Modelling and Simulation of ElasticSearch using CloudSim

Malika Bendechache*, Sergej Svorobej*, Patricia Takako Endo*‡, Manuel Noya Mario†,
M. Eduardo Ares†, James Byrne*, Theo Lynn*
*Dublin City University (DCU), Dublin, Ireland
Email: {malika.bendechache,sergej.svorobej, theo.lynn}@dcu.ie
† Linknovate, Santiago de Compostela, Spain
Email:manuel@linknovate.com
‡Universidade de Pernambuco, Recife, Brazil
Email: patricia.endo@upe.br

***Abstract*—Simulation can be a powerful technique for evaluating the performance of large-scale cloud computing services in a relatively low cost, low risk and time-sensitive manner. Large-scale data indexing, distribution and management is complex to analyse in a timely manner. In this paper, we extend the CloudSim cloud simulation framework to model and simulate a distributed search engine architecture and its workload characteristics. To test the simulation framework, we develop a model based on a real-world ElasticSearch deployment on Linknovate.com. An experimental evaluation of the framework, comparing simulated and actual query response time, precision and resource utilisation, suggests that the proposed framework is capable of predicting performance at different scales in a precise, accurate and efficient manner. The results can assist ElasticSearch users to manage their scalability and infrastructure requirements.**

***Keywords*—ElasticSearch, CloudSim, Simulation, Cloud, Workload, Query, Modelling, search engine.**

I. Introduction

Search engines are a complex two-sided network connecting billions of queries with billions of pages. Search engines are the most common method for consumers to source information on the Internet; in the UK, search engines are used by 94% of Internet adult users, by far the most popular source for information search [1]. In January 2019, nearly 10 billion search queries were processed by Google in the US alone [2]. Search result delays can lead to user frustration and result in loss of revenue [3]. The ubiquity and ease of use of search engines belie a deep layer of computational complexity to return relevant search results in fractions of a second. Search engine providers rely on the efficient provision, scaling, and optimisation of distributed compute infrastructure at hyper-scale to meet increasingly complex search functionality within and tightening service constraints [4].

ElasticSearch (ES) [5] is a popular open source search engine designed to be distributed, scalable, and capable of near real-time information retrieval [6]. With large and hyper-scale systems, it is not always feasible to emulate real production environments due to the high cost of accessing large clusters of computers, the downside risk of interfering with system performance, and logistical issues related to testing new algorithms with no access to actual online query traffic. Simulation can be a powerful technique for evaluating the performance of large-scale cloud computing services in a relatively low cost, low risk and time-sensitive manner [7], [8].

Search engines comprise of three major architectural components - web crawling, indexing, and query processing, all of which contribute to the scalability and efficiency of online search engines [4]. To accurately depict realistic search engine system behaviour, a simulation model must, therefore, consider (a) a realistic system workload in the form of queries submitted by users to the search engine, (b) a virtual resource provisioning allocation agnostic of the complexities of the underlying data centre hardware, and (c) a similar data flow logic as the actual system implementation, under examination.

In this paper, we model and simulate a search engine using Discrete Event Simulation (DES). To do so, we extend CloudSim [9], [10] with our simulation model and compare it with KPI (Key Performance Indicator) traces collected from a live ElasticSearch cluster deployed in a public cloud infrastructure by Linknovate.com. Our simulation framework supports a number of features that can help in search engine based system deployment and provisioning decisions:

- Modelling and simulation of a distributed data flow with a hierarchical architecture;
- Custom policy implementation for distributing workload in the hierarchical architecture;
- Synchronous communication between search engine components for data aggregation; and
- Flexible modelling that can be easily adapted to integrate with other CloudSim extensions.

The remaining of this paper is organised as follows. Section II introduces discrete event simulation, the CloudSim architecture, and the ES Search Engine. Section III summarises our modelling approach for a search engine. Section IV presents our use case based on Linknovate.com's deployment of ES. Section V outlines our methodology and Section VI presents and discusses the simulation results. Section VII briefly summarises selected related work. The paper concludes with a summary of the paper and a discussion of future work in Section VIII.

II. Background

A. Discrete Event Simulation: CloudSim

Discrete Event Simulation (DES) is a system modelling concept wherein the operation of a system is modelled as a chronological sequence of events [11]. DES-based decision support processes can be divided into three main phases: modelling, simulation, and finally results analysis. During the modelling phase, a simulated system is defined by grouping interacting entities that serve a particular purpose together in to a model. Once the representative system models are created, the simulation engine orchestrates a time-based event queue, where each event is admitted to the defined system model in sequence. An event represents actions happening in the system during operation time. Depending on the event type, the system reaction is simulated, and associated metrics captured. These metrics are collected at the end of the simulation for results analysis. Therefore, the system behaviour can be examined under different conditions. Using DES is beneficial in a complex, large scale, non-deterministic system environment where the system definition using mathematical equations may no longer be a feasible option [12].

There are many different DES frameworks that have been developed specifically for cloud computing and that provide a range of useful modelling features [13]. For this study we have chosen the CloudSim modelling framework, the most popular cloud computing simulation framework, due to its proven capability in simulating different cloud topologies and application architectures and its appropriateness for the scale envisioned in the use case [14]. CloudSim has two key features relevant to our paper. Firstly, it has a virtualisation engine that aids in the creation and management of multiple, independent, and co-hosted virtualised services on data centre nodes. Secondly, it is flexible and allows one to switch between space-shared and time-shared allocation of processing cores to virtualised services. These features speed up the development of new application provisioning algorithms for cloud computing [15].

CloudSim comprises two layers: (i) the Simulation layer provides support for modelling and simulation of virtualised Cloud-based data centre environments [15], and (ii) the User Code layer exposes basic entities for hosts, applications, VMs, number of users, application types, and broker scheduling policies [15]. Given its popularity, CloudSim has been extended significantly since the first version (e.g., [16]–[19]). In particular, CloudSimPlus [20] improved several engineering aspects, such as code maintainability, reusability and extensibility thereby enabling greater accuracy, usage simplicity, and extension facility. As such, we make use of CloudSimPlus in this paper.

B. ElasticSearch

ElasticSearch (ES) is an open source search engine which can provide distributed and real time searching capabilities. It is known as a document database for implementing Apache Lucene [21] as a back-end for document parsing and structuring [22]. ES has the following features:

- **Distributed**: Indices can be divided into shards (a chunk of data), with each shard capable of having any number of replicas. Routing and re-balancing operations are done automatically when new documents are added.
- **High Availability**: ES can form a cluster containing multiple copies of distributed shards providing error-resilient data storage. If any error is detected, it will automatically remove the failed nodes and re-organise itself to make sure data is safe and accessible.
- **Full Text Search**: It provides full query-based search capabilities using the Lucene information retrieval library.
- **Document Oriented**: ES uses NoSQL database to store data or documents as objects in JSON format. All documents are indexed by default, thus providing results at very fast speeds.
- **Schema-free**: ES automatically detects the data structure, data types, and indexes the data accordingly. Users can also define their own mapping and can change, if required. ES provides an automatic conflict resolution by versioning any changes within stored documents.
- **Scalability**: The ES server can start with a single node and can be scaled horizontally depending upon concrete requirements. More nodes can be added to the cluster dynamically if more capacity is needed.

III. Simulating a Search Engine in a Public Cloud

In order to simulate a workload in the ES search engine, we model it using the CloudSimPlus DES framework. Any task or event occurring in CloudSim is defined by a cloudlet which represents a submitted job. Therefore, in order to simulate the ES workload, we model each query as a set of cloudlets flowing through the nodes in the system (see Figure 2). Our simulation sets a parameter (# DN/Q) that represents the number of data nodes used to serve a query.

Providing that a node has the capacity to run a cloudlet, the execution time of the cloudlet is based on: (i) the total computational budget required by the cloudlet, (ii) the CPU of the VM, (iii) the number of cores the cloudlet is able to use in parallel, and (iv) the amount of CPU and RAM instructions the cloudlet is able to use at any given time.

The ES search engine has a distributed architecture with specific characteristics of parallel request processing and aggregation behaviour. Existing CloudSim models for load distribution only support sending cloudlets from one sender to one destination VM at pre-defined times creating one-to-one mapping between a cloudlet and a processing VM. To simulate the search engine behaviour, we extended the CloudSim framework to take into account the distributed system behaviour of both the ES architecture and the workload. The ES workload is distributed based on different criteria (e.g., the data shard distribution, the frequency access to a particular type of data that resides in a particular node in the system). This leads to different workloads on the data nodes. In our simulation, we present the workload as a probability distribution of the number of cloudlets running on every data node.

The following modelling functionalities were added to CloudSim:

- Single user queries consists of multiple cloudlets which can be processed in parallel or sequentially by available VMs.
- The ability to send cloudlets from one sender to several receiving VMs at the desired time (one-to-many).
- A synchronisation point in the model, where a defined search engine application component (parent) that generates multiple cloudlets (children) and waits for their execution to finish before continuing to process a user query.
- Allowing simulation users to design their own rules for workload distribution within cloud deployed distributed data systems. In this work, we simulate the workload distribution of an ES node as per the use case studied in section IV.

The CPU and the RAM usages are calculated automatically by the CloudSim simulator. However, the default CloudSim only logs/collects the evolution of the CPU usage of the VMs.

CloudSim takes as parameters how much CPU and RAM are consumed by a cloudlet and, based on these parameters, CloudSim calculates how long the cloudlet is running, and if many cloudlets are running at the same time on the VM, cloudsim will sum the CPU and RAM consummations of this VM. Every time there is a change in CPU or RAM consumption, CloudSim will create an event and save it in a log. CloudSim has different configurations for the amount of CPU and RAM a cloudlet is able to use at a time (i.e. full, absolute or relative amount). *Full* corresponds to a full CPU utilisation by the cloudlet; *absolute* corresponds to a fixed CPU or RAM amount defined by the user; and *relative* refers to a percentage usage of the CPU or RAM by the cloudlet. In our simulation, the amount of CPU and RAM instructions used by a cloudlet is set to absolute.

IV. Use Case: Linknovate.com

In order to show the results of our search engine simulation, we took a real use case of the ES search engine deployment at Linknovate.com. Founded in 2012 in Spain, Linknovate.com provides a business intelligence service to its clients. Linknovate.com clients primarily access competitive intelligence through a discovery engine deployed on the Microsoft Azure cloud, as opposed to a classic search engine. Advanced data processing at large scale is one of Linknovate.com's core activities. By deploying ES, Linknovate.com harvests metadata (e.g., authors name, affiliation, abstract etc.) from multiple sources, not just publications (Elseviers Scopus) or patent analysis (Thomson Reuters) but also more up-to-date sources like conference proceedings, presentations, grants (e.g. CORDIS) specialised blogs (e.g. Clean Technica) and specialised outlets (e.g. MIT Tech Review).

Linknovate.com manages vast amounts of information throughout different offline and online layers. The off-line layer, *Data Acquisition* comprises several pre-processing components working in parallel over raw data to homogenise structure and identify entities and semantic relations. The online layer, *Processing and Indexing*, is done over a virtual cluster of search nodes based on ES. Finally, the *Web and Search* layer is where user queries execute several internal queries over Linknovate.com indices, retrieving the data to be shown in the User Interface (UI). User queries are received by the virtual nginx web server that also renders the results pages (see Figure 1). In this paper, we are focusing on simulating the online virtual layers (web/search and ES cluster) of the Linknovate.com search engine. Figure 1 represents an overview of the virtualised layer of the Linknovate.com architecture.

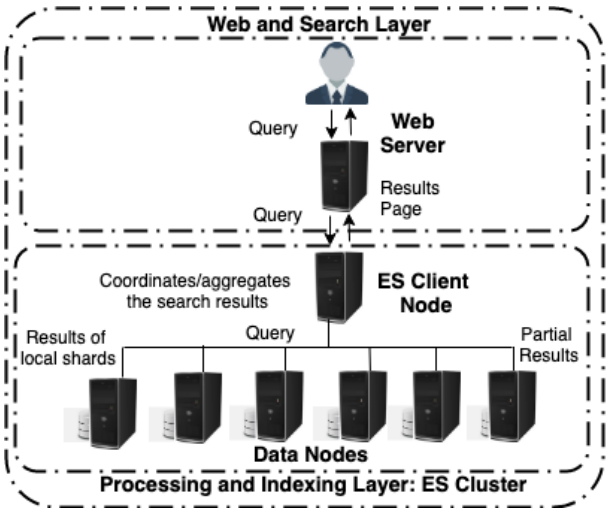

Fig. 1. An overview of the Linknovate.com online architecture.

The deployed search service stack of Linknovate.com consists of a web server where the users input their queries and an ES cluster which is responsible for the search and returning the response to the user query. The ES cluster consists of an ES node and data nodes as shown in Figure 1.

The ES node is responsible for: (i) passing and distributing the queries among the data nodes; (ii) coordinating and aggregating the search results of different data nodes; (iii) returning the query result to the web server which in turn will return it to the user. The data nodes are responsible for storing and processing old and fresh data.

Our simulation model reflects the behaviour of the real ES-based system deployed in a public cloud. As we can see from Figure 2, when a query is launched, a set of cloudlets are generated and executed in sequential manner; the first cloudlet is executed at a web server, then the second cloudlet is executed at the ES node. From the ES node, a set of cloudlets (which is less or equal to the number of data nodes) are distributed and executed at data nodes. Afterwards, another cloudlet is executed again at the ES node to merge the partial results coming back from the data nodes. Finally, a last cloudlet is going from the ES node to the web server as a response to the user query. Therefore, the total number of cloudlets that our simulation generates in order to model a

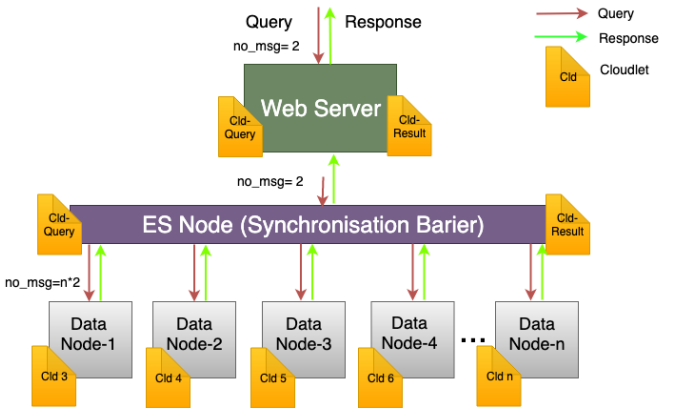

Fig. 2. Modelling the workload in ES using CloudSim

query load is $Cloudlet\_no = 4 + n$, where n is the number of data nodes queried by ES.

In terms of number of messages exchanged between nodes in the ES architecture, the total number of messages is represented by the total arrows in Figure 2. We have in total $Messages\_no = 4 + 2 * n$ messages, where n is the total number of data nodes queried by ES. The number of messages depends on the synchronisation at the ES node, where the ES node waits to receive all the messages from all data nodes in order to proceed with the aggregation (see Figure 2).

Query response time is calculated as follows:

$$ResponseTime = Time(EndOfLastCloudlet) - Time(QueryArrival) \quad (1)$$

Where $EndOfLastCloudlet$ corresponds to the final cloudlet at the web server that returns the query result to the user. $Time(QueryArrival)$ corresponds to the time the query arrives at the system (web server).

We can also calculate the Effective Time which is the time spent doing computations (sum of cloudlet execution times), without considering the wasted time (networking and waiting times). Note that the sum of cloudlet processing times is given by summing all the times for : cloudlets of the Web Server (WS) (two cloudlets), and ES node cloudlets (two cloudlets). Given that the data node cloudlets are running in parallel, we only sum the processing time of the longest (latest) data node (DN) cloudlet (see Figure 2). Therefore, as shown in Equation 2, we have a total sum of five cloudlet processing times.

$$\begin{aligned} EffectiveTime = {} & Time(WS\_Query_{Cloudlet}) \\ & + Time(WS\_Result_{Cloudlet}) \\ & + Time(ES\_Query_{Cloudlet}) \\ & + Time(ES\_Result_{Cloudlet}) \\ & + \max_{i=1}^{n} (Time(DN_{Cloudlet_n})) \end{aligned} \quad (2)$$

In CloudSim, every cloudlet c is set to run for a predefined number of CPU units $U(c)$ (to be defined by the simulation operator) that reflects the weight of the task it is meant to represent. On the other hand, every CPU core is only capable of executing a maximum number of CPU units per second relative to its frequency and other performance characteristics. We assume in our simulation that machines are equipped with 3000MHz CPU cores and we consider in our scenario that these CPU cores are able to run up to 3000 workload CPU units per second (i.e., one workload CPU unit per 1 MHz). If we set the CPU consumption of every cloudlet to 10% of the VM CPU capability (i.e., 300 CPU units per second), a VM would be able to fully execute a cloudlet within $\frac{U(c)}{300} second$ (assuming no preemption in the scheduling).

While we define the number of CPU units for all the cloudlets running in the WS and ES as 30, we assign the number of CPU units for the cloudlets in the data nodes dynamically based on the real workload that we have obtained from Linknovate.com. For every cloudlet c that runs on a data node and belonging to aquery q with a query response time RT(q), we define the number of CPU units for c as shows in Equation. 3. After computing the total CPU units required for the query q over all the cloudlets, we take out the CPU units for the cloudlets on WS and ES, leaving only the CPU units for the data nodes. Note that after filtering, the Linknovate.com workload does not contain any query q with a response time lower than 0.4s.

$$U(c) = 300 \times RT(q) - 4 \times 30 \quad (3)$$

In addition to the CPU consumption, CloudSim requires that the amount of RAM that will be utilised by every cloudlet throughout its execution to be provided. In our experiments, we allocate 200MB to all the cloudlets.

As we can see in Figure 1, Linknovate.com consists of eight VMs nodes in total, one web server VM, one ES VM, and six data nodes VMs.

TABLE I
LINKNOVATE.COM VM CHARACTERISTICS

	VM-ID	CPU (Cores)	RAM (GB)	STORAGE (GB)
Web-Server	VM0	8	28	1081
ES-Client	VM1	16	112	1081
DataNode1	VM2	8	28	1081
DataNode2	VM3	8	28	1081
DataNode3	VM4	8	28	1081
DataNode4	VM5	8	28	1081
DataNode5	VM6	8	28	1081
DataNode6	VM7	8	28	1081

Table I summarises the characteristics of the different nodes (VMs) forming the Linknovate.com topology.

In terms of workload distribution among the data nodes, our analysis of the six data nodes in the Linknovate.com architecture showed that they have different workload distributions (number of cloudlets run on each data node over a period of time) that follow these probabilities respectively: 0.13, 0.14, 0.16, 0.16, 0.18, and 0.23. These probabilities were calculated based on a real data set provided by Linknovate.com. See Section V for more details about the data set.

V. Methodology

The goals of our simulation are: *(a)* to evaluate the query response time, and resource consumption (CPU and memory) under different scenarios, and *(b)* to analyse the scalability of the simulator.

Linknovate.com deploys their infrastructure in a public cloud environment (i.e. Microsoft Azure). Therefore, in this simulation, we neither focus on the physical machines nor on the network physical topology.

To run our experiments, we used a real query data set provided by Linknovate.com. This data set is composed of 1185 queries submitted to Linknovate.com search engine between 11:23:00 and 13:23:00 on June 07, 2018. We first pre-processed the query log file by excluding errors or invalid queries and kept only the valid search queries (OK queries with HTTP Status=200).

In order to analyse the CPU and RAM consumption of the Linknovate.com system, we explore four scenarios as described in Table II.

TABLE II
SCENARIOS STUDIED BASED ON NUMBER OF DATA NODES SERVING A QUERY.

Scenario	Node Distribution
I	2 data nodes/q
II	3 data nodes/q
III	4 data nodes/q
IV	5 data nodes/q

The scenarios are defined based on the number of data nodes assigned to serve incoming queries. Given a query q, the set of nodes used to process it is randomly chosen. Therefore we run each scenario 30 times and calculated the average consumption across the 30 iterations for each VM. We also calculated the average of the average consumption of all the VMs in the system to reflect the CPU and RAM consumption of the whole system. The standard deviation of both CPU and RAM usage is also calculated.

VI. Simulation Results

We modelled and simulated the Linknovate.com architecture as per Section III.

A. Response Time Results

We compare the simulated response time of a query against its actual time as collected from real system traces. We extracted a subset of 100 valid queries from the data set used.

Figure 3 shows the comparison of actual and simulation times across the 100 queries. As one can see, the actual time and the simulation time are very close and they are highly positively correlated across all the 100 queries tested.

We evaluate the accuracy of our simulation model by computing the Pearson correlation [23] and the relative error (e_r) [24] between the simulation and the real system traces.

The accuracy of the query response time metric achieved by our simulation model reported a Pearson correlation of 0.9996, a positive correlation between the values reported by the simulation model and the real system traces. We obtained a small relative error of 0.0354; this indicates how close the query response time computed by the simulation ($SimTime$) model is relative to the actual query response time ($ActualTime$). The relative error is computed as:

$e_r = \epsilon_m / \overline{x}$ where

$$\epsilon_m = \sqrt{(\sum_{n=1}^{n} (SimTime_n - ActualTime_n)^2 / n}$$

where n is the number of queries (size of the sample) and $\overline{x}$ is the average of actual time.

B. Response Time vs Query Traffic

We analyse the performance of the Linknovate.com system by running the simulation with different workloads (query traffic) to see how much traffic the Linknovate.com system can handle. We monitor the query response time while varying the number of queries per second received by the system.

Figure 4 represents a box plot (min, max, lower quartile, upper quartile) that shows the query response time based on the number of queries per second the system receives.

Figure 4 shows that with query traffic of up to 80 q/s, the query response time for all the queries is the same and it is equal to having one query per second. That means the system is capable of handling 80 q/s with no waiting time.

Then, between 80 and 120 q/s, we notice a slight increase in the response time. However, this increase affects all the queries in the same way (i.e. no difference in response time between the queries).

As we increase the query traffic beyond 120 q/s, we start to notice a divergence in query response times. Between 130 and 170 q/s, we see that the system manages to execute several queries within a short time by delaying the excess of queries. However, with the increase in query traffic past 170 q/s, the system fails to even execute a single query in a short time.

C. CPU and RAM utilisation

We analyse the CPU and memory consumption of the Linknovate.com system based on the defined scenarios in Section V. As previously mentioned, the CPU units of both the ES and web server are set to a constant value equal to 300, whereas the CPU units of the data nodes are assigned dynamically based on the Linknovate.com workload.

As configured, the CPU and RAM consumption of the web server (VM0) and ES nodes (VM1) are constant across all the scenarios, while the CPU and RAM consumption of the data nodes vary based on number of nodes used (Figures 5 and 6). As expected, the more data nodes used by a query, the more CPU and RAM the system consumes. For example, Scenario I shows the least consumption for all the VMs in the system. The probability distributions simulated based on the Linknovate.com workload traces are clearly shown in both figures.

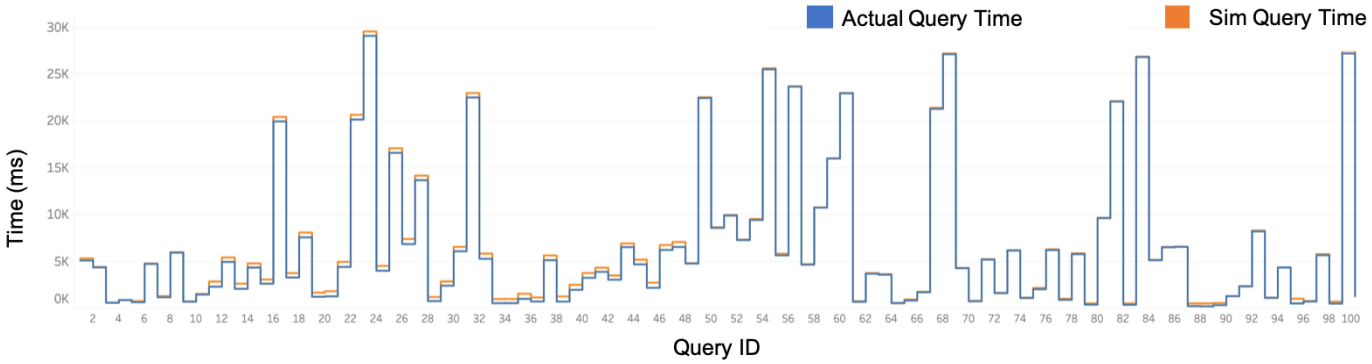

Fig. 3. Actual query time Vs Simulation query time.

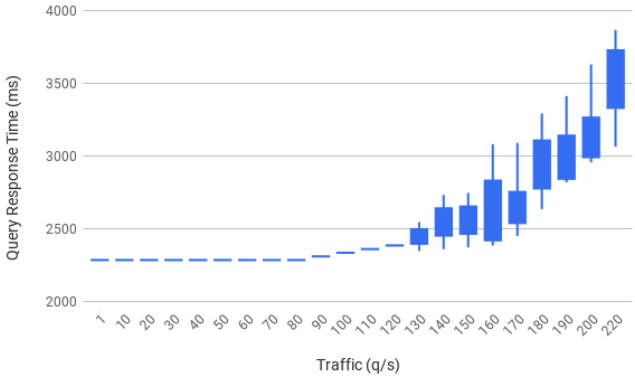

Fig. 4. Query response time achieved with different query traffic volumes.

VM7 always has the highest consumption of CPU and RAM across all the scenarios because it has the highest probability distribution (0.23) followed by VM6 with the second highest probability (0.18). VM4 and VM5 have the same probability distribution. Therefore they tend to have the same consumption. Finally, VM2 and VM3 report less consumption due to their low probabilities (0.13, 0.14, respectively).

D. Simulation Framework Scalability

In order to show the capability of our simulator in handling large datasets, we run the simulator with different log files that have different volumes of queries. We conducted our experiments by running our extended CloudSim simulator on a DELL-XPS machine with 8GB of RAM and 2.70 GHz quad Intel Core i5-6400 processor. The user query log file is collected from the Linknovate.com system for a period of two days; June 07 and 08, 2018. As one can see from Figure 7, the simulator takes only about 30 minutes to simulate a log file that contains up to 100k queries. In fact, up to 80k, the simulation time increases gradually with the number of queries. After that the simulation time starts to increase faster. This shows that the simulator can still handle comfortably medium to large files. However, for a faster simulation of very large files, a distributed version of the CloudSim simulator like *Cloud2Sim* [25] is needed for faster results. Note that CloudSim can be easily substituted by Cloud2Sim in our work for a faster simulation.

In summary, based on the results above, we conclude that our simulation results are very close to the real system measurements in terms of query response time (service time). The analysis of CPU and memory metrics across different scenarios shows how the system consumption responds to changes in the number of data nodes serving a query. As a result, this can help the company to manage their cost in terms of both CPU and memory consumption. Furthermore, the results also serve as feedback to Linknovate.com in terms of how much query traffic their system can accommodate at the same time. The company can consider increasing their system nodes' capacities to handle their desired traffic.

VII. Related Work

Web search engines are complex systems. Constructing a testbed for such complex systems with a high degree of verisimilitude is a complex, costly, resource and time-intensive task. To overcome these issues, simulation has been introduced. Simulation frameworks provide a relatively low cost mean to model, understand and evaluate a real system [26].

Although simulation is widely used for cloud computing research, there are few research articles that use simulation to model web search engine systems. Some are based on mathematical models and as such, they are complex. For example, in [27], the author modelled and simulated a search engine by developing two mathematical approaches - adaptive and selective approaches. These approaches seek to express the characteristics of the search engine based on the constraints in the information space.

Other research focused on using DES to model and simulate search engines. For instance, in [24], the authors modelled and simulated a web search engine using the Discrete Event System Specification (DEVS) formalism. The validation of the proposed model was done by comparing it against an actual MPI implementation of the WSE and a process-oriented simulation. DEVS was also used in [28] along with a discrete-event realisation of timed coloured Petri nets (CPN), and

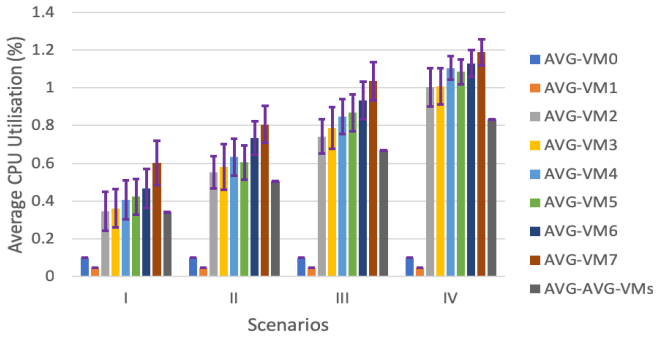

Fig. 5. Average CPU utilisation by the Linknovate.com system

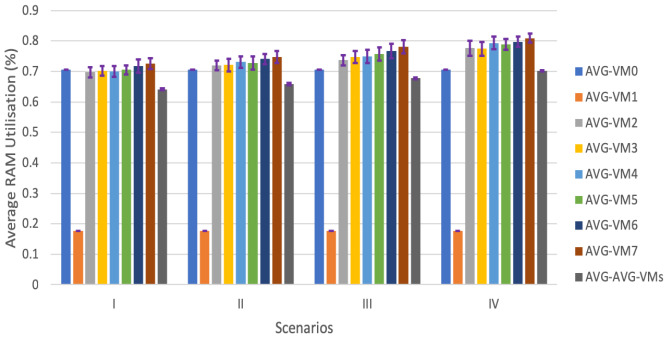

Fig. 6. Average RAM utilisation by the Linknovate.com System

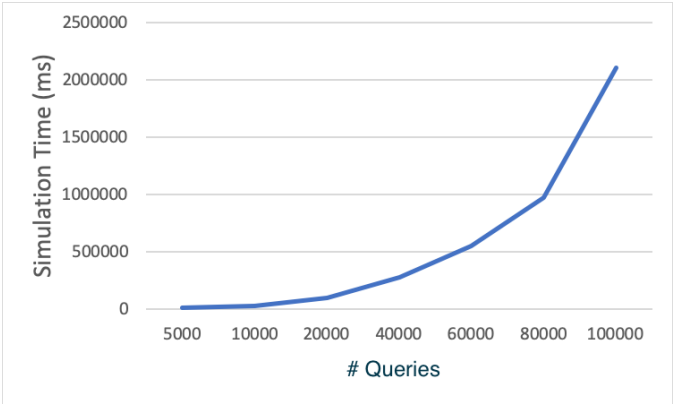

Fig. 7. Simulator scalability

process-oriented simulation (POS) to simulate user behaviour in search engines. They used a circulating tokens approach to represent sequences of operations that compete for search engine resources and benchmark programs to measure the cost of relevant operations. However, the authors of [28] only focused on simulating the computational cost of the search operations.

CloudSim is one of the most popular DES simulation frameworks, it has been used extensively to model and simulate cloud computing systems and application provisioning environments [29], [30]. The CloudSim toolkit supports both system and behaviour modelling of cloud system components such as data centres, VMs and resource provisioning policies. In fact, work in [17] looks at modelling parallel applications in the cloud using NetworkCloudSim, a CloudSim extension. However, the focus of their implementation lies on modelling network interplay between switches and routers traffic flow which is the area of infrastructure management useful more for the infrastructure provider. While the current study looks at a cloud application simulation, it is agnostic to the underlying data centre network.

Despite the popularity of ES search engine, no articles could be identified that simulated ES performance in the cloud. In this paper, we took advantage of the popularity of both CloudSim as a cloud simulator and ES as a powerful search engine to extend the CloudSim simulator framework in order to simulate the performance of ES search engine on a public cloud and thus inform better decision making on ES cloud deployments.

VIII. Conclusion and Future Work

We have described, modelled and simulated the Elastic-Search architecture. We proposed a new simulation model based on extending the CloudSim simulator. We have added to CloudSim two main features that characterise any distributed architecture: (i) modelling one-to-many cloudlets, and (ii) adding a synchronisation barrier at the ES node to allow it to wait for all the data nodes cloudlets to finish their execution before proceeding with aggregation of the data node results. The simulation helps us understand how the ElasticSearch search engine works. The Linknovate.com search engine is used as a use case study to validate our modelling and simulation. We compared the results of our simulation in terms of query response time against the actual query response time collected from actual Linknovate.com system traces. The evaluation of the accuracy of the simulation results shows no significant statistical differences between the simulated results and the real data. We also looked at the CPU and RAM consumption of the Linknovate.com system by taking different scenarios where we evaluate how the CPU and RAM usage vary based on the number of data nodes responding to a query. This can help the company to manage their costs. We also evaluated the query traffic volumes that Linknovate.com system can handle at a given time. This can help the company understand their system architecture and how they can improve/scale it in order to support their desirable query traffic.

For future research, we will use this simulation framework to examine key areas for optimisation of ElasticSearch in the cloud including efficient query balancing on the search servers, replica and shard allocations, and balancing CPU and memory usage of the different virtual nodes. We also plan to look at how the simulator can be used to inform auto-scaling to cope with changing system demands including deployment strategies and other balancing approaches.

ACKNOWLEDGEMENT

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 732667 (RECAP).

REFERENCES

[1] "Adults: Media use and attitudes report 2019," https://www.ofcom.org.uk, accessed: 2019-06-07.

[2] "Number of explicit core search queries powered by search engines in the united states as of january 2019 (in billions)," https://www.statista.com/statistics/265796/us-search-engines-ranked-by-number-of-core-searches, accessed: 2019-06-07.

[3] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, Dec. 2008. [Online]. Available: http://doi.acm.org/10.1145/1496091.1496103

[4] B. B. Cambazoglu and R. Baeza-Yates, "Scalability and efficiency challenges in large-scale web search engines," in *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '16. New York, NY, USA: ACM, 2016, pp. 1223–1226. [Online]. Available: http://doi.acm.org.dcu.idm.oclc.org/10.1145/2911451.2914808

[5] Elasticsearch B.V, "Open Source Search Analytics - ElasticSearch," 2019. [Online]. Available: https://www.elastic.co/

[6] O. Kononenko, O. Baysal, R. Holmes, and M. W. Godfrey, "Mining modern repositories with elasticsearch," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, ser. MSR 2014. New York, NY, USA: ACM, 2014, pp. 328–331. [Online]. Available: http://doi.acm.org.dcu.idm.oclc.org/10.1145/2597073.2597091

[7] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities," in *2009 international conference on high performance computing & simulation*. IEEE, 2009, pp. 1–11.

[8] S. Svorobej, P. Takako Endo, M. Bendechache, C. Filelis-Papadopoulos, K. M. Giannoutakis, G. A. Gravvanis, D. Tzovaras, J. Byrne, and T. Lynn, "Simulating fog and edge computing scenarios: An overview and research challenges," *Future Internet*, vol. 11, no. 3, p. 55, 2019.

[9] S. Mehmi, H. K. Verma, and A. Sangal, "Simulation modeling of cloud computing for smart grid using cloudsim," *Journal of Electrical Systems and Information Technology*, vol. 4, no. 1, pp. 159–172, 2017.

[10] G. T. Hicham and E. A. Chaker, "Cloud computing cpu allocation and scheduling algorithms using cloudsim simulator." *International Journal of Electrical & Computer Engineering (2088-8708)*, vol. 6, no. 4, 2016.

[11] A. M. Law, W. D. Kelton, and W. D. Kelton, *Simulation modeling and analysis*. McGraw-Hill New York, 2000, vol. 3.

[12] J. Idziorek, "Discrete event simulation model for analysis of horizontal scaling in the cloud computing model," in *Proceedings of the 2010 Winter Simulation Conference*. IEEE, 2010, pp. 3004–3014.

[13] J. Byrne, S. Svorobej, K. M. Giannoutakis, D. Tzovaras, P. J. Byrne, P. stberg, A. Gourinovitch, and T. Lynn, "A review of cloud computing simulation platforms and related environments," in *Proceedings of the 7th International Conference on Cloud Computing and Services Science - Volume 1: CLOSER,*, INSTICC. SciTePress, 2017, pp. 679–691.

[14] T. Lynn, A. Gourinovitch, J. Byrne, P. J. Byrne, S. Svorobej, K. Giannoutakis, D. Kenny, and J. Morrison, "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 - Volume 1: CLOSER,*, INSTICC. SciTePress, 2017, pp. 565–573.

[15] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.

[16] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, "Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications," in *2010 24th IEEE international conference on advanced information networking and applications*. IEEE, 2010, pp. 446–452.

[17] S. K. Garg and R. Buyya, "Networkcloudsim: Modelling parallel applications in cloud simulations," in *2011 Fourth IEEE International Conference on Utility and Cloud Computing*. IEEE, 2011, pp. 105–113.

[18] M. Barika, S. Garg, A. Chan, R. N. Calheiros, and R. Ranjan, "Iotsim-stream: Modelling stream graph application in cloud simulation," *Future Generation Computer Systems*, vol. 99, pp. 86–105, 2019.

[19] A. Siavashi and M. Momtazpour, "Gpucloudsim: an extension of cloudsim for modeling and simulation of gpus in cloud data centers," *The Journal of Supercomputing*, vol. 75, no. 5, pp. 2535–2561, 2019.

[20] M. C. Silva Filho, R. L. Oliveira, C. C. Monteiro, P. R. Inácio, and M. M. Freire, "Cloudsim plus: a cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2017, pp. 400–406.

[21] M. McCandless, E. Hatcher, and O. Gospodnetic, *Lucene in action: covers Apache Lucene 3.0*. Manning Publications Co., 2010.

[22] R. Kuc and M. Rogozinski, *Elasticsearch server*. Packt Publishing Ltd, 2013.

[23] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise reduction in speech processing*. Springer, 2009, pp. 1–4.

[24] A. Inostrosa-Psijas, G. Wainer, V. Gil-Costa, and M. Marin, "Devs modeling of large scale web search engines," in *Proceedings of the Winter Simulation Conference 2014*. IEEE, 2014, pp. 3060–3071.

[25] P. Kathiravelu and L. Veiga, "Concurrent and distributed cloudsim simulations," in *2014 IEEE 22nd International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems*. IEEE, 2014, pp. 490–493.

[26] V. Moysiadis, P. Sarigiannidis, and I. Moscholios, "Towards distributed data management in fog computing," *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[27] M. K. Nasution, "Modelling and simulation of search engine," in *Journal of Physics: Conference Series*, vol. 801, no. 1. IOP Publishing, 2017, p. 012078.

[28] M. Marin, V. Gil-Costa, C. Bonacic, and A. Inostrosa, "Simulating search engines," *Computing in Science & Engineering*, vol. 19, no. 1, p. 62, 2017.

[29] R. Kumar and G. Sahoo, "Cloud computing simulation using cloudsim," *arXiv preprint arXiv:1403.3253*, 2014.

[30] W. Long, L. Yuqing, and X. Qingxin, "Using cloudsim to model and simulate cloud computing environment," in *2013 Ninth International Conference on Computational Intelligence and Security*. IEEE, 2013, pp. 323–328.