nodes were sold in 2018. By using formula (24) the total annual cost of assets, including depreciation, equals €15.4k for the Company X data centre.

The cost of employees who provide support and maintenance services for the data centre equipment is estimated using equations (5) and (6). The exact data is not available to us as Company Y's data centre maintenance costs are absorbed by staff spread across the broader organisation, therefore, these calculations are based on data derived from external sources. To complete this estimation the comparable area measurements and number of employees of a typical data centre was used with specific estimations based on a standalone data centre project in Ireland with plans to employ 150 people in a 31,000sq.m data centre building (O'Brien, 2016) . Next, the average employee salary comes from mapping advertised job roles i.e. Operations Manager,

Hardware Operations Engineer, Facilities Technician, Hardware Operations Quality Assurance lead and Data Centre Technician Assistant (Google, 2018) where salaries range from €74,287 to €50,226 per year (Glassdoor, 2018). The additional area estimation coefficient K value was set to 2.5 as an estimation of extra additional non-data hall space needed to host auxiliary employee facilities and cooling equipment. Finally, the rack area measurements were derived by using actual measurements of the width and the length of the rack unit (42U, 2018).

$$Cost_{maintenance} = Datahall_{area} \times N_{employees_{sqm}} \times Salary_{avg} = 0.70 \times 2.5 \times 0.005517946 \times 62{,}257 = 601 \quad (25)$$

The cost of facilities which are used to host the data centre equipment similarly to the maintenance costs are estimated based on the space taken by the number of racks in a data hall area and the depreciation of the building over 20-year period. Firstly, the cost of facilities per square meter is calculated by dividing the total cost of building construction by the total area of the site where the building is located (2). Estimating this presents a value of $Cost_{building_{facilities}}$ as €200m (O'Brien, 2016) and $Site_{area}$as 31,000sq.m with a cost of facilities per square meter set at:

$$Cost_{facilities_{sqm}} = \frac{Cost_{building_{facilities}}}{Site_{area}} = \frac{200{,}000{,}000}{31{,}000} = 6{,}452 \quad (26)$$

Our $Datahall_{area}$amounts to the space estimation of a single rack which equals 1.75 sq.m. Estimated yearly facilities cost over a 20 year depreciation period using formula (3) is calculated as:

$$Cost_{facilities} = Cost_{facilities_{sqm}} \times Datahall_{area} \times A_t = 6451.612903 \times 1.75 \times \frac{1}{20} = 565 \quad (27)$$

Full annual TCO value also requires calculating energy consumption of the data centre. As shown in Equation 21 a $Cost_{energy}$ value is a combination of static energy consumption calculations and simulated dynamic energy calculations. Static energy is calculated by using manufacturer specifications for the auxiliary equipment which is not used for hosting VMs, such as network switches, controller and storage nodes.

The power draw values in Table 6-15 for each individual asset were obtained from manufacturers component specifications and consolidated using equation (18), bringing the estimated E_{static} value to 9,960.12 kWh/year. An estimated energy cost for

this exercise was taken as €0.091 per kWh, which was calculated by using the average of day and night tariff rates taken from a large electricity supplier and was valid as of August 2018.

Asset name	Power draw (W)	Energy (kWh/year)	Cost Per Annum (€)
NEC PF5240	264	2,312.64	210.45024
NEC HPC 144Rc	281	2,461.56	224.00196
NEC HPC 124Rd	269	2,396.59	218.08969
NEC HPC 1212Rd	323	2,829.48	257.48268
Total ($Cost_{static}$)	1,137	9,960.12	906.37092

Table 6-15: Static energy consumption calculation

To simulate power consumption and performance data for the VM hosting compute nodes the simulation model of the data centre was implemented using PDCM and LDCM meta models. To reflect a larger range of application types both types of application presented in the validation experiments in Section 6.2 were used to create a pool of application templates and VM flavours. The merged application pool (shown in Figure 6-21) is then used to generate an experiment scenario workload which represents a VM lifecycle i.e. VM admission and VM termination events.

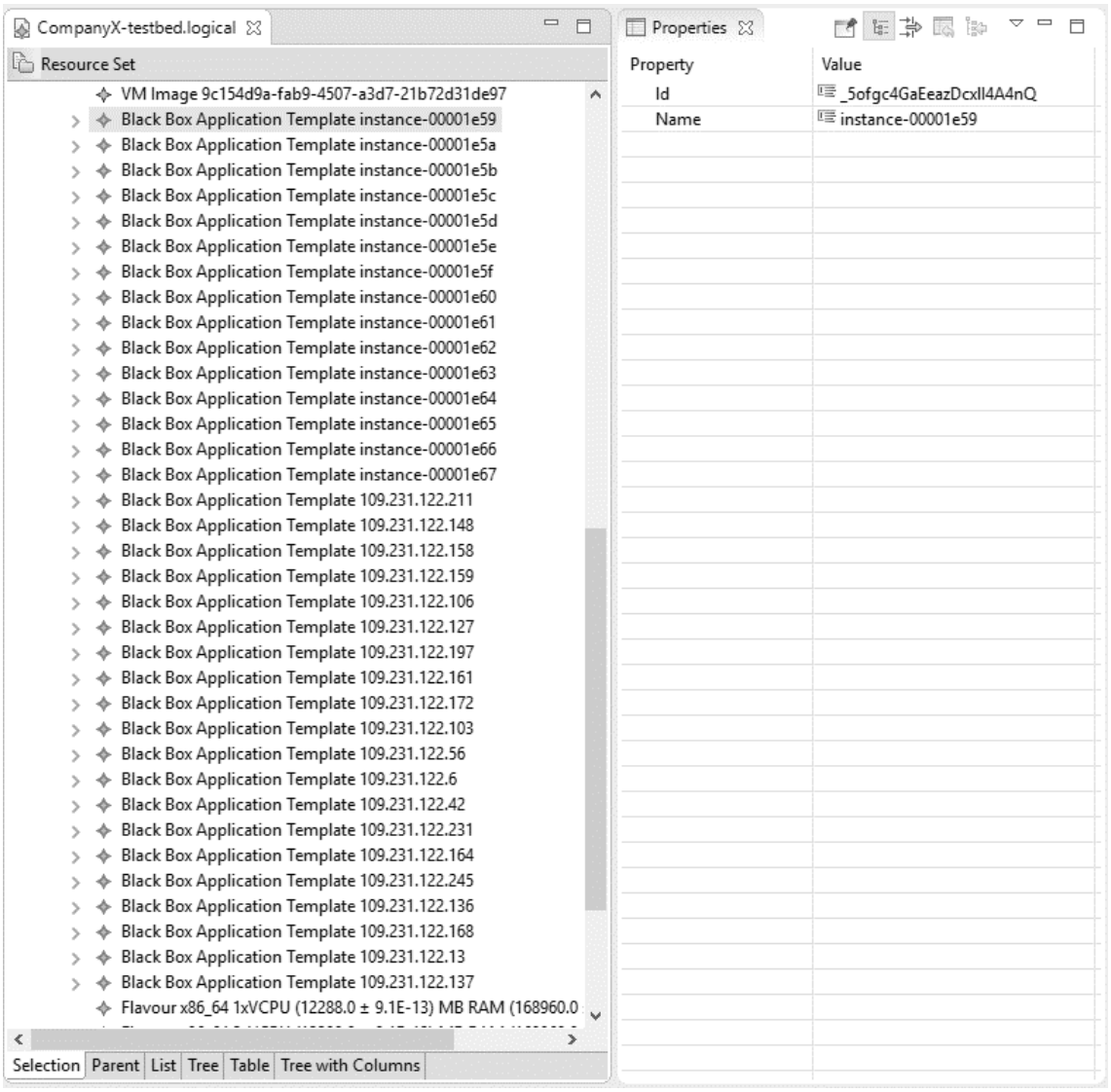


Figure 6-21: LDCM of Company Y with merged application models

During the experiment scenario execution, a total of 500 VMs were deployed to the system reaching a peak of just under 200 VMs running simultaneously (shown in Figure 6-22). The workload was designed to gradually peak resource demand of the compute nodes of the data centre but avoiding CPU core sharing among VMs.

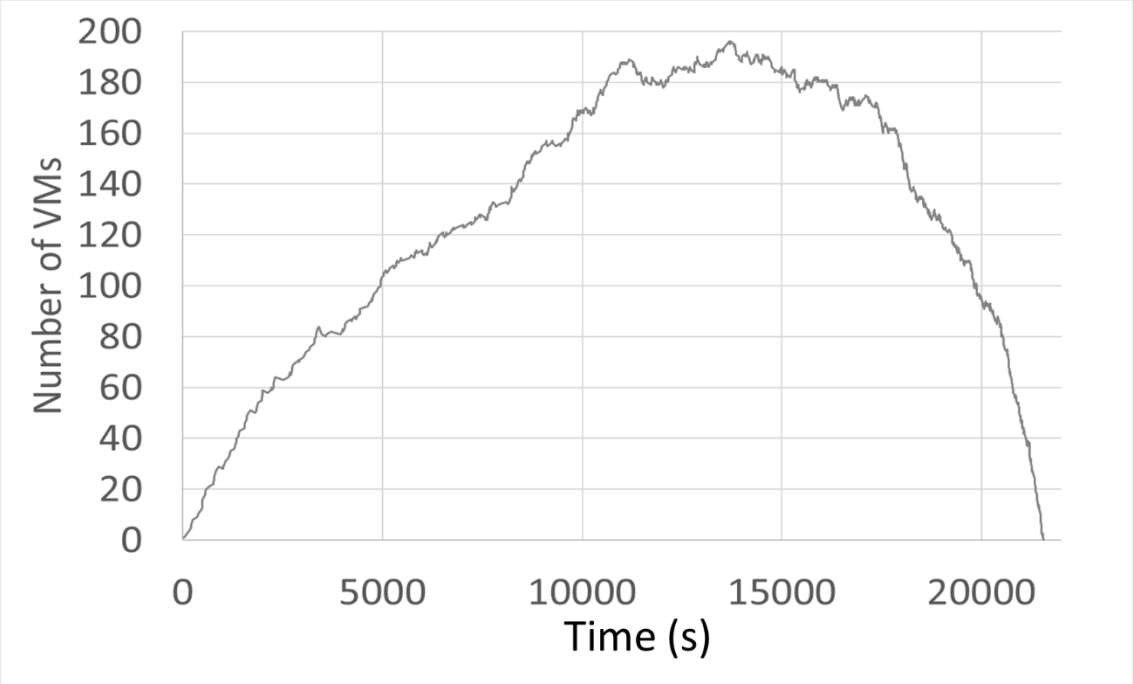


Figure 6-22: VM admission experiment scenario

For the energy consumption prediction four different resource management policies were simulated by conducting four separate simulation runs, changing only the policy settings, shown in Table 6-16. First experiment EO-1 was executed with a simple VM admission algorithm that attempts to spread the VM workload evenly inside the cloud data centre. Every new VM is deployed to a compute node with the least VMs deployed. No migration policy is implemented meaning that VMs remain on the initially selected node throughout the lifecycle. No energy saving actions are implemented, meaning that the compute nodes always remain ON even when there are no VMs deployed. The second experiment, EO-2, uses the same VM admission policy, however in this instance the nodes are switched off when there are no VMs deployed. Experiment EO-3 changes the admission policy to CPU core based consolidation, which admits VMs to a node until the CPU cores resource is fully depleted, then it moves to next node to “fill it up” until no free CPU cores remain available. EO3 also keeps nodes switched off if no active VMs are deployed. Finally, the EO-4 experiment in addition to CPU based consolidation admission policy adds a dynamic migration policy which is also based on available CPU cores. The migration policy periodically checks if already running VMs could be consolidated into fewer nodes based on the CPU cores availability. If the CPU resources

are available, then VMs are moved to increase node utilisation and keep fewer nodes running.

ID	Admission Policy	Migration	Energy Saving
EO-1	Load balancing	No	No
EO-2	Load balancing	No	Yes
EO-3	Consolidation (CPU)	No	Yes
EO-4	Consolidation (CPU)	Consolidation (CPU)	Yes

Table 6-16: Simulated resource management policies

Simulation results were used to calculate power consumption based on estimated compute resource utilisation using equation (13) and (14). Figure 6-23 presents the time series data side-by-side for experiment power demand using different resource management policies as per Table 6-17. Experiment EO-1 and EO-2 power demand is almost identical with the exception of the beginning of the experiment where EO-2 starts VM admission with all compute nodes switched off and EO-1 starts with all compute node running in idle state taking slightly more power. Experiment EO-3 cuts across EO-1, EO-2 and EO-4. EO-3 is using a CPU core based consolidation VM admission policy, at the beginning of the experiment when the bulk of the VMs arrive to the data centre, where the trend matches EO-4 consolidation levels. However, further into the experiment it can be seen at around the 5,000th second mark, power consumption continues to grow until it reaches a similar level of EO-1 and EO-2 experiments. This behaviour can be explained by the lack of a consolidation policy, hence as some VMs are being terminated nodes are still hosting spread out VMs and cannot be shut down to save power. Power estimation for experiment EO-4 yields the lowest results. By continuous VM consolidation through VM admission and migration policies fewer nodes are being used to process the workload leading to more efficient resource utilisation and fewer powered on nodes to process the same workload.

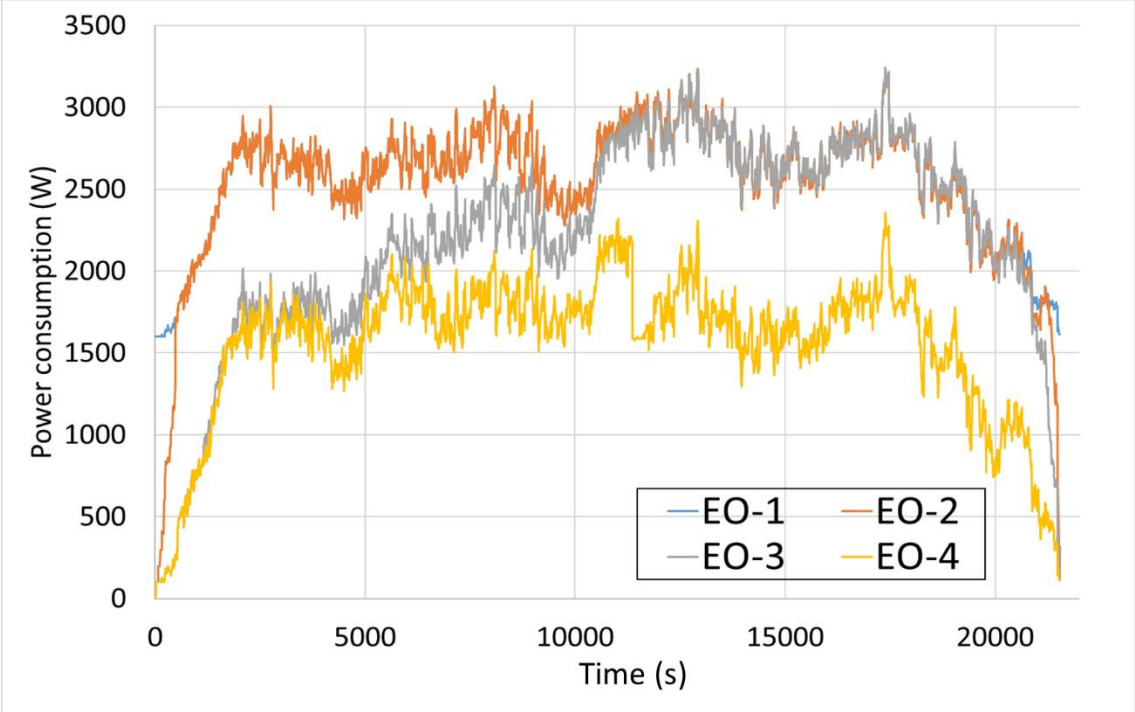


Figure 6-23: Power demand estimation for different resource management policies

Aggressive VM consolidation policies can lead to service performance drop caused by VM resource contention, leading to response delays, longer processing times and in severe cases requests drops. By looking at the CPU utilisation data for the EO-4 experiment, shown in Figure 6-24, a CPU utilisation peak is observed for the *SandyBridge07* (blue) and *SandyBridge02* (light green) nodes. The peak duration appears to be short under current workload conditions, but still can cause service performance problems for mission critical applications.

Name	Energy (kWh)	Cost (€)	Cost Per Annum (€)
EO-1	15.4	1.5	2,240
EO-2	15.2	1.5	2,212
EO-3	13.2	1.3	1,914
EO-4	9.3	0.9	1,348

Table 6-17: Energy demand and costs estimation per policy

The energy cost is calculated by firstly, using the power conversion formula (15) applied to the time series simulation power consumption to calculate kWh consumed during the experiment time by all the compute nodes (16) and then multiplied by cost per kWh acquired from power supplier (17). The cost per annum value was calculated by

dividing the number of seconds in a year to the experiment duration and then multiplying it by the cost per experiment.

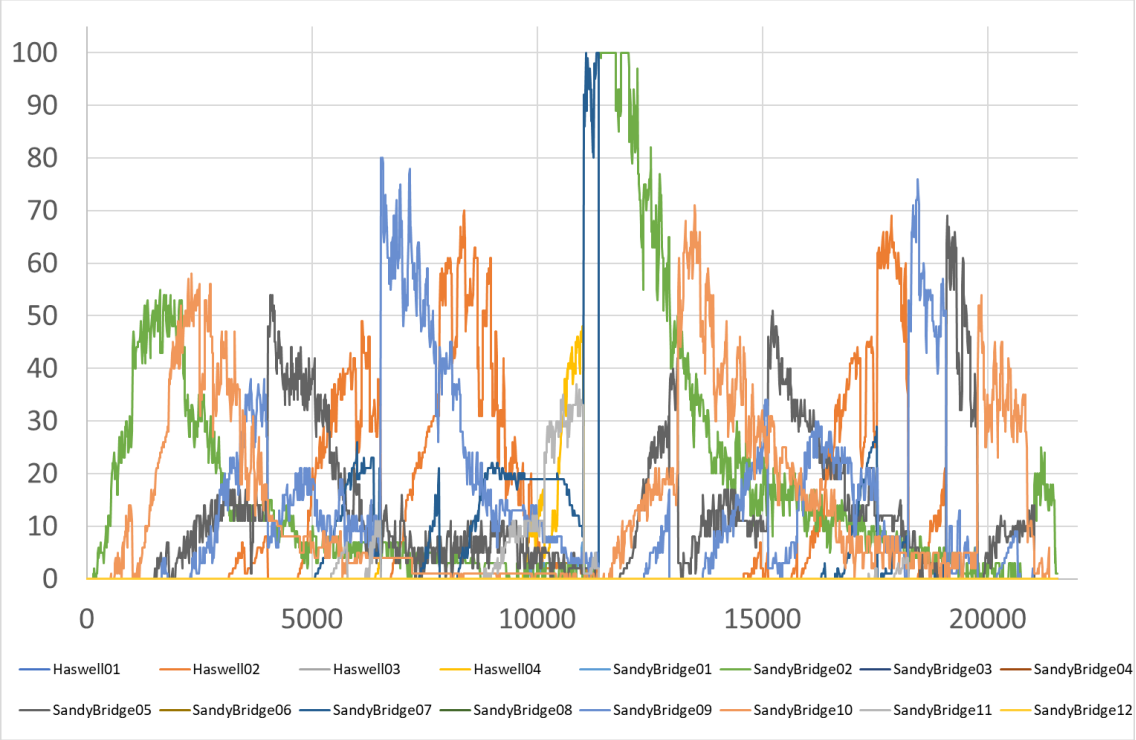


Figure 6-24: CPU utilisation of EO-4 experiment

By processing workload, the nodes emit heat and are required to be cooled off to remain in an operational state. Cooling solutions are setup to regulate the temperature inside a server room. In the current experiment, data on cooling systems and their power demand is not explicitly available. Therefore, the Power Usage Effectiveness (PUE) metric is used to calculate additional power expenditure outside the data hall. For every experiment the sum of the $Cost_{compusim}$ and $Cost_{static}$ is taken and multiplied by the average PUE value of 1.8 (Avgerinou, Bertoldi and Castellazzi, 2017) as shown in equation (20). The PUE of 1.8 means that for every Watt consumed inside the data hall 1.8 Watt is used for cooling. The results of energy estimation costs consumed by cooling and total data centre energy costs ($Cost_{energy}$) per each experiment are presented in Table 6-18. The calculation results show the lowest power consumption cost comes from network switches and controller nodes and the highest for cooling energy estimation with the lowest total energy cost estimated result for EO-4.

Name	$Cost_{compsim}$ (€)	$Cost_{static}$ (€)	$Cost_{cooling}$ (€)	$Cost_{energy}$ (€)
EO-1	2,240	906	5,663	**8,809**
EO-2	2,212		5,614	**8,732**
EO-3	1,914		5,077	**7,898**
EO-4	1,348		4,059	**6,313**

Table 6-18: Energy cost estimation for each experiment

Summing together estimated annual energy costs, data hall equipment cost, maintenance cost and facilities cost the TCO of the data centre was calculated with results presented in Table 6-19. Results show that by using different resource management policies cloud data centre energy cost can be reduced, yielding up to ~10% yearly savings in TCO figures.

Name	$Cost_{energy}$ (€)	$Cost_{racknodes}$ (€)	$Cost_{maintenance}$ (€)	$Cost_{facilities}$ (€)	$Cost_{total}$ (€)	Savings (%)
EO-1	8,809	15,400	601	565	**25,375**	0
EO-2	8,732				**25,298**	0.3
EO-3	7,898				**24,464**	3.6
EO-4	6,313				**22,879**	9.9

Table 6-19: TCO component summary and calculation

The proposed TCO estimation method calculates a fraction of cost for maintenance and facilities based on the rack occupied space in the data hall of a large-scale cloud data centre allowing estimation of costs based on a single rack. If data centre size is extrapolated by adding additional racks the cost relationship should hold making it possible to estimate TCO at a larger scale. Current estimation results also show that the equipment cost ($\boldsymbol{Cost_{racknodes}}$) is the highest variable in TCO calculation equation making it an important consideration point when expanding or upgrading cloud data centre.

6.3.2 Hardware upgrade TCO analysis

In the following section the proposed TCO estimation method is used to analyse the data centre equipment upgrade decision. The aim of this exercise is to demonstrate the value of using simulated data to gain insights on operational efficiency and costs in relation to the data centre under investigation. Firstly, TCO is estimated and the performance of the data centre assessed if the new equipment were to be installed by

creating new data centre simulation models and executing the same workload experiments as shown in Table 6-16. Secondly, the new simulation-based results are compared with the previous initial data centre setup to identify differences in system behaviour and cost assessment.

In early 2018 Company X had the opportunity to upgrade existing data centre equipment described in Table 6-14 by partially replacing it with new equipment described in Table 6-20. The plan was to replace the existing 2U form factor "Cloud Worker Server NEC HPC 1812Rf" containing four Haswell CPU compute nodes with 3U form factor "MicroBlade MBI-6118G-T41X" installation containing 14 compute nodes. During the upgrade, to increase network throughput new network switch NIC's (Network Card Interfaces) and cabling would be added to the system. To understand the impact on QoS and costs the proposed modelling methodology was used for the upgrade option evaluation.

Nr	Qty	Equipment upgrade description	Price (€)
1	1	**Mellanox SN2100 Switch (1U)**	9k
2	23	**Mellanox MCX4131A-GCAT ConnectX-4 Lx EN Network Interface Card 50GbE**	9k
3	1	**Cabling**	2k
4	1	**MicroBlade MBI-6118G-T41X (3U)** with 14 Nodes: - 1x Intel Xeon D-1541 8 Core Processor (2.6 GHz) - 128 GB ECC Reg. DDR4 Memory - 2x 2TB SATA HDD, 7.2k rpm -2x512GB SATA SSD	50k

Table 6-20: Data centre upgrades asset list made in 2018

As also shown in Table 6-20, a MicroBlade server is a high-density installation of 14 compute nodes that fit in a small space of 3U form factor. Each node contains 128 GB DDR4 Memory and 2x2TB HDD and 2x512GB SSD disks, additional memory and storage capacity is significant improvement comparing to the older NEC HPC 1812Rf server. The CPU performance of the new MicroBlade nodes have only one Intel Xeon D-1541 CPU with 8 (16 virtual) cores comparing to two Intel Haswell CPUs with a total of 16 (32 virtual) cores.

$$Cost_{racknodes} = \frac{(P_{servers} + P_{network}) - P_{salvage}}{Years} = \frac{(50{,}000 + 20{,}000) - 7{,}000}{5} = 12{,}600 \quad (28)$$

The annual cost of the new equipment upgrade is calculated as the sum of compute and network equipment minus the arbitrary 10% salvage value. As shown in equation (28) the annual equipment cost equals €12,600.

	Intel Xeon E5-2630 v3 @ 2.40GHz	Intel Xeon D-1541 @ 2.10GHz
Price	$582.99 BUY NOW!	Search Online
Socket Type	LGA2011-v3	FCBGA1667
CPU Class	Server	Server
Clockspeed	2.4 GHz	2.1 GHz
Turbo Speed	Up to 3.2 GHz	Up to 2.7 GHz
# of Physical Cores	8 (2 logical cores per physical)	8 (2 logical cores per physical)
Max TDP	85W	45W
Yearly Running Cost	$15.51	$8.21
First Seen on Chart	Q4 2014	Q1 2016
# of Samples	132	15
Single Thread Rating	1747	1423
CPU Mark	12812	11056

Figure 6-25: Intel Haswell (Xeon E5-2630) vs new Intel Xeon D-1541 comparison (PassMark Software, 2018)

From more detailed CPU data shown in Figure 6-23 it can be seen that the new CPU replacement produces a similar CPU Mark benchmark performance at almost half of Thermal Design Power (TDP) of 45W meaning also much lower actual power consumption per CPU. From the power consumption experiments presented by the MicroBlade manufacturer Supermicro and 3rd party benchmarks of a similar configuration system using Intel Xeon D-1541 processors the idle minimum power demand per node equals 36W and maximum power demand at a maximum CPU load reaches 90W per node. Both values are used when constructing our simulation linear power models as P_{idle} and P_{busy} values respectively, as shown in equation (13).

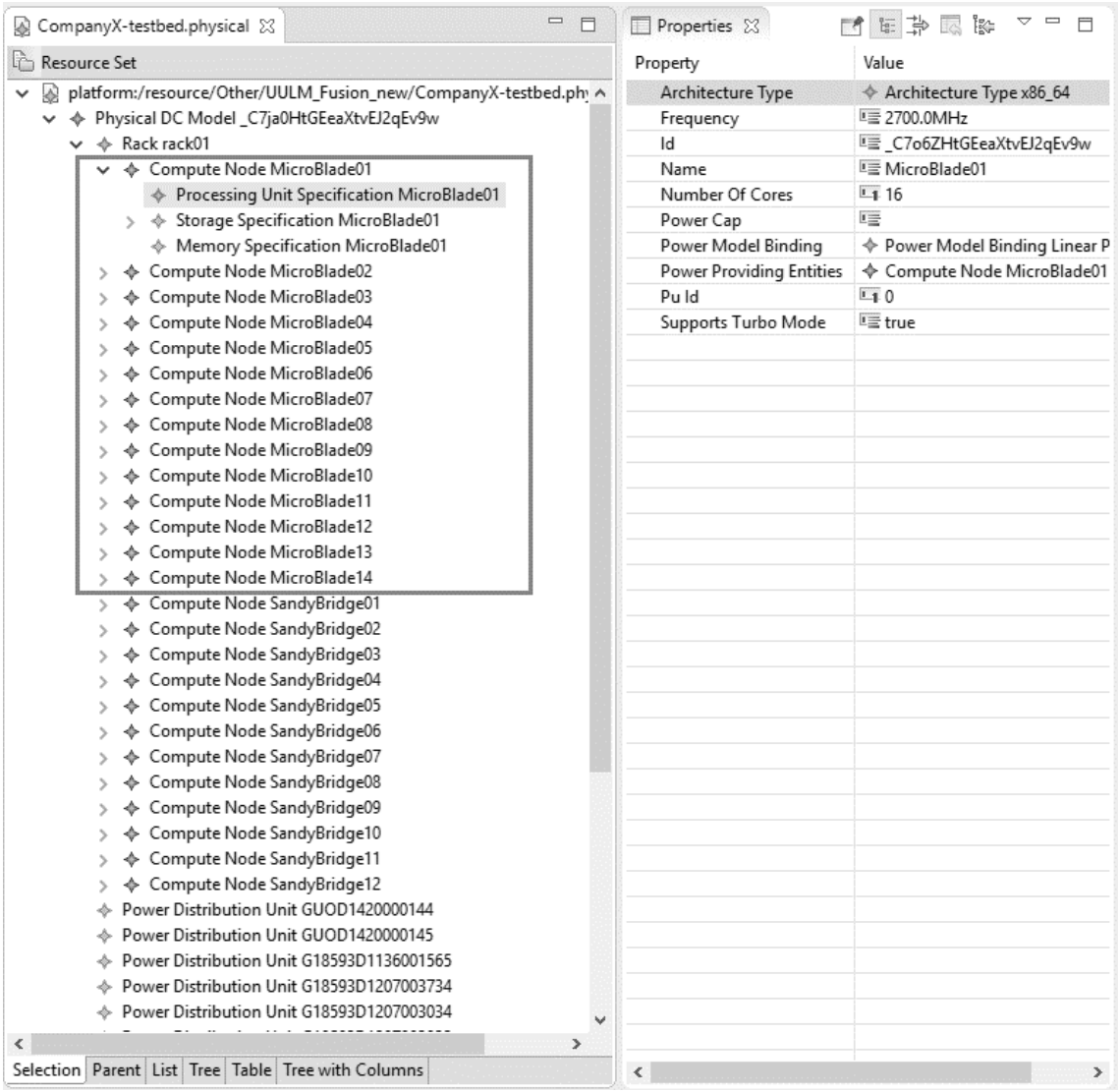


Figure 6-26: PDCM of Company Y with new hardware substitution

The changes included the addition of 14 new MicroBlade nodes with a new switch while also removing 4 Haswell CPU based nodes from the physical infrastructure model (shown in Figure 6-26). It also required an adjustment to the power binding models with new power consumption bracket information and the addition of a new range of hypervisors to the logical infrastructure model to make the connection to new available compute nodes. Next, the simulations EOU-1,2,3 and 4 (U-for upgrade) were run using the same resource management policies shown in Table 6-16, the same applications shown in Figure 6-21 and the same workload demand shown in Figure 6-22.

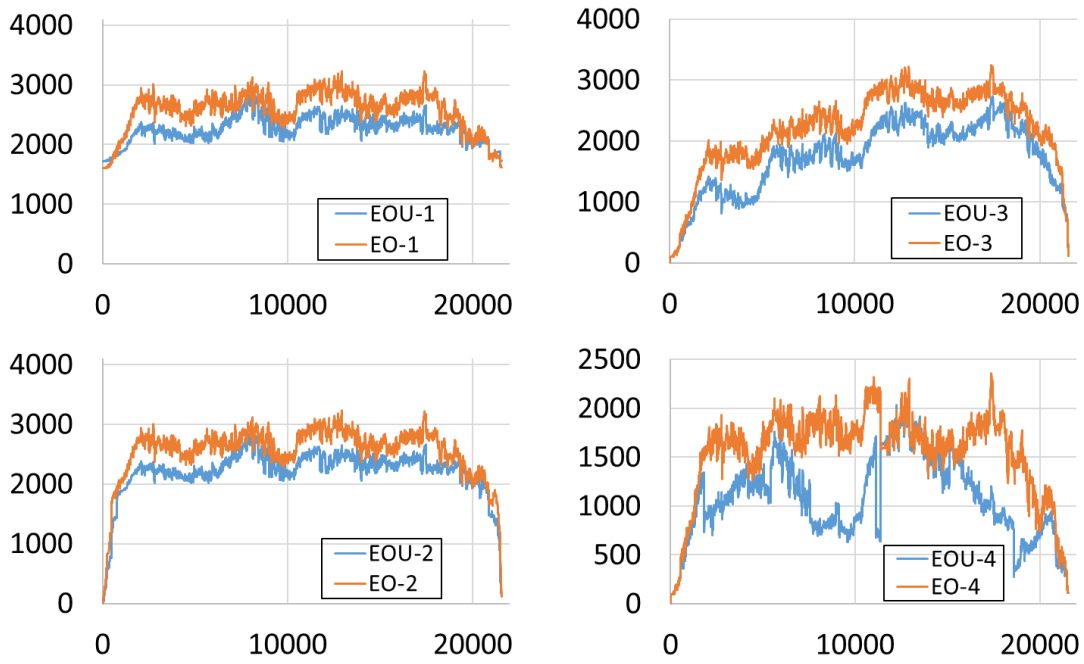


Figure 6-27: Power demand comparison with upgraded data centre

From the simulated power demand results shown in Figure 6-27 similar trends are observed for each policy, with the lower power demand profile. The hardware upgrade increases compute resource capacity at lower power demand rates. Interesting behaviour is observed during comparison of EOU-4 and EO-4, where both experiments are using CPU core based VM admission policies and VM migration (consolidation) policies. Where other experiments have almost identical power demand trends the EOU-4 results show a significant difference in demand patterns with sharp drops in power consumption. Through closer analysis of the simulation experiment logs the power demand drops appear because of termination of large VMs on SandyBridge nodes making it possible to free up and shut down completely the MicroBlade nodes. This effect is occurring because of the lower CPU core quantity on the newly installed MicroBlades making them more modular allowing them to switch off unused capacity leading to better power efficiency under the current experimental circumstances.

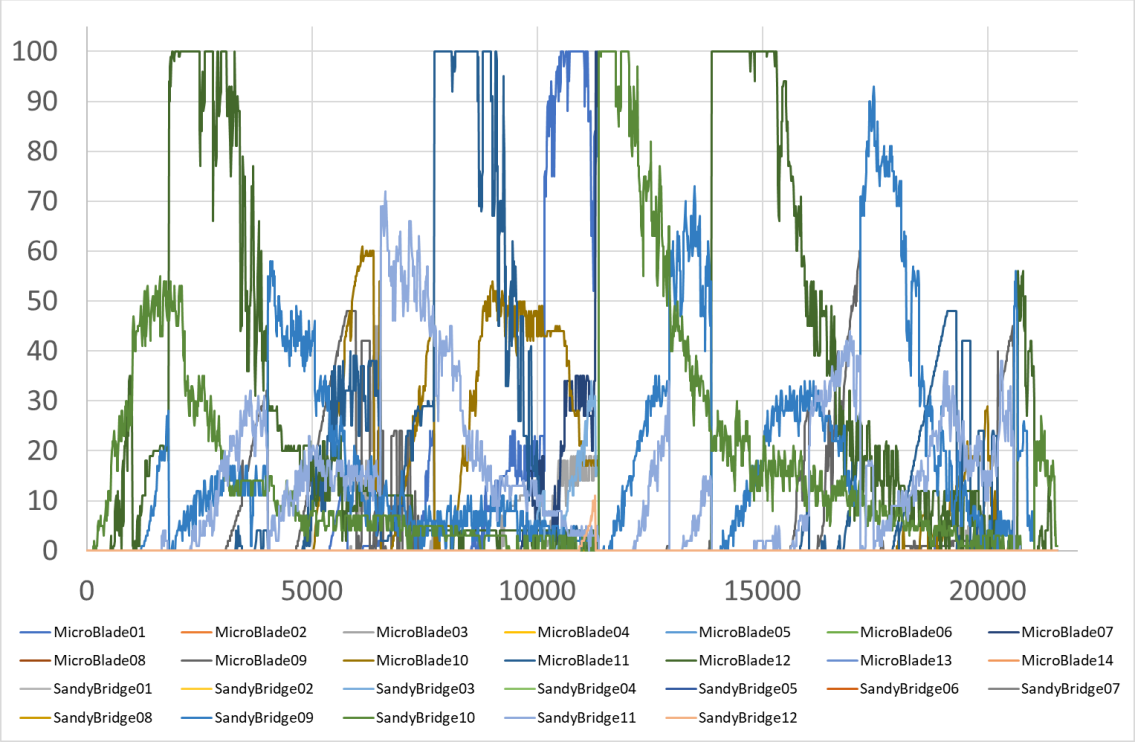


Figure 6-28: CPU utilisation of EOU-4 experiment

To better understand the trade-off in experiment EOU-4, between power saving and resource availability, the CPU utilisation shown in Figure 6-28 is examined. From this figure it can be seen that the MicroBlade compute nodes Nr. 01, 11, 12 and the SandyBridge 10 nodes reach lengthy periods of 100% utilisation. By using the particular VM consolidation policy the lower CPU core capacity MicroBlade nodes are filling up quicker leading to the increased workload contention and performance bottlenecks. Even though the new equipment upgrade consists of higher overall CPU core numbers, the performance is hindered by the migration policy which is not considering new hardware properties.

Name	Energy (kWh)	Cost (€)	Cost Per Annum (€)
EOU-1	13.6	1.3	1,966
EOU-2	13.2	1.3	1,915
EOU-3	10.4	1	1,505
EOU-4	6.5	0.6	941

Table 6-21: Energy demand and cost estimation per policy of upgraded data centre

To proceed with the TCO calculation, firstly annual energy demand and cost is estimated by using the power demand simulation results. The results shown in Table 6-21 as expected show a significant drop in power consumption for the EOU-4 experiment which is using a CPU core based VM consolidation policy.

Asset name	Power draw (W)	Energy (kWh/year)
Mellanox SN2100 Switch	248	2,172.48

Table 6-22: Static power estimation of upgraded switch (Mellanox Technologies, 2018)

To estimate static equipment power consumption for the existing hardware listed in Table 6-15 a newly acquired Mellanox switch is added and its yearly energy consumption estimation shown in Table 6-22. This brings the total static energy consumption to 12,132.6 kWh/year and $Cost_{static}$ value to €1,202.

Name	$Cost_{compsim}$ (€)	$Cost_{static}$ (€)	$Cost_{cooling}$ (€)	$Cost_{energy}$ (€)
EOU-1	1,966	1,202	5,703	**8,872**
EOU-2	1,915		5,611	**8,729**
EOU-3	1,505		4,873	**7,580**
EOU-4	941		3,857	**5,999**

Table 6-23: Energy cost estimation for each experiment on upgraded hardware

All three types of energy cost i.e. simulated compute nodes, static equipment estimations and cooling are summed to calculate total estimated energy cost with results presented in Table 6-23. From the presented estimations it can be seen that the increase in static costs also increases cooling costs which makes the gap in overall power consumption smaller when comparing to results shown in Table 6-18. Even when comparing total energy cost between EO-1 and EOU-1, the older equipment is more cost efficient under the tested VM allocation policy.

Name	$Cost_{energy}$ (€)	$Cost_{racknodes}$ (€)	$Cost_{maintenance}$ (€)	$Cost_{facilities}$ (€)	$Cost_{total}$ (€)	Savings (%)
EOU-1	8,872	12,600	601	565	22,638	0
EOU-2	8,729				22,494	0.6
EOU-3	7,580				21,346	5.7
EOU-4	5,999				19,765	12.8

Table 6-24: TCO component summary and calculation

Yearly values are added for the estimated energy consumption to calculate the TCO for each experiment, equipment costs, cost of maintenance and facilities. Since the size of the data centre remains unchanged the estimated costs for maintenance and facilities remain the same. The variables for each experiment are presented in Table 6-24. TCO comparison shows that from the cost benefit point of view a potential 12.8% savings can be made by choosing different VM admission and migration policies. However, such a finding comes with a warning from the resource utilisation analysis shown in Figure 6-28 which identifies bottlenecks in CPU demand.

$$ROI = \frac{Savings}{Investment} \times 100\% \quad (29)$$

The investment in new equipment is made to ensure service competitiveness and better cost efficiency. Both of these improvements should result in profit growth of the data centre. The Return on Investment (ROI) metric allows for the calculation of the percentage of recovered funds from the investment i.e. new hardware purchase in the given case. The ROI is calculated by dividing achieved savings by the investment amount and multiplied by 100% to get the percentile value, the formula is shown in (29)[2].

Name	TCO	Saving	ROI (%)
EO-1	25,375	2,737	22
EOU-1	22,638		
EO-2	25,298	2,804	22
EOU-2	22,494		
EO-3	24,464	3,118	25
EOU-3	21,346		
EO-4	22,879	3,114	25
EOU-4	19,765		

Table 6-25: Energy and cost comparison across data centres

In the final analysis, TCO estimations from all eight experiments are presented in Table 6-25. The findings are compared by calculating yearly cost savings and the ROI that new equipment brings. Taking into the account that the presented savings are achieved based only on superior power efficiency of the new proposed equipment, the result appears favourable. However, when analysing the ROI, the saving in energy only

[2] It should be noted, that because amortization costs were already subtracted this assumption will lead to an underestimation of actual costs for longer-lived assets.

amounts to a maximum of 25%. This means that to recover the rest of the new equipment investment costs other means of revenue has to be considered. However, since the new equipment also has more CPU cores, memory and storage an additional increase in revenue could be obtained by selling more services to utilise more available compute resources.

6.4 Conclusions

This chapter presented two test cases of different types of cloud applications, a Business Analytics case and a Scientific Computing case. For both of the case applications a testbed environment was created to enable controlled, isolated benchmark experiments. During these experiments the data was collected which served as an input for the automated generation of parallel simulation models. Identical experiments, as far as was possible were then run on both the testbed scenario and in simulation to verify accuracy, and thus were used for validating the simulation platform and associated results.

During the Business Analytics validation experiments the average error rate was estimated at 7.8%, but due to a proprietary VM migration algorithm implemented in the real system, simulation results began to lose accuracy towards the experimentation finish. In identifying this, these experiments have shown the importance in system behaviour and the impact different resource management policies have and the importance of understanding the logic of cloud management systems for building simulation experiments.

The Scientific Computing use case experiments were more diverse allowing for the testing of multiple different resource management policies against real application behaviour. Since the CACTOS optimisation framework was fully integrated with an OpenStack installation on the Company X testbed, both VM admission and VM consolidation policies were controlled by CactoOpt. Full integration allowed for the simulation of system behaviour with more precision when compared to the Business Analytics experiments. During the comparison of simulation experiments and real results the average error rate was found to be 5.6%. In both validation exercises the simulation experiments average error rate fell below the 10% threshold, as set by the consortium, thus validating the simulation framework as a suitable decision support tool for data centre management.

The latter half of the chapter demonstrates the application of the simulation based TCO estimation method proposed in this thesis. A full physical data centre simulation model was created using the hardware specification obtained directly from an equipment purchase quote. The application model templates were copied over to the logical data centre model from both the Business Analytics and Scientific Computing validation experiments. The workload (VM lifecycle) model was generated to admit VMs to the data centre using the full application template range. Based on these system models the simulation experiments were executed to estimate system performance and TCO. The proposed method allows for the simulation of the cloud data centres behaviour under different VM admission and consolidation policies, estimating energy consumption and compute resource utilisation. A simulation-based energy prediction enables more precise TCO estimation based on actual resource demand patterns collected from real deployed cloud services while also taking into account cloud resource management policies. The presented experiments have shown the difference in energy consumption and costs, associated with various resource management policies. In addition, resource consumption prediction analysis can help to identify possible bottlenecks, some of the power saving techniques can create which may impact upon overall service performance. It was also interesting to note the different system performance results when analysing hardware upgrade decisions. The CPU consumption in the EOU-4 experiment was too high raising a concern in relation to the performance of running this service. By identifying such scenarios at a simulation stage, using the proposed technique, adjustments can be made to the resource management policy so as to better suit new hardware features when deployed to the production system.

In summary, this chapter has shown the practical application of the designed application framework developed in this thesis. What is presented here is a proof of concept analysis which demonstrates the successful validation of the developed approach and demonstrates its usefulness in a small-scale data centre technology upgrade scenario. The models developed within this chapter were at times restricted due to the proprietary nature of a number of the resource management techniques being used within the case study data centres. While this is a slight drawback the results presented in such scenarios were easily identifiable. What this helped to confirm was the value of a fully integrated approach, whereby all data is represented in the modelling platform.

7 Discussion and Conclusions

Cloud computing is a growing trend with an increasing number of users choosing to run their workloads on remote hardware in cloud data centres globally. With this trend for cloud users to avail of cost effective, on demand cloud services, this puts pressure on cloud infrastructure providers to run data centres in a more efficient, cost competitive and reliable manner. To keep up with these demands and technology trends, infrastructure providers need to update their infrastructure and resource management approaches on a regular basis. Constant change in system composition increases the chances of misalignment between user resource demand, hardware capabilities and resource management approaches. This can potentially lead to service quality degradation and cost inefficiency. Hence, comprehensive planning of data centre changes is key to avoiding such issues. However, to date there has been a lack of research that assists infrastructure providers in holistically assessing the impact that different hardware acquisition options and resource management solutions can have on their core business metrics i.e. TCO and QoS.

The growth in cloud computing demand has led to an increase in scale of cloud data centres. Data centre management requires a thorough planning approach that takes into account key aspects of performance and cost. Although there are methods available to aid data centre operators, they are limited in the functionality they offer. The issue of efficient cloud resource management is particularly lacking in the cost estimation area that can operate at the scale of modern cloud data centres. Existing solutions rely on manual user data input which is no longer feasible for a system consisting of thousands of different entities, thus highlighting the need for the development of an automated approach.

Cloud computing data centres form a complex environment that consist of virtual and physical layers which require constant monitoring, frequent maintenance and efficient management at scale. Design and evaluation of configuration options of such complex systems is a challenging task due to the dynamic and nonlinear relations between user demand, infrastructure capacity and costs. Existing research in the area of decision support tools addresses system performance, resource management and costs in a disjointed way, often requiring the use of separate tools for each aspect. Such an isolated approach leads to duplication of effort and generalisation in estimations. To

address shortcomings of existing methods, a solution is required which provides a united approach that is capable of providing cost estimations, system performance and resource management policies.

Findings from literature analysis and formal interaction with CACTOS project industry research partners who manage cloud infrastructure suggests that one of the main goals is to provide a high quality of service while remaining cost competitive. According to Astri (2015) low cloud services costs are also expected from customers, making cost an important topic from infrastructure provider and consumer point of view. However, existing TCO approaches do not take into account the effects of resource management policies resulting in an incomplete cost estimation. Literature shows that TCO is calculated in an isolated process using static formulas in a spreadsheet or in an embedded user interface implementation. Existing disjointed approach might be considered sufficient for a small in-house server room, but for a modern cloud data centre such processes becomes simply infeasible due to the large scale and complexity of the systems. Existing approaches require users to provide manual data entry of cost items, but large-scale cloud data centres have thousands of units of heterogenous server hardware, hence manual data entry can become a very lengthy task. Hence, a research gap was identified in the lack of decision support approaches for cloud data centre TCO estimation connecting core cloud data centre management areas such as hardware capacity planning, resource optimisation and QoS.

7.1 Conclusions

The aim of this thesis was to review and analyse existing cloud data centre decision support techniques and tools with the prospect of advancing current state of the art in order to provide better insights on data centre performance and costs. Essentially creating an enhanced cost-aware decision support approach to aid cloud infrastructure providers. The aim of this work was formulated in the hypothesis and supporting research objectives presented in Chapter 1. In the process, a methodology and set of tools were developed which address the problem. The proposed methodology is capable of delivering more precise cost estimations as it is taking into account user demands, hardware configuration and resource management policies. By linking TCO estimation with resource monitoring and utilisation data, one can gauge cost-based decision impact on the provided services and vice versa. This unique cost calculation approach enables

access to holistic decision base information that provides assistance for end-to-end data centre planning, initial procurement and upgrade decisions.

From the theoretical literature exploration and related research analysis a DES approach was identified as best fitting for task at hand, resulting in simulation framework development and implementation. The framework requirements and design were drawn from the CACTOS project collaboration based on use cases described in the project. This collaboration allowed for deeper simulation-based tooling integration with data collection and resource management frameworks, leading to direct access to data describing the real system and related resource optimisation algorithms. Through the integration with a data collection framework, the work in this thesis proposes autonomous model generation improving user experience with simulation adoption and leading to advancement in the current state of the art. Similarly, integration with the cloud resource management framework allows for direct use of resource management policies from a production environment within simulation experiments. Currently available simulation frameworks require a custom implementation of resource scheduling policies logic, whilst in this improvement the resource management policy needs to be implemented once for the resource management framework only, simplifying the process and saving users effort.

The CACTOS project collaboration also gave access to real cloud infrastructures and data from the test scenarios of Company X and Company Y. Based on these scenarios, simulation models containing infrastructure configuration and cloud application behaviour were created. Since simulation models are generally considered an approximation of real system behaviour it is important to measure both how accurate the approximation is and if simulation delivers results that can be reliable enough to support a decision. Validation methods are quite common in cloud computing simulation with the examples such as iCanCloud validation by Castañé, Núñez, *et al.* (2013) or MDCSim validation by Lim, Sharma, *et al.* (2009). To demonstrate the accuracy of the simulation results, the presented work followed recognised validation procedures such as visual validation and an adopted version of statistical validation developed by Sargent (2010). The results of this statistical comparison aid in the understanding of differences between simulated and measured data in a more granular way, which in turn can be used to aid in future system model development.

Preliminary results from the proposed methodology presented in Chapter 6.3 have received a positive feedback from Company X. Further collaborative research touch-points were identified, and the presented research methodology is planned to be improved through collecting further data and extending applicability in the area of distributed infrastructure systems.

In carrying out this research the following conclusions were drawn:

- The literature review revealed a strong research interest in cloud computing, particularly in resource management algorithms and framework development carried forward from the predecessor paradigm of grid computing. Cloud simulation and modelling frameworks were identified as available decision support tools for compute resource utilisation prediction under different experiment scenarios. TCO calculation frameworks and models were also identified as mechanisms for cost-based decision support aiding cloud data centre management.
- Literature review findings confirmed absence of a single solution that joins resource consumption estimation and TCO estimation, making it difficult to analyse the impact of resource management on costs. To address the issue, an integration between cloud data centre simulation and TCO calculation processes has been proposed as an approach to support the decision-making process, combining cloud data centre performance estimations and related costs. This novel approach uses simulation models and simulation experiment outputs to improve TCO calculation accuracy uniting hardware specification, resource demand patterns and costs attributes into one process. The proposed integration delivers a holistic view of cloud computing data centre performance enriching crucial decision information points.
- From literature analysis and case study evaluation it has been noted that the current approach requires manual data entry for model creation. Since existing approaches do not cater for large scale systems, a software solution has been developed for simulation model creation automation that significantly reduces cloud data centre infrastructure modelling effort. Manual data input to the model of thousand nodes would take too much

time and therefore is no longer considered a feasible solution given the scale of a modern data centre. The software uses a data centre monitoring and data collection framework and a pre-defined simulation meta model to pull data from a real environment and construct a snapshot infrastructure model of the environment.

- Cloud discrete event simulation integration with a TCO model allows for the estimation of cost implications related to a selected resource management strategy. Resource management policies play a significant role in cloud data centre performance and cost. Conducted experimentation results demonstrate the difference in resource utilisation and energy costs between different resource management algorithms. The significance becomes even more apparent when cost estimations are projected over long periods of time, for example over an asset useful life span.
- Simulation models allow users to experiment with a large range of scenarios going beyond test bed capabilities. Such scenarios can include modelling outlier experiments with spikes in resource demands or hardware failures. Simulation also makes it possible to explore design options of hardware topologies using the hardware specification of existing equipment or equipment that has not yet been acquired. Simulation models allow for the impact on QoS and costs to be quickly assessed, inclusive of alternative equipment options and design decisions.
- The simulation results validation process can be challenging when working with real system data. Presented research showed discrepancies of various degrees in simulation results when compared to real system measurements particularly when unknown resource management policies were used. This means that while dealing appropriately with resource demand some of the proprietary cloud resource management solutions were difficult to simulate due to undisclosed behaviour algorithms. However, the majority of cloud management solutions would allow for resource distribution policies to be changed in order to fulfil the

goals of cloud infrastructure providers. Hence resource management behaviour can be moved to a simulation environment.

- Through experimenting with alternative resource management policies, experiments show up to a 12.8% difference in costs that can be achieved when using different resource management techniques. These findings provide strong justification in favour of the use of the presented simulation-based methodology for the TCO calculation of cloud data centres.
- When in a process of acquiring new hardware for a data hall expansion or replacement of assets at the end of their life span, it is important to experiment with different hardware profiles. Presented findings (Chapter 6.3) show an indication of possible bottlenecks due to resource overbooking when using existing resource management approaches on new computing hardware. Inbuilt cost models provide estimated cost information alongside hardware performance data creating a more complete information base for decision making. Such experimentation findings can be used to anticipate the issue at a design stage and come up with a solution prior to asset acquisition.
- Simulation modelling requires additional expertise which is not typically part of the knowledge base of a cloud data centre operator. In order to stimulate simulation adoption, the following steps were taken:
  - Integration with a data centre monitoring framework to aid in automated model creation
  - Direct integration with a cloud optimisation framework for the simulation of resource management policies
  - Development of a user interface for model retrieval, model editing and simulation parameterisation

Overall, the development of the application framework, and its associated testing with the two case organisations, have helped to address a number of research gaps in the literature, which include the following contributions to knowledge:

- Automation of cloud data centre simulation model creation

- Production grade resource management policy analysis
- TCO calculation method improvement to account for resource management decisions
- A decision support methodology that integrates TCO and QoS data in single output.

7.2 Future research

A number of areas touched by the work presented in this thesis can be developed further forming standalone research bodies. Future work recommendations are in the areas of software engineering, simulation modelling and further integration.

Current work is built on simulation experiments using the two real data centres of Company X and Company Y as access was provided through the CACTOS project collaboration. The scenarios executed using these data centres were used to collect real environment data on hardware specification and performance to validate the proposed thesis approach. Future research can be carried out to advance the scale of cloud data centre models in the simulation experiments. Scale would be an interesting research avenue to explore as one of the benefits of simulation techniques. The existing research can easily be extended to cover large scale scenarios provided that detailed information about data centre configuration can be obtained.

The linear energy estimation model (Fan, Weber, *et al.*, 2007) was used for a CPU utilisation-based power draw calculation, however other power estimation models exist. The study of Makaratzis, Giannoutakis, *et al.* (2018) identified also cubic, square and linear interpolation models that have also been applied within other tools. Since power estimation was not the focus of the thesis, the linear model was considered to be sufficient for purpose. However, the existing model implementation can be updated if needed. This way the developed simulation-based framework can be extended to test existing power consumption models or used for further work in developing new power estimation approaches.

The proposed simulation framework was integrated with specific data collection and optimisation frameworks as part of the CACTOS project. Using the simulation framework on its own with another environment would require additional software development effort in order to obtain data for populating simulation models. The adoption of the proposed solution would benefit greatly from wider integration across industry standard monitoring systems. Development effort can be reduced by complying with the existing CACTOS meta-model structure as it can be re-used during other system integrations. In this way the proposed simulation framework can be adopted for use outside of the CACTOS project tooling for the wider compatibility.

Each test scenario presented in the validation work was executed only once. For the duration of every scenario, the infrastructure was isolated from any external interference, however future work can be carried out to improve the accuracy of measurements if the same scenario was to be executed multiple times. Collecting a higher number of measurements would also ensure there are no outliers in the data due to some unforeseen system behaviour. The same logic is applicable to the simulation application model composition where more data would create more natural behaviour. The application model proposed in this work can be further extended to adopt this approach.

While there was an adequate amount of technical data that formed the basis for the cloud data centre hardware infrastructure and workload models, financial data was difficult to obtain. This opens another research avenue of simulation modelling integration with accountancy systems such as ERP and EBP systems. Such work would bridge the gap for cost data acquisition, automating this process and potentially greatly improving simulation application adoption rate.

With annual global IP traffic predicted to triple from 1.5 ZB per year in 2017 to 4.8 ZB per year by 2022 (Cisco, 2018) the use of a distributed compute infrastructure as part of future Edge and Fog computing paradigms unveils additional opportunities for faster, more cost-efficient content delivery that compliments a traditional centralised cloud data centre. Edge computing is tightly coupled with mobile cloud architectures that provide cloud services at the edge of the network locations (Mahmoudi, Mourlin and Battou, 2018). In comparison Fog computing is referred to as a paradigm that resides between smart-end devices and traditional cloud devices (Iorga, Feldman, *et al.*, 2017). Compared to centralised cloud computing, the nature of such geographically distributed

systems comprised of heterogeneous infrastructure and smart devices creates additional compute resource management challenges for infrastructure and service providers as well as service consumers. The research presented in this thesis can serve as a basis and support future work moving into area of distributed infrastructure performance and cost estimation. Presented models and calculation methods can be extended to be able to reflect behaviour of Edge and Fog based systems investigating the domain of virtual Content Delivery Networks (vCDN), Internet of Things (IoT) and Network Function Virtualisation (NFV).

8 References

42U (2018) *42U Rack Dimensions, Cabinet Size, Specifications*. Available at: https://www.42u.com/42U-cabinets.htm (Accessed: 20 July 2018).

Aceto, G., Botta, A., *et al.* (2013) 'Cloud monitoring: A survey', *Computer Networks*. Elsevier B.V., 57(9), pp. 2093–2115. doi: 10.1016/j.comnet.2013.04.001.

Ahmad, R. W., Gani, A., *et al.* (2015) 'A survey on virtual machine migration and server consolidation frameworks for cloud data centers', *Journal of Network and Computer Applications*, 52, pp. 11–25. doi: 10.1016/j.jnca.2015.02.002.

Ahmed, A. and Sabyasachi, A. S. (2014) 'Cloud computing simulators: A detailed survey and future direction', *2014 IEEE International Advance Computing Conference (IACC)*. Ieee, pp. 866–872. doi: 10.1109/IAdCC.2014.6779436.

Alger, D. (2012) *The Art of the Data Center: A Look Inside the Worl's Most Innovative and Compelling Computing Environments*. Available at: https://books.google.com/books?id=vQCeea8Y1cUC&pgis=1.

Ali-Eldin, A., Tordsson, J. and Elmroth, E. (2012) 'An adaptive hybrid elasticity controller for cloud infrastructures', *Proceedings of the 2012 IEEE Network Operations and Management Symposium, NOMS 2012*, (978), pp. 204–212. doi: 10.1109/NOMS.2012.6211900.

Alves, D. C., Batista, B. G., *et al.* (no date) 'CM Cloud Simulator : A Cost Model Simulator Module for Cloudsim'.

Amazon (2016) *Amazon Web Services (AWS) - Cloud Computing Services*. Available at: https://aws.amazon.com/ (Accessed: 12 November 2015).

Apache (2015) *Apache Tomcat*. Available at: https://tomcat.apache.org/ (Accessed: 12 November 2015).

Apache Software Foundation (2017) *Welcome to Apache™ Hadoop®!* Available at: https://hadoop.apache.org/ (Accessed: 13 July 2017).

Arizton (2018) *Data Center Market - Global Outlook and Forecast 2018-2023*, *Research and Markets*. Available at: https://www.researchandmarkets.com/reports/4577457.

Armbrust, M., Armbrust, M., *et al.* (2009) 'Above the clouds: A Berkeley view of cloud computing', *University of California, Berkeley, Tech. Rep. UCB* , pp. 07–013. doi: 10.1145/1721654.1721672.

Armstrong, D., Espling, D., *et al.* (2015) 'Contextualization: dynamic configuration of virtual machines', *Journal of Cloud Computing*, 4(1), p. 17. doi: 10.1186/s13677-015-0042-8.

Astri, L. Y. (2015) 'A Study Literature of Critical Success Factors of Cloud Computing in Organizations', *Procedia Computer Science*. Elsevier Masson SAS, 59(Iccsci), pp. 188–194. doi: 10.1016/j.procs.2015.07.548.

Avgerinou, M., Bertoldi, P. and Castellazzi, L. (2017) 'Trends in Data Centre Energy Consumption under the European Code of Conduct for Data Centre Energy Efficiency', *Energies*, 10(10). doi: 10.3390/en10101470.

Awada, U. and Barker, A. (2017) 'Improving Resource Efficiency of Container-Instance Clusters on Clouds', in *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE, pp. 929–934. doi: 10.1109/CCGRID.2017.113.

Bailey, J. T. and Heidt, S. R. (2003) *Why is Total Cost of Ownership (TCO) Important?* Available at: darwinmag.com.

Bantz, D. F., Bisdikian, C., *et al.* (2003) 'Autonomic personal computing', *IBM Systems Journal*, 42(1), pp. 165–176. doi: 10.1147/sj.421.0165.

Bar-Yam, Y. (1997) 'General Features of Complex Systems', *Knowledge Management, Organisational Intelligence and Learning and Complexity*, I(1), pp. 1–10.

Barroso, L. and Hölzle, U. (2013) 'The Datacenter as a Computer', *Morgan & Claypool Publishers (May 2009)*, 24, p. 156. doi: 10.2200/S00516ED2V01Y201306CAC024.

Bause, F. and Dortmund, U. (1993) 'Queueing Petri Nets A Formalism for the Combined Qualitative and Quantitative Analysis of Systems', pp. 14–23.

Becker, M. (2016) *SimuLizar - SDQ Wiki*. Available at: https://sdqweb.ipd.kit.edu/wiki/SimuLizar (Accessed: 15 August 2017).

Becker, M., Becker, S. and Meyer, J. (2013) 'SimuLizar: Design-Time Modeling and Performance Analysis of Self-Adaptive Systems', *Software Engineering*, 213, pp. 71–

84.

Becker, M., Luckey, M. and Becker, S. (2013) ‘Performance analysis of self-adaptive systems for requirements validation at design-time’, *Proceedings of the 9th international ACM Sigsoft conference on Quality of software architectures - QoSA ’13*. New York, New York, USA: ACM Press, p. 43. doi: 10.1145/2465478.2465489.

Behavior, S. E., Copil, G., *et al.* (2011) ‘ADVISE – a Framework for Evaluating Cloud’.

Bell, W. H., Cameron, D. G., *et al.* (2002) ‘Simulation of dynamic grid replication strategies in optorsim’, *IEEE Workshop on Grid Computing (Grid’2002)*, pp. 46–57. doi: 10.1007/3-540-36133-2.

Beloglazov, A. and Buyya, R. (2012) ‘Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers’, *Concurrency Computation Practice and Experience*, 24(13), pp. 1397–1420. doi: 10.1002/cpe.1867.

Beloglazov, A. and Buyya, R. (2014) ‘OpenStack Neat: a framework for dynamic and energy-efficient consolidation of virtual machines in OpenStack clouds’, *Concurrency and Computation: Practice and Experience*, 27(5), pp. 1310–1333. doi: 10.1002/cpe.3314.

Biran, O., Corradi, A., *et al.* (2012) ‘A stable network-aware VM placement for cloud systems’, *Proceedings - 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2012*, pp. 498–506. doi: 10.1109/CCGrid.2012.119.

Bobroff, N., Kochut, A. and Beaty, K. (2007) ‘Dynamic placement of virtual machines for managing SLA violations - Wagner’, *10th IFIP/IEEE International Symposium on Integrated Network Management 2007, IM ’07*, 5, pp. 119–128. doi: 10.1109/INM.2007.374776.

Bohn, R. B., Messina, J., *et al.* (2011) ‘NIST cloud computing reference architecture’, *Proceedings - 2011 IEEE World Congress on Services, SERVICES 2011*, pp. 594–596. doi: 10.1109/SERVICES.2011.105.

Bousselmi, K., Brahmi, Z. and Gammoudi, M. M. (2014) ‘Cloud services orchestration: A comparative study of existing approaches’, *Proceedings - 2014 IEEE 28th International Conference on Advanced Information Networking and Applications Workshops, IEEE WAINA 2014*, pp. 410–416. doi: 10.1109/WAINA.2014.72.

Bowden, R. (1998) ‘The spectrum of simulation software’, *Iie Solutions*, pp. 44–46.

Brosch, F., Koziolek, H., *et al.* (2012) ‘Architecture-based reliability prediction with the palladio component model’, *IEEE Transactions on Software Engineering*, 38(6), pp. 1319–1339. doi: 10.1109/TSE.2011.94.

Brown, D. J. (1979) ‘A Lower Bound for On-line One-dimensional Bin Packing Algorithms’.

Buyya, R., Yeo, C. S., *et al.* (2009) ‘Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility’, *Future Generation Computer Systems*. Elsevier B.V., 25(6), pp. 599–616. doi: 10.1016/j.future.2008.12.001.

Buyya, R. and Murshed, M. (2002) ‘Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing’, *Concurrency and computation: practice …*, pp. 1–37.

Buyya, R., Ranjan, R. and Calheiros, R. N. (2009) ‘Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities’, *2009 International Conference on High Performance Computing & Simulation*. Ieee, pp. 1–11. doi: 10.1109/HPCSIM.2009.5192685.

Buyya, R., Vecchiola, C. and Selvi, S. T. (2013) ‘Mastering Cloud Computing’, in *Mastering Cloud Computing*. Elsevier, pp. 3–27. doi: 10.1016/B978-0-12-411454-8.00001-2.

Byrne, J., Svorobej, S., *et al.* (2017) ‘RECAP simulator: Simulation of cloud/edge/fog computing scenarios’, in *2017 Winter Simulation Conference (WSC)*. IEEE, pp. 4568–4569. doi: 10.1109/WSC.2017.8248208.

Calheiros, R. N., Netto, M. A. S., *et al.* (2012) ‘EMUSIM: An Integrated Emulation and Simulation Environment for Modeling, Evaluation, and Validation of Performance of Cloud Computing Applications’, *Software - Practice and Experience*, 39(7), pp. 661–699. doi: 10.1002/spe.

Calheiros, R. N., Masoumi, E., *et al.* (2015) ‘Workload prediction using ARIMA model and its impact on cloud applications’ QoS’, *IEEE Transactions on Cloud Computing*, 3(4), pp. 449–458. doi: 10.1109/TCC.2014.2350475.

Calheiros, R. N., Ranjan, R. and Buyya, R. (2011) 'Virtual Machine Provisioning Based on Analytical Performance and QoS in Cloud Computing Environments', in *2011 International Conference on Parallel Processing*. IEEE, pp. 295–304. doi: 10.1109/ICPP.2011.17.

Calheiros, R. and Ranjan, R. (2011) 'CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms', *Software: Practice …*, (August 2010), pp. 23–50. doi: 10.1002/spe.

Carrozza, G., Loffreda, M. and Manetti, V. (2012) 'Exploiting cloud computing for enabling distributed testing of complex systems: The SELEX-SI roadmap', *Proceedings - 2012 7th International Conference on System of Systems Engineering, SoSE 2012*. IEEE, pp. 350–355. doi: 10.1109/SYSoSE.2012.6384142.

Castañé, G. G., Núñez, A., *et al.* (2013) 'E-mc2: A formal framework for energy modelling in cloud computing', *Simulation Modelling Practice and Theory*. Elsevier B.V., 39, pp. 56–75. doi: 10.1016/j.simpat.2013.05.002.

Catteddu, D. and Hogben, G. (2010) 'Cloud Computing: Benefits, Risks and Recommendations for Information Security', in *Computing*, pp. 17–17. doi: 10.1007/978-3-642-16120-9_9.

Chaisiri, S. and Niyato, D. (2009) 'Optimal virtual machine placement across multiple cloud providers', *2009 IEEE Asia-Pacific Services Computing Conference (APSCC)*. Ieee, pp. 103–110. doi: 10.1109/APSCC.2009.5394134.

Chang, Y.-S., Lee, Y.-K., *et al.* (2013) 'Cost Evaluation on Building and Operating Cloud Platform', *International Journal of Grid and High Performance Computing*. IGI Global, 5(2), pp. 43–53. doi: 10.4018/jghpc.2013040103.

Chen, C. J., Liu, Y. S. and Chang, R. G. (2012) 'DCSim: Design Analysis on Virtualization Data Center', in *In Ubiquitous Intelligence & Computing and 9th International Conference on Autonomic & Trusted Computing (UIC/ATC)*. IEEE. doi: 10.1109/UIC-ATC.2012.66.

Chen, Q., Chen, J., *et al.* (2015) 'Utilization-based VM consolidation scheme for power efficiency in cloud data centers', in *2015 IEEE International Conference on Communication Workshop, ICCW 2015*, pp. 1928–1933. doi: 10.1109/ICCW.2015.7247462.

Chen, W. and Deelman, E. (2012) 'WorkflowSim: A toolkit for simulating scientific workflows in distributed environments', *E-Science (e-Science), 2012 IEEE 8th ….* Available at: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6404430 (Accessed: 8 October 2013).

Chihi, H., Chainbi, W. and Ghedira, K. (2013) 'An energy-efficient self-provisioning approach for cloud resources management', *ACM SIGOPS Operating Systems Review*, 47, pp. 2–9. doi: 10.1145/2553070.2553072.

Choi, Y., Lee, S., *et al.* (2015) 'The method to secure scalability and high density in cloud data-center', *Information Systems*. Elsevier, 48, pp. 274–278. doi: 10.1016/j.is.2014.05.013.

Cisco (2014) 'Cisco Global Cloud Index: Forecast and Methodology, 2013–2018', *Cisco Press*, pp. 2014–2019. Available at: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.pdf.

Cisco (2018) *White paper: Cisco Visual Networking Index: Forecast and Trends, 2017–2022*. Available at: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.pdf.

Clark, C., Fraser, K., *et al.* (2005) 'Live migration of virtual machines', *NSDI'05 Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, (Vmm), pp. 273–286. doi: 10.1145/1251203.1251223.

Corradi, A., Fanelli, M. and Foschini, L. (2014) 'VM consolidation: A real case based on OpenStack Cloud', *Future Generation Computer Systems*. Elsevier B.V., 32, pp. 118–127. doi: 10.1016/j.future.2012.05.012.

Corrado, A. J. (2019) *Dynamics of complex systems*. CRC Press.

Coutinho, E. F., De Carvalho Sousa, F. R., *et al.* (2014) *Elasticity in cloud computing: a survey*, *Annales des Telecommunications/Annals of Telecommunications*. doi: 10.1007/s12243-014-0450-7.

Dabbagh, M., Hamdaoui, B., *et al.* (2015) 'Energy-Efficient Resource Allocation and Provisioning Framework for Cloud Data Centers', *IEEE Transactions on Network and Service Management*, 12(3), pp. 377–391. doi: 10.1109/TNSM.2015.2436408.

David, H., Gorbatov, E., *et al.* (2010) ‘RAPL: Memory power estimation and capping’, *Low-Power Electronics and Design (ISLPED), 2010 ACM/IEEE International Symposium on*, pp. 189–194. doi: 10.1145/1840845.1840883.

Dillon, T., Wu, C. W. C. and Chang, E. (2010) ‘Cloud Computing: Issues and Challenges’, *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pp. 27–33. doi: 10.1109/AINA.2010.187.

Drury, D. H. (2001) ‘Determining IT TCO: Lessons and extensions.’, *9th ECIS*, pp. 825–836. Available at: http://is2.lse.ac.uk/asp/aspecis/20010030.pdf.

Duan, Y., Fu, G., *et al.* (2015) ‘Everything as a Service (XaaS) on the Cloud : Origins , Current and Future Trends’, pp. 621–628. doi: 10.1109/CLOUD.2015.88.

Dupont, S., Lejeune, J., *et al.* (2015) ‘Experimental Analysis on Autonomic Strategies for Cloud Elasticity’, *International Conference on Cloud and Autonomic Computing (ICCAC)*, pp. 81–92. doi: 10.1109/ICCAC.2015.22.

Eclipse Foundation (2017) *Eclipse CDO Model Repository | projects.eclipse.org*. Available at: https://projects.eclipse.org/projects/modeling.emf.cdo (Accessed: 23 August 2017).

Eclipse Foundation (2018) *Eclipse Modeling Project*. Available at: https://eclipse.org/modeling/emf/ (Accessed: 23 August 2017).

EIA (2015) *Frequently Asked Questions: How much electricity does an American home use ?* Available at: www.eia.gov/tools/faqs/faq.cfm?id=97&t=3 (Accessed: 11 June 2017).

El-Kassabi, H., Serhani, M. A., *et al.* (2018) ‘Cloud Workflow Resource Shortage Prediction and Fulfillment Using Multiple Adaptation Strategies’, in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. IEEE, pp. 974–977. doi: 10.1109/CLOUD.2018.00149.

Emeakaroha, V. C., Fatema, K., *et al.* (2016) ‘A Trust Label System for Communicating Trust in Cloud Services’, *IEEE Transactions Journal*, XX(XX), pp. 1–13. doi: 10.1109/TSC.2016.2553036.

European Commission (2007) ‘FP7 in Brief: How to get involved in the Framework Programme for Research’, pp. 1–32. Available at:

https://ec.europa.eu/research/fp7/pdf/fp7-inbrief_en.pdf.

European Commission (2013) ‘Work programme for the ICT theme’, (June).

European Commission (2015) *Press - FP7 - Research - Europa*. Available at: https://ec.europa.eu/research/fp7/index_en.cfm?pg=press (Accessed: 21 June 2017).

European Union (2017) *CORDIS: Project Details*. Available at: http://cordis.europa.eu/project/rcn/110456_en.html.

Eurostat (2016) *Eurostat*. Available at: http://appsso.eurostat.ec.europa.eu/nui/show.do?dataset=isoc_cicce_use&lang=en# (Accessed: 6 June 2017).

Fan, X., Weber, W.-D. and Barroso, L. A. (2007) ‘Power provisioning for a warehouse-sized computer’, *ACM SIGARCH Computer Architecture News*, 35(2), p. 13. doi: 10.1145/1273440.1250665.

Fernandes, D. A. B., Soares, L. F. B., *et al.* (2014) ‘Security issues in cloud environments: A survey’, *International Journal of Information Security*, 13(2), pp. 113–170. doi: 10.1007/s10207-013-0208-7.

FIFA (2016) ‘Laws of the game’. Available at: https://www.fifa.com/mm/Document/FootballDevelopment/Refereeing/02/36/01/11/LawsofthegamewebEN_Neutral.pdf.

Filho, M. C. S., Oliveira, R. L., *et al.* (2017) ‘CloudSim Plus: A cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness’, *Proceedings of the IM 2017 - 2017 IFIP/IEEE International Symposium on Integrated Network and Service Management*, (i), pp. 400–406. doi: 10.23919/INM.2017.7987304.

Fittkau, F., Frey, S. and Hasselbring, W. (2012) ‘CDOSim: Simulating cloud deployment options for software migration support’, *2012 IEEE 6th International Workshop on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA)*. Ieee, pp. 37–46. doi: 10.1109/MESOCA.2012.6392599.

Flexiant (2017) *Flexiant Cloud Orchestrator - Software Features Tour*. Available at: https://www.flexiant.com/flexiant-cloud-orchestrator/ (Accessed: 27 June 2017).

Fonseca, C. M. and Fleming, P. J. (1995) ‘An Overview of Evolutionary Algorithms in

Multiobjective Optimization', *Evolutionary Computation*, 3(1), pp. 1–16. doi: 10.1162/evco.1995.3.1.1.

Foster, I. and Kesselman, C. (1998) *The Grid: Blueprint for a New Computing Infrastructure*. Edited by I. Foster and C. Kesselman. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Galante, G. and De Bona, L. C. E. (2012) 'A survey on cloud computing elasticity', *Proceedings - 2012 IEEE/ACM 5th International Conference on Utility and Cloud Computing, UCC 2012*, (1), pp. 263–270. doi: 10.1109/UCC.2012.30.

Garg, S. K. and Buyya, R. (2011) 'NetworkCloudSim: Modelling Parallel Applications in Cloud Simulations', in *2011 Fourth IEEE International Conference on Utility and Cloud Computing*. IEEE, pp. 105–113. doi: 10.1109/UCC.2011.24.

Gartner Inc. (2018) *Gartner Forecasts Worldwide Public Cloud Revenue to Grow 21.4 Percent in 2018*, *gartner.com*. doi: 3871416.

Geng, H. (2014) *Data Center Handbook*, *Data Center Handbook*. John Wiley & Sons. doi: 10.1002/9781118937563.

Glassdoor (2018) *Glassdoor Job Search | Find the job that fits your life*. Available at: https://www.glassdoor.com/index.htm (Accessed: 20 July 2018).

Glover, F. (1989) 'Tabu Search - Part I', *ORSA journal on Computing*, 2 1(3), pp. 4–32. doi: 10.1287/ijoc.2.1.4.

Google (2015) *App Engine - Platform as a Service — Google Cloud Platform*. Available at: https://cloud.google.com/appengine/ (Accessed: 12 November 2015).

Google (2016) *Android Emulator | Android Developers*. Available at: https://developer.android.com/tools/help/emulator.html (Accessed: 28 January 2016).

Google (2018) *Data Center Facilities Technician - Google - The Dalles, OR 97058 - Google Careers*. Available at: https://careers.google.com/jobs#!t=jo&jid=/google/data-center-facilities-technician-the-dalles-or-97058-usa-2660013& (Accessed: 20 July 2018).

Goudarzi, H. and Pedram, M. (2011) 'Multi-dimensional SLA-based resource allocation for multi-tier cloud computing systems', *Proceedings - 2011 IEEE 4th International Conference on Cloud Computing, CLOUD 2011*, pp. 324–331. doi:

10.1109/CLOUD.2011.106.

Goudarzi, H. and Pedram, M. (2012) ‘Energy-Efficient Virtual Machine Replication and Placement in a Cloud Computing System’, *2012 IEEE Fifth International Conference on Cloud Computing*. Ieee, pp. 750–757. doi: 10.1109/CLOUD.2012.107.

Greenberg, A., Hamilton, J., *et al.* (2008) ‘The cost of a cloud: research problems in data center networks’, *SIGCOMM Comput. Commun. Rev.*, 39(1), pp. 68–73. doi: 10.1145/1496091.1496103.

Groenda, H., Stier, C., Östberg, P.-O., *et al.* (2014) *CACTOS Project Deliverable 5.1: Model Integration Method and Supporting Tooling*.

Groenda, H., Stier, C., Krzywda, J., *et al.* (2014) ‘Model integration method and supporting tooling: project deliverable D5.1’. Open Access Repositorium der Universität Ulm. doi: 10.18725/OPARU-4317.

Groenda, H. and Stier, C. (2015) ‘Improving IaaS Cloud Analyses by Black-Box Resource Demand Modeling’, *Symposium on Software Performance 2015*, pp. 7–9. Available at: https://sdqweb.ipd.kit.edu/publications/pdfs/groenda2015c.pdf.

Grunewald, D., Bye, R., *et al.* (2011) ‘Agent-based Network Security Simulation (Demonstration)’, *Aamas*, pp. 1325–1326.

Gupta, H., Dastjerdi, A. V., *et al.* (2016) ‘iFogSim : A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things , Edge and Fog’, pp. 1–22. Available at: https://books.google.ie/books?hl=en&lr=&id=g8nlBAAAQBAJ&oi=fnd&pg=PA127&ots=xkQJmprKkA&sig=Xmlk-bzndrutKVygVhCzG4qY40k&redir_esc=y#v=onepage&q&f=false.

Gupta, S. K. S., Banerjee, A., *et al.* (2014) ‘GDCSim : A Simulator for Green Data Center Design and Analysis’, 24(1), pp. 1–27.

Hameed, A., Khoshkbarforoushha, A., *et al.* (2016) ‘A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems’, *Computing*, 98(7), pp. 751–774. doi: 10.1007/s00607-014-0407-8.

Hardy, D., Sideris, I., *et al.* (2011) ‘EETCO: a tool to Estimate and Explore the implications of datacenter design choices on the TCO and the environmental impact’, *in*

Workshop on Energy-efficient Computing for a Sustainable World in conjunction with the 44th Annual IEEE/ACM International Symposium on Microarchitecture (Micro-44), Dec 2011. Available at: http://www.cs.ucy.ac.cy/carch/xi/papers/EESC2011.pdf.

Hardy, D., Kleanthous, M., *et al.* (2013) 'An analytical framework for estimating TCO and exploring data center design space', *ISPASS 2013 - IEEE International Symposium on Performance Analysis of Systems and Software*, pp. 54–63. doi: 10.1109/ISPASS.2013.6557146.

Hariri, S., Khargharia, B., *et al.* (2006) 'The autonomic computing paradigm', *Cluster Computing*, 9(1), pp. 5–17. doi: 10.1007/s10586-006-4893-0.

Hashizume, K., Rosado, D. G., *et al.* (2013) 'An analysis of security issues for cloud computing', *Journal of Internet Services and Applications*, 4(1), p. 5. doi: 10.1186/1869-0238-4-5.

Hauser, C., Domaschka, J., *et al.* (2016) 'Validation goals and metrics: project deliverable D7.3.2'. Open Access Repositorium der Universität Ulm. doi: 10.18725/OPARU-4313.

Higashino, W. A., Capretz, M. A. M. and Bittencourt, L. F. (2015) 'CEPSim : A Simulator for Cloud-Based Complex Event Processing'. doi: 10.1109/BigDataCongress.2015.34.

Horst, R. and Tuy, H. (1996) *Global Optimization*. Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/978-3-662-03199-5.

Huebscher, M. C. and McCann, J. a (2008) 'A survey of autonomic computing—degrees, models, and applications', *ACM Computing Surveys*, 40(3), pp. 1–28. doi: 10.1145/1380584.1380585.

Hwang, E. and Kim, K. H. (2012) 'Minimizing Cost of Virtual Machines for Deadline-Constrained MapReduce Applications in the Cloud', *2012 ACM/IEEE 13th International Conference on Grid Computing*. Ieee, pp. 130–138. doi: 10.1109/Grid.2012.19.

IBM (2005) 'Autonomic Computing White Paper: An Architectural Blueprint for Autonomic Computing', *IBM White Paper*, (June), p. 34. doi: 10.1021/am900608j.

IEEE (2017) *IEEE Xplore digital library*. Available at: http://ieeexplore.ieee.org.

Ilkhechi, A. R., Korpeoglu, I. and Ulusoy, Ö. (2015) 'Network-aware virtual machine

placement in cloud data centers with multiple traffic-intensive components', *Computer Networks*. Elsevier Ltd., 91, pp. 508–527. doi: 10.1016/j.comnet.2015.08.042.

Infoholic Research LLP (2017) *Cloud Data Center Market - Global Drivers, Restraints, Opportunities, Trends, and Forecasts to 2023*. Available at: https://www.researchandmarkets.com/research/cqrxj7/cloud_data_center.

Intel Corporation (2018) *Intel*. Available at: www.intel.com.

International Institute of Business Analysis (2015) *A Guide to the Business Analysis Body of Knowledge (BABOK Guide)*. 3rd edn, *IIBA*. 3rd edn. Available at: http://www.innovativeprojectguide.com/documents/BABOK_Guide_v3_Member.pdf.

Investopedia (2019) *Time Value of Money (TVM)*. Available at: https://www.investopedia.com/terms/t/timevalueofmoney.asp (Accessed: 7 July 2019).

Iorga, M., Feldman, L., *et al.* (2017) 'Draft SP 800-191, The NIST Definition of Fog Computing', *NIST Special Publication*, 800(March). doi: 10.6028/NIST.SP.800-145.

Jamsa, K. (2013) *Cloud Computing: SaaS, PaaS, IaaS, Virtualization, Business Models, Mobile, Security and More*. Jones & Bartlett Publishers.

Jararweh, Y., Alshara, Z., *et al.* (2012) 'TeachCloud: a cloud computing educational toolkit', *International Journal of Cloud Computing*, 2(2012), pp. 237–257. doi: 10.1504/IJCC.2013.055269.

Jararweh, Y., Jarrah, M., *et al.* (2014) 'CloudExp: A comprehensive cloud computing experimental framework', *Simulation Modelling Practice and Theory*. Elsevier B.V., 49, pp. 180–192. doi: 10.1016/j.simpat.2014.09.003.

Jelden, K. E., Domaschka, J., *et al.* (2014) 'Validation goals and metrics: project deliverable D7.3.1'. Open Access Repositorium der Universität Ulm. doi: 10.18725/OPARU-4337.

Jennings, B. and Stadler, R. (2014) 'Resource Management in Clouds: Survey and Research Challenges', *Journal of Network and Systems Management*, pp. 1–53. doi: 10.1007/s10922-014-9307-7.

Jiang, Y., Perng, C. S., *et al.* (2012) 'Self-adaptive cloud capacity planning', *Proceedings - 2012 IEEE 9th International Conference on Services Computing, SCC 2012*, pp. 73–80. doi: 10.1109/SCC.2012.8.

Jing, S.-Y., Ali, S., *et al.* (2013) 'State-of-the-art research study for green cloud computing', *The Journal of Supercomputing*, 65(1), pp. 445–468. doi: 10.1007/s11227-011-0722-1.

Jung, J. and Kim, H. (2012) 'MR-CloudSim: Designing and implementing MapReduce computing model on CloudSim', *International Conference on ICT Convergence*, pp. 504–509. doi: 10.1109/ICTC.2012.6387186.

K. Preston White, J. and Ingalls, R. G. (2011) 'Introduction To Simulation', *Winter Simulation Conference 2011*, pp. 1379–1393. Available at: http://www.informs-sim.org/wsc11papers/124.pdf.

Kathiravelu, P. and Veiga, L. (2014) 'An Adaptive Distributed Simulator for Cloud and MapReduce Algorithms and Architectures'.

Kaur, P. D. and Chana, I. (2014) 'A resource elasticity framework for QoS-aware execution of cloud applications', *Future Generation Computer Systems*. Elsevier B.V., 37, pp. 14–25. doi: 10.1016/j.future.2014.02.018.

Kecskemeti, G. (2015) 'DISSECT-CF : A simulator to foster energy-aware scheduling in infrastructure clouds'. Elsevier B.V., 58, pp. 188–218. doi: 10.1016/j.simpat.2015.05.009.

Kelton, D. and Law, A. (2000) *Simulation modelling and analysis*. 3rd edn. New York: McGraw-Hill Higher Education.

Kephart, J. O. and Chess, D. M. (2003) 'The vision of autonomic computing', *Computer*, 36(1), pp. 41–50. doi: 10.1109/MC.2003.1160055.

Khajeh-Hosseini, A., Greenwood, D. and Sommerville, I. (2010) 'Cloud migration: A case study of migrating an enterprise IT system to IaaS', *Proceedings - 2010 IEEE 3rd International Conference on Cloud Computing, CLOUD 2010*, pp. 450–457. doi: 10.1109/CLOUD.2010.37.

Khanna, G., Beaty, K., *et al.* (2006) 'Application Performance Management in Virtualized Server Environments', *2006 IEEEIFIP Network Operations and Management Symposium NOMS 2006*, 20(D), pp. 373–381. doi: 10.1109/NOMS.2006.1687567.

Kim, H., Han, J., *et al.* (2017) 'IsV2C: An Integrated Road Traffic-Network-Cloud

Simulator for V2C Connected Car Services', *Proceedings - 2017 IEEE 14th International Conference on Services Computing, SCC 2017*, pp. 434–441. doi: 10.1109/SCC.2017.62.

Kliazovich, D., Bouvry, P., *et al.* (2010) 'GreenCloud: A Packet-Level Simulator of Energy-Aware Cloud Computing Data Centers', *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*. Ieee, pp. 1–5. doi: 10.1109/GLOCOM.2010.5683561.

Krauter, K., Buyya, R. and Maheswaran, M. (2002) 'A taxonomy and survey of grid resource management systems for distributed computing', *Software: Practice and Experience*, 32(2), pp. 135–164. doi: 10.1002/spe.432.

Krishnaswamy, D., Krishnan, R., *et al.* (2015) 'An open NFV and cloud architectural framework for managing application virality behaviour', *2015 12th Annual IEEE Consumer Communications and Networking Conference, CCNC 2015*, pp. 746–754. doi: 10.1109/CCNC.2015.7158071.

Krzywda, J., Ali-Eldin, A., *et al.* (2014) 'Prototype optimisation model: project deliverable D3.1'. Open Access Repositorium der Universität Ulm. doi: 10.18725/OPARU-4305.

Krzywda, J., Rezaie, A., *et al.* (2015) 'Extended optimization model: project deliverable D3.3'. Open Access Repositorium der Universität Ulm. doi: 10.18725/OPARU-4307.

Kumar, J., Malik, A., *et al.* (2017) 'Demand-Based Computation Offloading Framework for Mobile Devices', *IEEE Systems Journal*, pp. 1–10. doi: 10.1109/JSYST.2017.2706178.

Kumar, S., Talwar, V., *et al.* (2011) 'Loosely coupled coordinated management in virtualized data centers', *Cluster Computing*, 14(3), pp. 259–274. doi: 10.1007/s10586-010-0124-9.

Lango, J. (2014) 'Toward software-defined SLAs', *Communications of the ACM*, 57(1), pp. 54–60. doi: 10.1145/2541883.2541894.

Lechler, T. and Page, B. (1999) 'DESMO-J: An object oriented discrete simulation framework in Java', in *Proceedings of the 11th European Simulation Symposium*, pp. 46–50.

Lee, K. and Park, S. (2015) ‘A CPU Overhead-Aware VM Placement Algorithm for Network Bandwidth Guarantee in Virtualized Data Centers’, *2015 International Conference on Cloud and Autonomic Computing*, pp. 268–274. doi: 10.1109/ICCAC.2015.40.

Lee, S., Prabhakaran, R. P. V., *et al.* (2010) ‘Validating Heuristics for Virtual Machines Consolidation’, *Research.Microsoft.Com*, pp. 81–97. Available at: http://research.microsoft.com/pubs/144571/virtualization.pdf.

Legrand, a., Marchal, L. and Casanova, H. (2003) ‘Scheduling distributed applications: the SimGrid simulation framework’, *CCGrid 2003. 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid, 2003. Proceedings.*, pp. 1–8. doi: 10.1109/CCGRID.2003.1199362.

Legrand, I. C. and Newman, H. B. (2000) ‘The MONARC toolset for simulating large network-distributed processing systems’, *2000 Winter Simulation Conference Proceedings (Cat. No.00CH37165)*, 2. doi: 10.1109/WSC.2000.899171.

Li, K., Xu, G., *et al.* (2011) ‘Cloud Task Scheduling Based on Load Balancing Ant Colony Optimization’, *2011 Sixth Annual Chinagrid Conference*, pp. 3–9. doi: 10.1109/ChinaGrid.2011.17.

Li, Q. and Guo, Y. (2011) ‘Optimization of resource scheduling in cloud computing’, *Proceedings - 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2010*, (1), pp. 315–320. doi: 10.1109/SYNASC.2010.8.

Li, X., Li, Y., *et al.* (2009) ‘The Method and Tool of Cost Analysis for Cloud Computing’, *2009 IEEE International Conference on Cloud Computing*, 978-0–7695, pp. 93–100. doi: 10.1109/CLOUD.2009.84.

Li, X., Jiang, X., *et al.* (2012) ‘DARTCSIM : AN ENHANCED USER-FRIENDLY CLOUD SIMULATION SYSTEM BASED ON CLOUDSIM WITH’, *IEEE CCIS2012*.

Li, X., Li, K., *et al.* (2017) ‘An Orchestration Based Cloud Auto-Healing Service Framework’, in *2017 IEEE International Conference on Edge Computing (EDGE)*. IEEE, pp. 190–193. doi: 10.1109/IEEE.EDGE.2017.33.

Li, Y. and Lan, Z. (2005) ‘A survey of load balancing in grid computing’, *Computational and Information science*, pp. 280–285. doi: 10.1007/978-3-540-30497-5_44.

Lim, S. H., Sharma, B., *et al.* (2009) 'MDCSim: A multi-tier data center simulation platform', *Proceedings - IEEE International Conference on Cluster Computing, ICCC*. doi: 10.1109/CLUSTR.2009.5289159.

Lin, J.-W., Chen, C.-H. and Lin, C.-Y. (2014) 'Integrating QoS awareness with virtualization in cloud computing systems for delay-sensitive applications', *Future Generation Computer Systems*. Elsevier B.V., 37, pp. 478–487. doi: 10.1016/j.future.2013.12.034.

Liu, C., Mao, Y., Van der Merwe, J. E., *et al.* (2011) 'Cloud Resource Orchestration: A Data-Centric Approach', *Proceedings of the …*, pp. 241–248. doi: 10.1.1.190.86.

Liu, C., Mao, Y., Chen, X., *et al.* (2011) 'T ROPIC : Transactional Resource Orchestration Platform In the Cloud', (April).

Liu, C., Loo, B. T. and Mao, Y. (2011) 'Declarative automated cloud resource orchestration', pp. 26:1–26:8. doi: 10.1145/2038916.2038942.

Liu, L., Wang, H., *et al.* (2009) 'GreenCloud: a new architecture for green data center', *Proceedings of the 6th international conference industry session on Autonomic computing and communications industry session. ACM,* pp. 29–38.

Long, W., Yuqing, L. and Qingxin, X. (2013) 'Using CloudSim to Model and Simulate Cloud Computing Environment', *2013 Ninth International Conference on Computational Intelligence and Security*. Ieee, pp. 323–328. doi: 10.1109/CIS.2013.75.

Louis, B. (2015) *CloudSimDisk : Energy-Aware Storage Simulation in CloudSim*.

Lu, M. and Zhou, X. (2018) 'A big data on private cloud agile provisioning framework based on OpenStack', in *2018 IEEE 3rd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*. IEEE, pp. 253–260. doi: 10.1109/ICCCBDA.2018.8386522.

Lycette, B. and Lowenstein, D. (2010) 'The Real "Total Cost of Ownership" of Your Test Equipment', *Proceedings of the 2010 IEEE AUTOTESTCON*, p. 5 pp.

Maguire, D. (2008) *The business benefits of GIS : an ROI approach*. ESRI Press.

Maguluri, S. T., Srikant, R. and Ying, L. (2012) 'Stochastic models of load balancing and scheduling in cloud computing clusters', *2012 Proceedings IEEE INFOCOM*. Ieee, pp. 702–710. doi: 10.1109/INFCOM.2012.6195815.

Mahmoudi, C., Mourlin, F. and Battou, A. (2018) 'Formal definition of edge computing: An emphasis on mobile cloud and IoT composition', *2018 Third International Conference on Fog and Mobile Edge Computing (FMEC)*, pp. 34–42. doi: 10.1109/FMEC.2018.8364042.

Mak, W. K., Morton, D. P. and Wood, R. K. (1999) 'Monte Carlo bounding techniques for determining solution quality in stochastic programs', *Operations Research Letters*, 24(1), pp. 47–56. doi: 10.1016/S0167-6377(98)00054-6.

Makaratzis, A. T., Giannoutakis, K. M. and Tzovaras, D. (2018) 'Energy modeling in cloud simulation frameworks', *Future Generation Computer Systems*. Elsevier B.V., 79, pp. 715–725. doi: 10.1016/j.future.2017.06.016.

Makrani, H. M., Sayadi, H., *et al.* (2018) 'Energy-aware and Machine Learning-based Resource Provisioning of In-Memory Analytics on Cloud', *Proceedings of the ACM Symposium on Cloud Computing - SoCC '18*, pp. 517–517. doi: 10.1145/3267809.3275459.

Malik, A., Bilal, K., *et al.* (2015) 'CloudNetSim++: A GUI Based Framework for Modeling and Simulation of Data Centers in OMNeT++', *IEEE Transactions on Services Computing*, 1374(c), pp. 1–1. doi: 10.1109/TSC.2015.2496164.

Manvi, S. S. and Krishna Shyam, G. (2014) 'Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey', *Journal of Network and Computer Applications*. Elsevier, 41(1), pp. 424–440. doi: 10.1016/j.jnca.2013.10.004.

Marinescu, D. C. (2016) *Complex systems and clouds: a self-organization and self-management perspective*. Morgan Kaufmann.

Markets and Markets (2016) *Cloud Services Brokerage and Enablement Market by Component (Cloud Services Brokerage and Cloud Brokerage Enablement), by Organization Size, by Industry Vertical & by Region - Global Forecast to 2021*. Available at: http://www.researchandmarkets.com/reports/3775384/cloud-services-brokerage-and-enablement-market-by (Accessed: 17 April 2017).

Marston, S., Li, Z., *et al.* (2011) 'Cloud computing — The business perspective', *Decision Support Systems*, 51(1), pp. 176–189. Available at: http://www.sciencedirect.com/science/article/pii/S0167923610002393 (Accessed: 9 July 2014).

Mechtri, M., Ghribi, C., *et al.* (2017) 'ETSO: End-To-End SFC Orchestration Framework', in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, pp. 903–904. doi: 10.23919/INM.2017.7987405.

Mehmi, S., Sangal, A. L., *et al.* (2016) 'Economic Viability of Smart Grid Cloud in India', *International Journal of Grid and Distributed Computing*, 9(2), pp. 61–72. doi: 10.14257/ijgdc.2016.9.2.07.

Mell, P. and Grance, T. (2011) 'The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology', *National Institute of Standards and Technology, Information Technology Laboratory*, 145, p. 7. doi: 10.1136/emj.2010.096966.

Mellanox Technologies (2018) *Ethernet Switches - Mellanox Technologies*. Available at: http://www.mellanox.com/ethernet/switches.php (Accessed: 20 July 2018).

Metsch, T., Ibidunmoye, O., *et al.* (2015) 'Apex Lake: A Framework for Enabling Smart Orchestration', *Proceedings of the Industrial Track of the 16th International Middleware Conference*, pp. 1:1--1:7. doi: 10.1145/2830013.2830016.

Microsoft (2016a) *Microsoft Azure: Cloud Computing Platform & Services*. Available at: https://azure.microsoft.com/en-us/ (Accessed: 12 November 2015).

Microsoft (2016b) *White Paper: Why Hyper-V?* | *Why Microsoft*. Available at: http://www.whymicrosoft.com/see-why/why-hyper-v/ (Accessed: 28 January 2016).

Mistry, S., Bouguettaya, A., *et al.* (2015) 'Service-Oriented Computing', 9435, pp. 333–342. doi: 10.1007/978-3-662-48616-0.

Mitchell, J. (2017) *Top 10 Largest Data Centers in the World*. Available at: https://www.racksolutions.com/news/data-center-news/top-10-largest-data-centers-world/ (Accessed: 11 June 2017).

Moore, L., Bean, K. and Ellahi, T. (2013) 'A Coordinated Reactive and Predictive Approach to Cloud Elasticity', *Cloud Computing 2013*, (c). Available at: http://www.thinkmind.org/index.php?view=article&articleid=cloud_computing_2013_4_20_20112.

Al Morsy, M., Grundy, J. and Müller, I. (2010) 'An analysis of the cloud computing security problem', *17th Asia-Pacific Software Engineering Conference (APSEC 2010)*

Cloud Workshop,Sydney, Australia, (December), p. 7. doi: arXiv:1609.01107.

Muñoz, A., Gonzalez, J. and Maña, A. (2012) 'A performance-oriented monitoring system for security properties in cloud computing applications', *Computer Journal*, 55(8), pp. 979–994. doi: 10.1093/comjnl/bxs042.

Nadjaran Toosi, A., Sinnott, R. O. and Buyya, R. (2018) 'Resource provisioning for data-intensive applications with deadline constraints on hybrid clouds using Aneka', *Future Generation Computer Systems*. Elsevier B.V., 79, pp. 765–775. doi: 10.1016/j.future.2017.05.042.

Nita, M. C., Pop, F., *et al.* (2014) 'FIM-SIM: Fault injection module for CloudSim based on statistical distributions', *Journal of Telecommunications and Information Technology*, 2014(4), pp. 14–23.

NS-2 (2014) *nsnam*. Available at: http://nsnam.sourceforge.net/wiki/index.php/Main_Page (Accessed: 10 July 2017).

Núñez, A., Vázquez-Poletti, J. L., *et al.* (2012) 'iCanCloud: A Flexible and Scalable Cloud Infrastructure Simulator', *Journal of Grid Computing*, 10(1), pp. 185–209. doi: 10.1007/s10723-012-9208-5.

O'Brien, C. (2016) *Facebook data centre to generate 2,000 construction jobs*, *Irishtimes*. Available at: https://www.irishtimes.com/business/technology/facebook-data-centre-to-generate-2-000-construction-jobs-1.2508628 (Accessed: 20 July 2018).

Open Stack (2017) *OpenStack Open Source Cloud Computing Software*. Available at: https://www.openstack.org/ (Accessed: 27 June 2017).

OpenSim (2015) *OMNeT++ Discrete Event Simulator - Home*. Available at: https://omnetpp.org/ (Accessed: 25 November 2015).

OpenStack (2017) *OpenStack Open Source Cloud Computing Software*. Available at: https://www.openstack.org/ (Accessed: 13 April 2015).

Ostberg, P.-O., Groenda, H., *et al.* (2014) 'The CACTOS Vision of Context-Aware Cloud Topology Optimization and Simulation', *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*, pp. 26–31. doi: 10.1109/CloudCom.2014.62.

Ostermann, S., Plankensteiner, K., *et al.* (2011a) 'GroudSim: An event-based

simulation framework for computational grids and clouds', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6586 LNCS(261585), pp. 305–313. doi: 10.1007/978-3-642-21878-1_38.

Ostermann, S., Plankensteiner, K., *et al.* (2011b) 'GroudSim: An event-based simulation framework for computational grids and clouds', in *Proceedings of the 2010 conference on Parallel processing (Euro-Par 2010)*. Springer Berlin Heidelberg, pp. 305–313. doi: 10.1007/978-3-642-21878-1_38.

Palladio (2017) *Palladio Software Architecture Simulator: Home*. Available at: http://www.palladio-simulator.com/home/ (Accessed: 15 August 2017).

Pandi, M. and Karthick, A. V. (2016) 'An Overview of Cost Provisioning Strategies for Cloud Computing', *International Journal of Advanced Research in IT and Engineering*, 5(3).

Papazachos, Z., Bharbuiya, S., *et al.* (2015) 'Data collection framework: project deliverable D4.1, revision 2'. Open Access Repositorium der Universität Ulm. doi: 10.18725/OPARU-4323.

Pardeshi, V. H. (2014) 'Cloud Computing for Higher Education Institutes: Architecture, Strategy and Recommendations for Effective Adaptation', *Symbiosis Institute of Management Studies Annual Research Conference (SIMSARC13), December 12-13, 2013, Pune, India*. Elsevier B.V., 11(14), pp. 589–599. doi: 10.1016/S2212-5671(14)00224-X.

PassMark Software (2018) *PassMark Software - CPU Benchmark Charts*. Available at: https://www.cpubenchmark.net/ (Accessed: 20 July 2018).

Patel, C. D. and Shah, A. J. (2005) 'Cost Model for Planning , Development and Operation of a Data Center', *Development*, 107, pp. 1–36. Available at: http://www.hpl.hp.com/techreports/2005/HPL-2005-107R1.pdf.

Patelli, E., Feng, G., *et al.* (2017) 'Simulation methods for system reliability using the survival signature', *Reliability Engineering and System Safety*. Elsevier Ltd, 167(June), pp. 327–337. doi: 10.1016/j.ress.2017.06.018.

Pham, C., Tran, H. D., *et al.* (2015) 'A General and Practical Consolidation Framework in CloudNFV', pp. 295–300.

Pittl, B. and Schikuta, E. (2016) ‘Bazaar-Extension : A CloudSim Extension for Simulating Negotiation based Resource Allocations’. doi: 10.1109/SCC.2016.62.

PRNewswire LLC (2017) *Switch TAHOE RENO Now Open: Largest, Most Advanced Data Center Campus in the World*. Available at: https://www.prnewswire.com/news-releases/switch-tahoe-reno-now-open-largest-most-advanced-data-center-campus-in-the-world-300407648.html (Accessed: 21 November 2018).

Quang-Hung, N. and Thoai, N. (2015) ‘EMinRET: Heuristic for Energy-Aware VM Placement with Fixed Intervals and Non-preemption’, *The International Conference on Advanced Computing and Applications (ACOMP)*, p. 8. Available at: http://arxiv.org/abs/1511.06825.

Quarati, A., Clematis, A. and D’Agostino, D. (2016) ‘Delivering cloud services with QoS requirements: Business opportunities, architectural solutions and energy-saving aspects’, *Future Generation Computer Systems*. Elsevier B.V., 55, pp. 403–427. doi: 10.1016/j.future.2015.02.009.

Raghavendra, G. S. and Krishna, P. R. (2015) ‘Innovation in IT sector and future advances in cloud computing’, *2015 International Conference on Signal Processing and Communication Engineering Systems*, pp. 389–390. doi: 10.1109/SPACES.2015.7058291.

Ragusa, C., Robinson, P. and Svorobej, S. (2013) ‘A Framework for Modeling and Execution of Infrastructure Contention Experiments’, *2nd Internation Workshop on Measurement-based Experimental Research, Methodology and Tools, MERMAT, FIA*. Available at: http://atc.udg.edu/MERMAT/papers/paper_3_Carmelo_Ragusa_et_al.pdf.

Rajasekar, S., Philominathan, P. and Chinnathambi, V. (2014) ‘Research Methodology’, p. 53. Available at: https://arxiv.org/pdf/physics/0601009.pdf.

Ramchand, K., Chhetri, M. B. and Kowalczyk, R. (2018) ‘Towards A Comprehensive Cloud Decision Framework with Financial Viability Assessment’, *PACIS 2018 Proceedings*. Available at: https://aisel.aisnet.org/pacis2018/32.

Rathfelder, C. and Klatt, B. (2011) ‘Palladio Workbench: A Quality-Prediction Tool for Component-Based Architectures’, *2011 Ninth Working IEEE/IFIP Conference on Software Architecture*. Ieee, pp. 347–350. doi: 10.1109/WICSA.2011.55.

RedHat (2015) *WildFly*. Available at: http://wildfly.org/about/ (Accessed: 12 November

2015).

Reussner, R., Becker, S. and Happe, J. (2011) 'The Palladio Component Model'. doi: 10.5445 / IR / 1000022503.

Rimal, B. P., Jukan, A., *et al.* (2011) 'Architectural Requirements for Cloud Computing Systems: An Enterprise Cloud Approach', *Journal of Grid Computing*, 9(1), pp. 3–26. doi: 10.1007/s10723-010-9171-y.

Rochwerger, B., Vázquez, C., *et al.* (2011) 'An Architecture for Federated Cloud Computing', in *Cloud Computing*. Hoboken, NJ, USA: John Wiley & Sons, Inc., pp. 391–411. doi: 10.1002/9780470940105.ch15.

Rossi, F., Van Beek, P. and Walsh, T. (2006) 'Constraint programming', in *Elsevier*.

Sá, T. T., Calheiros, R. N. and Gomes, D. G. (2014) 'CloudReports: An Extensible Simulation Tool for Energy-Aware Cloud Computing Environments', in *Cloud Computing*. Springer, pp. 127–142. Available at: https://books.google.ie/books?hl=en&lr=&id=g8nlBAAAQBAJ&oi=fnd&pg=PA127&ots=xkQJmprKkA&sig=Xmlk-bzndrutKVygVhCzG4qY40k&redir_esc=y#v=onepage&q&f=false.

Sakellari, G. and Loukas, G. (2013) 'A survey of mathematical models, simulation approaches and testbeds used for research in cloud computing', *Simulation Modelling Practice and Theory*. Elsevier B.V., 39, pp. 92–103. doi: 10.1016/j.simpat.2013.04.002.

Sargent, R. G. (2010) *A New Statistical Procedure for Validation of Simulation and Stochastic Models*.

Savolainen, E. (2012) 'Cloud Service Models Seminar – Cloud Computing and Web Services'.

Sedaghat, M., Hernández-Rodriguez, F. and Elmroth, E. (2016) 'Decentralized cloud datacenter reconsolidation through emergent and topology-aware behavior', *Future Generation Computer Systems*. Elsevier B.V., 56, pp. 51–63. doi: 10.1016/j.future.2015.09.023.

Seufert, M., Kwam, B. K., *et al.* (2017) 'Edgenetworkcloudsim: Placement of service chains in edge clouds using networkcloudsim', *2017 IEEE Conference on Network Softwarization: Softwarization Sustaining a Hyper-Connected World: en Route to 5G,*

NetSoft 2017. doi: 10.1109/NETSOFT.2017.8004247.

Shams, K. S., Powell., M. W., *et al.* (2010) 'Polyphony: A workflow orchestration framework for cloud computing', *CCGrid 2010 - 10th IEEE/ACM International Conference on Cluster, Cloud, and Grid Computing*, pp. 606–611. doi: 10.1109/CCGRID.2010.117.

Sharma, U., Shenoy, P., *et al.* (2011) 'Kingfisher: Cost-aware elasticity in the cloud', *Proceedings - IEEE INFOCOM*, pp. 206–210. doi: 10.1109/INFCOM.2011.5935016.

Shi, W. and Hong, B. (2011) 'Towards Profitable Virtual Machine Placement in the Data Center', *2011 Fourth IEEE International Conference on Utility and Cloud Computing*. Ieee, pp. 138–145. doi: 10.1109/UCC.2011.28.

Simonet, A., Lebre, A. and Orgerie, A.-C. (2016) 'Deploying Distributed Cloud Infrastructures: Who and at What Cost?', in *2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW)*. IEEE, pp. 178–183. doi: 10.1109/IC2EW.2016.48.

Singh, S. and Chana, I. (2015a) 'QoS-Aware Autonomic Resource Management in Cloud Computing', *ACM Computing Surveys*, 48(3), pp. 1–46. doi: 10.1145/2843889.

Singh, S. and Chana, I. (2015b) 'QRSF: QoS-aware resource scheduling framework in cloud computing', *The Journal of Supercomputing*, 71, pp. 241–292. doi: 10.1007/s11227-014-1295-6.

Sinha, U. and Shekhar, M. (2015) 'Comparison of Various Cloud Simulation tools available in Cloud Computing', *INternational Journal of Advanced Research in Computer and Communication Engineering*, 4(3).

Smith, J. E. and Nair, R. (2005) 'The architecture of virtual machines', *Computer*, 38(5), pp. 32–38. doi: 10.1109/MC.2005.173.

Son, J., Dastjerdi, A. V., *et al.* (2015) 'CloudSimSDN: Modeling and Simulation of Software-Defined Cloud Data Centers', in *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE, pp. 475–484. doi: 10.1109/CCGrid.2015.87.

Song, H. J., Liu, X., *et al.* (2000) 'The MicroGrid: a Scientific Tool for Modeling Computational Grids', *ACM/IEEE SC 2000 Conference (SC'00)*, 8(November), pp.

127–141. doi: 10.1109/SC.2000.10028.

Sonmez, C., Ozgovde, A. and Ersoy, C. (2017) 'EdgeCloudSim: An environment for performance evaluation of Edge Computing systems', in *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*. IEEE, pp. 39–44. doi: 10.1109/FMEC.2017.7946405.

Sotiriadis, S., Bessis, N., *et al.* (2013) 'SimIC: Designing a New Inter-cloud Simulation Platform for Integrating Large-Scale Resource Management', in *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*. IEEE, pp. 90–97. doi: 10.1109/AINA.2013.123.

Sousa, E., Lins, F., *et al.* (2015) 'A modeling approach for cloud infrastructure planning considering dependability and cost requirements', *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(4), pp. 549–558. doi: 10.1109/TSMC.2014.2358642.

Sqalli, M. H., Al-saeedi, M., *et al.* (2012) 'UCloud : A Simulated Hybrid Cloud for A University Environment', pp. 170–172.

Sriram, I. (2009) 'SPECI, a simulation tool exploring cloud-scale data centres', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5931 LNCS, pp. 381–392. doi: 10.1007/978-3-642-10665-1_35.

Stier, C. and Groenda, H. (2016) 'Ensuring Model Continuity when Simulating Self-Adaptive Software Systems', *Proceedings of the Modeling and Simulation of Complexity in Intelligent, Adaptive and Autonomous Systems 2016 (MSCIAAS 2016) and Space Simulation for Planetary Space Exploration (SPACE 2016)*, pp. 2:1--2:8. Available at: http://dl.acm.org/citation.cfm?id=2962664.2962666.

Stier, C., Groenda, H. and Koziolek, A. (2014) 'Towards Modeling and Analysis of Power Consumption of Self-Adaptive Software Systems in Palladio', in *SOSP'14 Symposium on Software Performance: Joint Descartes/Kieker/Palladio Days 2014*, pp. 28–45.

STUDENT (1908) 'THE PROBABLE ERROR OF A MEAN', *Biometrika*, 6(1), pp. 1–25. doi: 10.1093/biomet/6.1.1.

Subashini, S. and Kavitha, V. (2011) 'A survey on security issues in service delivery

models of cloud computing', *Journal of Network and Computer Applications*. Elsevier, 34(1), pp. 1–11. doi: 10.1016/j.jnca.2010.07.006.

Sulistio, A., Cibej, U., *et al.* (2008) 'A toolkit for modelling and simulating data Grids: an extension to GridSim', *Concurrency Computation Practice and Experience*, 22(6), pp. 685–701. doi: 10.1002/cpe.

Suresh, A. and Varatharajan, R. (2017) 'Competent resource provisioning and distribution techniques for cloud computing environment', *Cluster Computing*. Springer US, pp. 1–8. doi: 10.1007/s10586-017-1293-6.

Svärd, P., Li, W., *et al.* (2014) 'Continuous Datacenter Consolidation'.

Szabo, C. and Teo, Y. M. (2012) 'An objective-based approach for semantic validation of emergence in component-based simulation models', *Proceedings - 2012 ACM/IEEE/SCS 26th Workshop on Principles of Advanced and Distributed Simulation, PADS 2012*. IEEE, pp. 155–162. doi: 10.1109/PADS.2012.9.

Tchana, A., De Palma, N., *et al.* (2016) 'Software consolidation as an efficient energy and cost saving solution', *Future Generation Computer Systems*. Elsevier B.V., 58, pp. 1–12. doi: 10.1016/j.future.2015.11.027.

Thanakornworakij, T., Nassar, R., *et al.* (2012) 'An economic model for maximizing profit of a cloud service provider', *Proceedings - 2012 7th International Conference on Availability, Reliability and Security, ARES 2012*, pp. 274–279. doi: 10.1109/ARES.2012.44.

The Eclipse Foundation (2018a) *CDO Model Repository*. Available at: https://www.eclipse.org/cdo/ (Accessed: 21 January 2018).

The Eclipse Foundation (2018b) *Eclipse desktop and web IDEs*. Available at: https://www.eclipse.org/ide/ (Accessed: 20 January 2018).

The Eclipse Foundation (2018c) *Eclipse Modeling Project*. Available at: https://www.eclipse.org/modeling/ (Accessed: 20 January 2018).

Tian, W., Zhao, Y., *et al.* (2015) 'A toolkit for modeling and simulation of real-time virtual machine allocation in a cloud data center', *IEEE Transactions on Automation Science and Engineering*, 12(1), pp. 153–161. doi: 10.1109/TASE.2013.2266338.

Tighe, M. (2012) 'DCSim: A data centre simulation tool for evaluating dynamic

virtualized resource management.', *Network and service management (cnsm), 2012 8th international conference and 2012 workshop on systems virtualization management (svm). IEEE*.

TOP500.org (2016) *Top500 List - November 2016 | TOP500 Supercomputer Sites*. Available at: https://www.top500.org/list/2016/11/ (Accessed: 26 March 2017).

Umale, P. J. S. (2013) 'Survey on Job Scheduling Algorithms of Cloud Computing', 115(October), pp. 81–87.

Vaquero, L. M., Rodero-Merino, L. and Buyya, R. (2011) 'Dynamically scaling applications in the cloud', *ACM SIGCOMM Computer Communication Review*, 41(1), p. 45. doi: 10.1145/1925861.1925869.

Verma, A., Ahuja, P. and Neogi, A. (2008) 'pMapper: Power and migration cost aware application placement in virtualized systems', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5346 LNCS, pp. 243–264. doi: 10.1007/978-3-540-89856-6_13.

VMware (2011) 'Understanding Memory Resource Management in VMware vSphere 5.0', *VMware Whitepaper*, pp. 1–28. Available at: http://www.vmware.com/resources/techresources/10206.

VMware (2016a) *Cloud Computing is vCloud Air by VMware*. Available at: http://vcloud.vmware.com/uk/ (Accessed: 26 January 2016).

VMware (2016b) *vSphere ESXi Bare-Metal Hypervisor*. Available at: https://www.vmware.com/products/esxi-and-esx/overview (Accessed: 28 January 2016).

Wang, M., Meng, X. and Zhang, L. (2011) 'Consolidating virtual machines with dynamic bandwidth demand in data centers', *Proceedings - IEEE INFOCOM*, pp. 71–75. doi: 10.1109/INFCOM.2011.5935254.

Wang, X., Liu, Z., *et al.* (2012) 'LiveCloud: A lucid orchestrator for cloud datacenters', *CloudCom 2012 - Proceedings: 2012 4th IEEE International Conference on Cloud Computing Technology and Science*, pp. 341–348. doi: 10.1109/CloudCom.2012.6427544.

Weise, T. (2009) *Global Optimization Algorithms - Theory and Application*. Second.

Self-Published. Available at: http://www.vs.uni-kassel.de/staff/researchers/weise/.

Werner, H.-J., Knowles, P. J., *et al.* (2012) 'Molpro: a general-purpose quantum chemistry program package', *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 2(2), pp. 242–253. doi: 10.1002/wcms.82.

Wickremasinghe, B., Calheiros, R. N. and Buyya, R. (2010) 'CloudAnalyst: A CloudSim-Based Visual Modeller for Analysing Cloud Computing Environments and Applications', *2010 24th IEEE International Conference on Advanced Information Networking and Applications*. Ieee, pp. 446–452. doi: 10.1109/AINA.2010.32.

XenProject (2016) *The Xen Project, the powerful open source industry standard for virtualization.* Available at: http://www.xenproject.org/ (Accessed: 28 January 2016).

Xiao, Z., Song, W. and Chen, Q. (2012) 'Dynamic Resource Allocation using Virtual Machines for Cloud Computing Environment', *IEEE Transactions on Parallel and Distributed Systems*, 24(6), pp. 1–1. doi: 10.1109/TPDS.2012.283.

Xu, F., Liu, F., *et al.* (2014) 'Managing performance overhead of virtual machines in cloud computing: A survey, state of the art, and future directions', *Proceedings of the IEEE*, 102(1), pp. 11–31. doi: 10.1109/JPROC.2013.2287711.

Xu, F., Liu, F., *et al.* (2016) 'Heterogeneity and Interference-Aware Virtual Machine Provisioning for Predictable Performance in the Cloud', *IEEE TRANSACTIONS ON COMPUTERS*, 65(8), pp. 2470–2483.

Ye, K., Wu, Z., *et al.* (2015) 'Profiling-based workload consolidation and migration in virtualized data centers', *IEEE Transactions on Parallel and Distributed Systems*, 26(3), pp. 878–890. doi: 10.1109/TPDS.2014.2313335.

Younge, A. J., Henschel, R., *et al.* (2011) 'Analysis of Virtualization Technologies for High Performance Computing Environments', *2011 IEEE 4th International Conference on Cloud Computing*, pp. 9–16. doi: 10.1109/CLOUD.2011.29.

Youseff, L., Butrico, M. and Da Silva, D. (2008) 'Toward a unified ontology of cloud computing', *Grid Computing Environments Workshop, GCE 2008*. doi: 10.1109/GCE.2008.4738443.

Yue, M. (1991) 'A simple proof of the inequality FFD (L) ≤ 11/9 OPT (L) + 1, ∀L for the FFD bin-packing algorithm', *Acta Mathematicae Applicatae Sinica*, 7(4), pp. 321–331.

doi: 10.1007/BF02009683.

Zakarya, M. and Gillam, L. (2018) ‘Managing energy, performance and cost in large scale heterogeneous datacenters using migrations’, *Future Generation Computer Systems*. Elsevier B.V., 93, pp. 529–547. doi: 10.1016/j.future.2018.10.044.

Zhang, Z., Hsu, C.-C. and Chang, M. (2015) ‘Cool Cloud: A Practical Dynamic Virtual Machine Placement Framework for Energy Aware Data Centers’, *2015 IEEE 8th International Conference on Cloud Computing*, pp. 758–765. doi: 10.1109/CLOUD.2015.105.

Zhao, W., Peng, Y., *et al.* (2012) ‘Modeling and simulation of cloud computing: A review’, *2012 IEEE Asia Pacific Cloud Computing Congress (APCloudCC)*. Ieee, pp. 20–24. doi: 10.1109/APCloudCC.2012.6486505.

Zhu, J. and Wen, Q. (2012) ‘SaaS Access Control Research Based on UCON’, *2012 Fourth International Conference on Digital Home*, pp. 331–334. doi: 10.1109/ICDH.2012.50.

Ziegler, W. (2012) ‘SLAs for energy-efficient data centres: The standards-based approach of the OPTIMIS project’, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7396 LNCS, pp. 37–46. doi: 10.1007/978-3-642-33645-4_4.

Zissis, D. and Lekkas, D. (2012) ‘Addressing cloud computing security issues’, *Future Generation Computer Systems*. Elsevier B.V., 28(3), pp. 583–592. doi: 10.1016/j.future.2010.12.006.

Zowghi, D. and Coulin, C. (2005) ‘Requirements elicitation: A survey of techniques, approaches, and tools’, *Engineering and Managing Software Requirements*, pp. 19–46. doi: 10.1007/3-540-28244-0_2.

Appendix A. Published literature references

- Byrne, J., Svorobej, S., Castañé, G. G., Stier, C., Krach, S., Ali-Eldin, A., Krzywda, J. and Byrne, P. J. (2016) 'Final results from optimisation models validation and experimentation: project deliverable D6.5'. Open Access Repositorium der Universität Ulm. doi: 10.18725/OPARU-4312.

- Byrne, J., Svorobej, S., Giannoutakis, K. M., Tzovaras, D., Byrne, P. J., Östberg, P.-O., Gourinovitch, A. and Lynn, T. (2017) 'A Review of Cloud Computing Simulation Platforms and Related Environments', in Proceedings of the 7th International Conference on Cloud Computing and Services Science. SCITEPRESS - Science and Technology Publications, pp. 679–691. doi: 10.5220/0006373006790691.

- Castañé, G. G., Svorobej, S., Byrne, J., Stier, C., Krach, S., Krzywda, J., Hauser, C., Tsitsipas, A., Ahir, M., Allsop, J., Star, K., Byrne, P. J. and Ali-Eldin, A. (2016) 'CactoSim simulation framework final prototype: accompanying document for project deliverable D6.4'. Open Access Repositorium der Universität Ulm. doi: 10.18725/OPARU-4340.

- Groenda, H., Stier, C., Krzywda, J., Byrne, J., Svorobej, S., Papazachos, Z., Sheridan, C., Whigham, D. and Östberg, P.-O. (2014a) 'CACTOS toolkit version 1: project deliverable D5.2.1'. Open Access Repositorium der Universität Ulm. doi: 10.18725/OPARU-4318.

- Groenda, H., Stier, C., Krzywda, J., Byrne, J., Svorobej, S., Papazachos, Z., Sheridan, C., Whigham, D. and Östberg, P.-O. (2014b) 'Model integration method and supporting tooling: project deliverable D5.1'. Open Access Repositorium der Universität Ulm. doi: 10.18725/OPARU-4317.

- Groenda, H., Stier, C., Krzywda, J., Byrne, J., Svorobej, S., Castañé, G. G., Papazachos, Z., Sheridan, C., Whigham, D., Hauser, C., Tsitsipas, A., Domaschka, J., Ali-Eldin, A. and Östberg, P.-O. (2016) 'CACTOS toolkit version 2: accompanying document for prototype deliverable D5.2.2'. Open Access Repositorium der Universität Ulm. doi: 10.18725/OPARU-4319.

- Lynn, T., Gourinovitch, A., Byrne, J., Byrne, P. J., Svorobej, S., Giannoutakis, K., Kenny, D. and Morrison, J. (2017) 'A Preliminary Systematic Review of Computer

Science Literature on Cloud Computing Research using Open Source Simulation Platforms', in Proceedings of the 7th International Conference on Cloud Computing and Services Science. SCITEPRESS - Science and Technology Publications, pp. 565–573. doi: 10.5220/0006351805650573.

• Ostberg, P.-O. et al. (2014) 'The CACTOS Vision of Context-Aware Cloud Topology Optimization and Simulation', 2014 IEEE 6th International Conference on Cloud Computing Technology and Science, pp. 26–31. doi: 10.1109/CloudCom.2014.62.

• Svorobej, S., Byrne, J., Byrne, P. J., Groenda, H., Stier, C., Domaschka, J., Wesner, S., Krzywda, J. and Östberg, P.-O. (2014) 'CactoSim simulation framework initial prototype: project deliverable D6.1'. Open Access Repositorium der Universität Ulm. doi: 10.18725/OPARU-4321.

• Svorobej, S., Byrne, J., Castañé, G. G., Krzywda, J., Groenda, H., Stier, C., Domaschka, J., Ahir, M., Byrne, P. J. and Östberg, P.-O. (2015) 'Preliminary results from optimisation models validation and experimentation: project deliverable D6.2'. Open Access Repositorium der Universität Ulm. doi: 10.18725/OPARU-4322.

• Svorobej, S., Byrne, J., Liston, P., Byrne, P., Stier, C., Groenda, H., Papazachos, Z. and Nikolopoulos, D. (2015) 'Towards Automated Data-Driven Model Creation for Cloud Computing Simulation', in Proceedings of the Eighth EAI International Conference on Simulation Tools and Techniques. Athens, Greece: ACM. doi: 10.4108/eai.24-8-2015.2261129.

• Wesner, S., Groenda, H., Byrne, J., Svorobej, S., Hauser, C. and Domaschka, J. (2014) 'Optimised Cloud Data Centre Operation Supported by Simulation', eChallenges e-2014, IEEE, pp. 1–9. Available at: http://www.echallenges.org/e2014/outbox/eChallenges_ref_52_doc_9020.pdf.