Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 108C (2017) 79–88

Procedia Computer Science

International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland

Simulating a Search Engine Service focusing on Network Performance

Joe Carrión[1], Daniel Franco Puntes[1], and Emilio Luque[1]

Universitat Autónoma de Barcelona, Computer Architecture and Operative Systems Department, 08193, Bellaterra, Spain

joe.carrion@caos.uab.es, daniel.franco@uab.es, emilio.luque@uab.es

Abstract

Large-scale computer systems like Search Engines provide services to thousands of users, and their user demand can change suddenly. This unstable demand impacts sensitively to the service components (like network and hosts). The system should be able to address unexpected scenarios; otherwise, users would be forced to leave the service. Creating tests scenarios is an alternative to deal with this variable workload before implementing new configuration in the system. However, the complexity and size of the system are a huge constraint to create physical models. Simulation can help to test promising models of search engines. In this paper we propose a method to model a Search Engine Service (SES) on small scale to analyze the impact of different configurations. We model the interaction of a typical search engine with three main components: a Front Service (FS), a Cache Service (CS) and an Index service (IS). The FS takes as input a query of user and search into a database with the support of a CS to improve the performance of the system. The proposed model processes a trace file from a real SES and based on the dependency relation among the messages, services and queries, it is modeled the full functionality of the SES. The output is, on one hand a simulated trace file to compare the model with the real system and on the other hand statistics about performance. The simulation allow us to test configurations of FS, CS, and IS, which can be unlikely in the real system.

Peer-review under responsibility of the scientific committee of the International Conference on Computational Science

Keywords: Service simulation, network simulation, application-aware network, search engine services

1 Introduction

Search Engine Services are used by millions of users daily. Providers of these services are Google Inc., Yahoo, Baidu, Ask and although they prevail in the market[2] there are other environments where Search Engine Service are a useful tool like social networks or enterprise portals. This service is supported by the interaction of a vast set of small systems and services working together to create a complex structure able to scale from hundred of hosts to thousands of hosts interconnected. [13]. The planned and deployed resources for a search engine should

1877-0509
Peer-review under responsibility of the scientific committee of the International Conference on Computational Science
10.1016/j.procs.2017.05.202

be computed on expected demand. Although the capacity installed should support a dynamic workload related to variable user demand.

This research proposes a methodology to simulate the interaction of the hosts and services in order to analyze the impact of different scenarios. Additionally, it is possible to provide a baseline to design better promising configurations of a search engine. We studied a Search Engine Service (SES) with a typical configuration based on three components: Front Service (FS), Cache Service (CS) and Index Service (IS). These components are deployed over a cluster designed to support the service with a high speed network and redundant resources.

The performance of the system depends on the accepted workload and the installed capacity. In order to balance this two criteria, first we analyze the injection rate of queries and second we check metrics to manage the workload of the system. This analysis can be done in the real system on expenses of some overhead and to design hypothetical configurations. However, new resources and configurations need to be tested exhaustively. New configurations can be tested by the injection of artificial traffic. On literature like [3], [4], [6] we can find methods to simulate hypothetical conditions through the injection of synthetic traffic, we evaluated the importance to use real traffic. Therefore we propose a model with real traffic.

We analyze real traffic of the system for a period of time and simulate the services of the Front Service, Cache Service and Index Service. The model simulates the interaction of services. When a query arrives, it is submitted to next service based on the data flow of the system. When the query arrives to destination, it is simulated the time to solve the query and then it is returned to sender. Because of each query is divided in messages the simulation take into account the relation dependency among them. We modified a network simulator and we added a tier to simulate the search engine to process the traffic and inject it to the network. The output of the network is analyzed to simulate the services and re-inject the traffic to the network. Finally the output of the simulation should be analyzed to reconfigure the system with new parameters and create new scenarios for the service.

In section 2 we present a summary of relevant publications and previous work of the authors related to this work. In Section 3 an overview about the search engine architecture is presented. Section 4 details the methodology proposed and in Section 5 an evaluation of this work is given. Finally in 6 the conclusions and future work are presented.

2 Related Work

In order to model the SES a methodology is proposed in [12], where is defined a simulation process of a full SES. That methodology proposes modeling the services with parallel computation and it uses benchmarks to measure the cost of the services (FS, CS, IS and network). Authors in[9] present a discrete event simulation specification of a SES, where the queries are simulated with a message moving around different stages to compute the time to solve it, authors compared the results with a real web search engine trace file. The accuracy of the experiments of those simulations allows to generate trace files to compare with the real system. Additionally, the results enable to extend the test scenarios to try new experiments related to others components of the system like the network.

The authors in [6] proposed a network simulator as a tool to analyze network designs, and it is detailed the importance of the specification of performance requirements. For a complex system like SES the performance is based on the number of queries solved by second, it means in terms of application performance, then modelling the application with real traffic is necessary. There are critical issues regarding traffic of real applications, which is, the relation dependency among messages, authors in [8] addressed this issue by encoding the dependencies with two

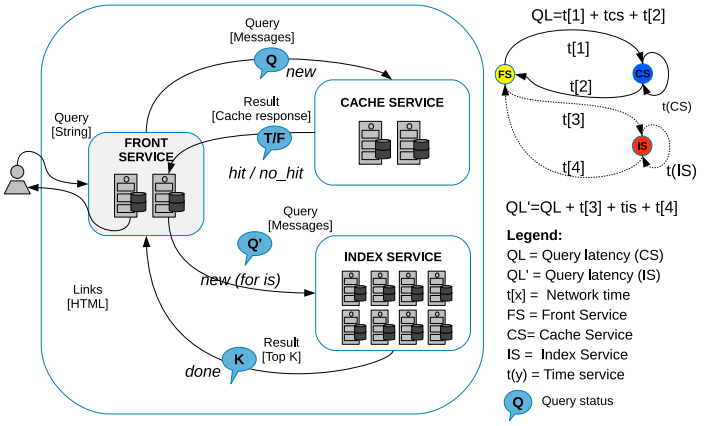


Figure 1: Overview of main flows of SES. Front service submit queries to Cache service, Cache Service returns Success or Fail. On Fail, queries are submitted to Index service.

steps, the first one is a full-simulation to detect and encode the dependencies and the second step includes a new simulation based in the execution of windows of time to ensure related messages are processed in order.

Authors have proposed simulation techniques to analyze the network and new algorithms to improve the network performance, based on a detailed traffic pattern analysis of SES [5].

3 Background

The SES analyzed has three main software components: Front Service (FS), Cache Service (CS) an Index Service (IS) [7]. SES service processes queries (Q) from users through a FS. The FS distributes the query to a set of CS, the CS checks if Q has been solved previously. CS returns a success or fail to FS, on fail, FS submits the Q to to IS. The IS generates a list with the relevant documents to Q (Top K). The Figure 1 illustrates the steps to solve a query.

The output of the system is a list of relevant user documents (K) for the query. The time to solve a query is called the Query Latency (QL). QL defined by the time to process the query by the FS plus the network time and plus the time to solve a query by the CS. When Q is solved by the IS we named Q'. QL' is defined by QL plus the time need to solve the query by IS. A graph to compute QL and QL' is showed in Figure 1.

The main requirement of a SES is solve Q in a defined period of time, a metric of the service is given by the Throughput (T). T is defined by K divided by Q in the same period of time.

3.1 Search Engine Service Architecture

The SES is deployed on a large cluster of computers. The nodes are arranged on arrays of P x D, where P defines the level of data partitioning and D the level of replication of the data, it is used a partitioning and redundancy schemes to improve the performance of the SES in terms of QL. The cluster of the SES is deployed on a fat-tree topology[1]. A full description of the architecture is published in [7]. The number of instances of each service is defined in the configuration of the SES. We show an overview in Figure 1.

3.2 Front Service

The FS accepts Q from users and the service returns a list of relevant documents to the query. A query is composed by a set of keywords. The FS is a cluster with a defined number of hosts running an instance of FS.

Each query has four states: *new*, *hit*, *no_hit* or *done*. FS processes *new* Q and submits it to different CS. If the Q state is *no_hit*, Q is submitted to IS. On *hit* or *done* results K are returned to user. Each state increases QL according to estimated time for each service. Only IS and CS change Q state. FS distributes Q based on algorithms (like Round Robin) to balance the workload among different CS and IS hosts. Figure 1 shows an scheme of the different Q states.

3.3 Cache Service

The CS keeps a set of the most frequent queries and results. Basically there are two main tasks on CS service, the first one is processing new queries and the second one is to keep updated its content to solve Q quickly. The operation of the CS involves time to balance the workload among CS nodes. At the same time, CS processes previous results based on a set of cache policies. Authors of [7] explain in detail the CS features. CS updates Q state from *new* to *hit* or *no_hit* and it increases QL.

3.4 Index Service

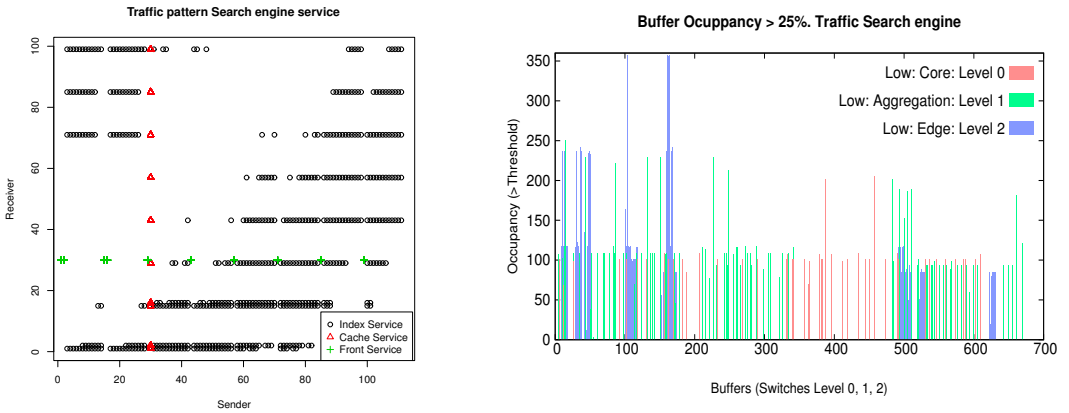
The IS computes a list of the top-K relevant documents for Q. To do this, there is parallel process to recover documents from the web to create a document collection. Each document is associated with an unique identifier and it is parsed to recover a list of relevant terms. The result of this process is a structure named inverted index. This structure can be huge and it is necessary distribute it among the hosts of the IS cluster. The IS accesses to the inverted index for relevant documents, because of the results can be stored in different nodes it is applied a ranking algorithm to compute a Top-K list with links to the original document. IS updates Q state from *no_hit* to *done* and it increases QL.

3.5 Traffic Pattern Analysis of SES

In order to design a model to simulate SES we have studied the traffic pattern of the real system. In a SES the communication pattern is defined by the flows of the service architecture. The load of the system depends on two elements, the volume of user requests (Q) and the size of the content stored in the system. The volume of users submitting queries in a period of time is unpredictable. This input rate triggers an unstable communication pattern.

We have conducted a set of experiments using simulation techniques to analyze the traffic pattern of SES. For instance below we introduce two basic looks related to traffic patterns.

The first one is shown in Figure 2 (a), where axis X and Y are the host identification from 1 to 128 hosts. The marks correspond to the service which submits the message. We can see that each node only contacts with a delimited set of services, a CS node submits messages to a single FS node, and IS node submits messages only to a single FS node. Using the flow-traffic conditions described on Figure 1, the set of couples is deterministic. Each pair S-R (Sender - Receiver) is created with a delimited set of nodes. The tendency reduces the set of couples and it creates an unbalanced traffic. On one hand if there is unused resources the network could be reduced, on the other hand there are overloaded buffers and they can generate bottlenecks.



(a) (b)

Figure 2: (a) Pairs of Senders and Receivers by kind of service. (b) Number on events where the buffer capacity was higher than a threshold of (25%).

The second experiment focuses on analyze the traffic workload among switches. In order to analyze the impact of traffic, we monitor the network in a window of time and we define a threshold of 25% of the buffer capacity to count the number of events when the occupancy is higher than the threshold. For instance with a fat-tree network with three levels, Figure 2(b) depicts the buffers occupancy, we can see imbalanced traffic among the network and there are some buffers with very high occupancy and unused buffers.

4 Network Simulation of SES

This paper proposes a methodology to simulate a search engine service with an approach based on real traces. In Figure 3(a) we show the current (actual) method based on workloads, where the analysis of network performance is based on traffic generated by mathematical model, random numbers or bit-based. In the Figure 3(b) we depict our a method based on real traces to simulate the application (proposed).

The proposed methodology is based on the interactions of application components, the dependency of the traffic (workload of services) and the flow-data conditions. Additionally It has an analytical phase (external) to reconfigure the system in order to create new test scenarios for experimentation, this phase is based on the information provided by simulation which contains details related to the services.

4.1 Application Tier

The proposed model creates a tier to simulate the application interacting with the network. Figure 4 illustrates the basic model, where a set of instances of FS, CS, and IS inject messages to the network and the simulator returns the messages to services.

We used a modified version of Booksim simulator published in [10] and [11]. The modified Booksim to support real traces of the SES. We included three main components to the simulator,

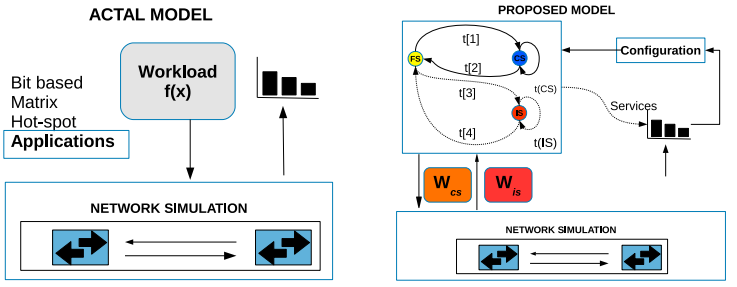


Figure 3: (a) Traditional model. (b) Proposed model.

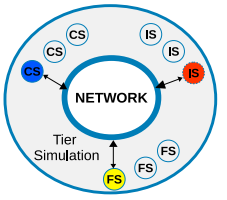


Figure 4: Basic scheme of the Application Tier interacting with the Network Simulator.

first a module to process an application trace file, second an injector traffic based on the services of the application and third a feature to validate the messages dependency among the services.

4.2 Application Simulation

The simulation takes as input a trace file (tf) and a mapping file (mf) generated by the real service. The *tf* contains the query (Q), and the services interchange messages (m) along a window of time (t).

The *tf* contains a the traffic generated by each host of a real system. The *mf* contains the configuration of the SES and the role of each host (IS, FS, CS) . The output of the simulation is a log with details about the network performance. The structure of *tf* and *mf* are shown in Table 1.

Host	POD	Rol	Replica	Host replica
IP	(0..10)	FS/CS/IS	R/NR	IP

(a) Mapping file structure (mf).

ID	timestamp	Op1	Op2	Event	CPU	Src	Dest	Size	Q	D

(b) Trace file structure (tf).

Table 1: Trace files structure.

The SES in defined with the number of instances of FS, CS and IS. For instance a configuration with one FS, two CS and three IS is used in Figure 5. The time (t) to solve Q by CS and FS are provides by *tf*. We call this cost as tFS, tCS, tIS.

The simulation starts with the injection of Q, then the FS creates m as number of instance of CS are defined, then m is injected to the kernel of network simulator. When m arrives to CS host, the tCS is simulated. CS submits messages to network simulator and FS wait until all from CS messages related to the same Q has been returned. If at least one message has changed to *hit* state the results are submitted to user and a new entry to the simulated trace

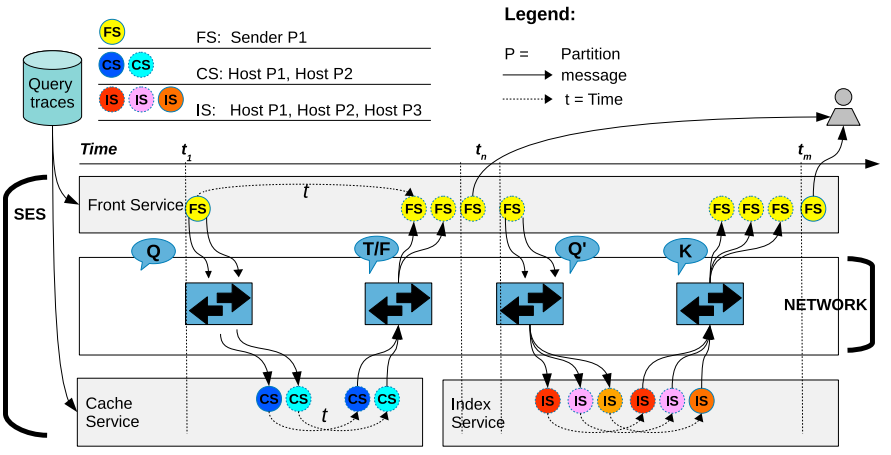
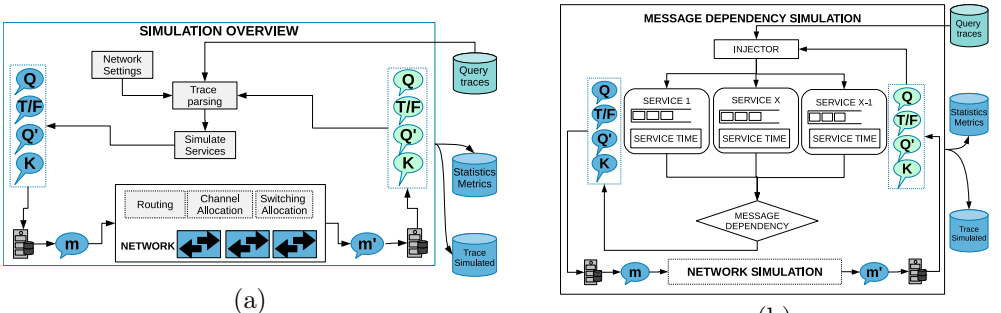


Figure 5: Pipeline phases of SES simulation process.

is created. When the state is *no_hit*, Q is submitted to IS hosts. Then IS submits the messages to the network and and the network simulator returns messages to FS. Finally FS waits for all messages for the same Q and submits K to user. The Figure 5 illustrates this process.

4.3 Message Dependency

A Q is solved by a set of m, by the flow of communication among services and by a sequence ordered of messages. Messages compete for network resources, therefore some times messages spend time on queues. This condition causes some messages for the same query arrive disordered to the destination, then checking if all messages of Q have arrived is necessary before injecting them again to next service or to submit it to user. Figure 6(a) shows an overview of the simulation process and Figure 6(b) shows the process to verify the message dependency.



(a) (b)

Figure 6: (a) Simulation is based on the interaction of FS, CS, IS and the network simulation. (b) The traffic from search engine is injected to queue services, simulation checks the messages dependency before injecting messages to next service.

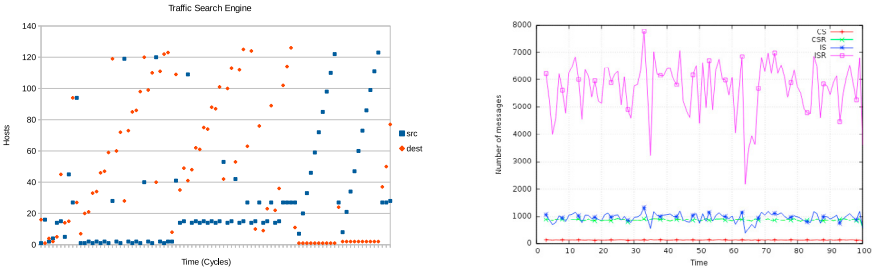
4.4 Query Injection

First we introduce the attributes included in *tf*. The *ID* is a unique identificator of each entry. *Timestamp* allows to measure the injection rate of Q, *Srv1* and *Srv2* are the services (source and destination), The *Event* defines each entry as *m* or *processing time*, *CPU* attribute is the cost of processing Q by the service. *Src* and *Des* are a host identifier. *Q* is the query identifier and finally *data* is the Q state. In Figure 5 we illustrate the process to inject the traffic to the simulator. FS read Q from *tf*. The CS destinations are selected from *tf*. The messages are submitted to network simulator, and the output of network is stored in a queue to wait for all messages from the same Q. The process continues until the full trace has been processed.

5 Evaluation

The data for the experimentation corresponds to trace files used by authors of [7]. Trace files were obtained by Yahoo Search Engine in 2005.

We are using a trace log file with a configuration of 128 host. The network topology is a Fat-tree topology. The topology is created using a feature of simulator named *anynet*. First we parse the *mf* to generate the *anynet* topology in the format required by the simulator. This topology is passed to simulator in the configuration file using the parameter *network_file*.



(a) (b)

Figure 7: (a) Mapping of messages among services along a window of time. (b) Workload comparison among services in a windows of time.

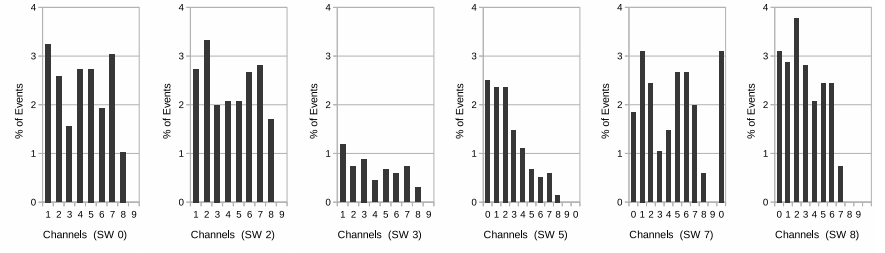


Figure 8: Buffer occupancy in a defined window of time for switches of Level 0. Axis X shows channels and axis Y shows percentage of events where Occupancy was higher than a threshold of 25% of the capacity (Switches 1, 4 and 6 reported 0 events).

In Figure 7 (a) we show a sequence of messages between nodes in a period of time. This chart

allows us to recognize a communication pattern where most of Senders belongs to nodes from 1 to 20 and the Receivers go from 1 to 128 with a constant trade-off. This should be analyzed with the current configuration in order to define if this is the expected work-balance over the network. Also the report can be compared with the workload of each service, in Figure 7 (b) is shown a chart with the workload by service. We can see that most of the traffic corresponds to IS replica, and the number of messages of CS is the lowest. Regarding to work balance, Figure 8 shows a comparison among six switches of Level 0 of the Fat-tre topology, it is shown the workload for nine channels, as you can see the traffic has been distributed among 5 switches and switch 3 has the lower workload. Next analysis is related to QL, Figure 9 shows the real QL and the simulator QL. We got in the simulation a QL of 35.8 ms. against 34.8 ms. in the real trace, it means 97.1% of accuracy. Finally Figure 10 shows the QL distribution for the same sample. We can see that simulator slightly increases QL however the distribution belongs to the same range of values.

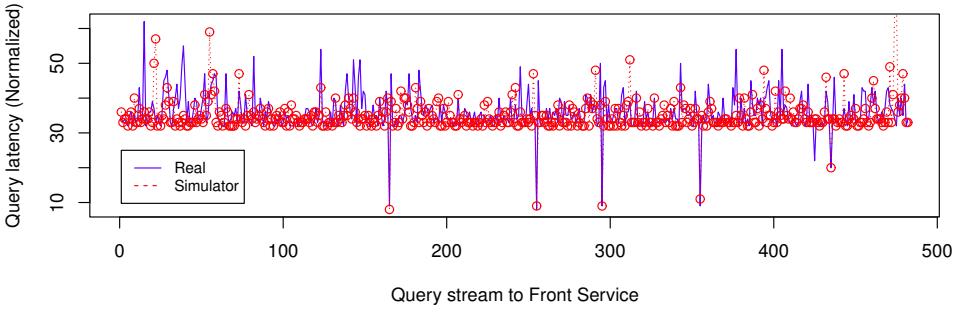
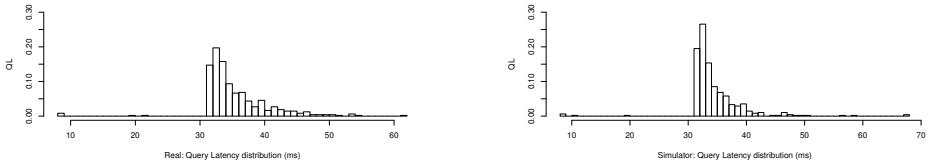


Figure 9: Sequence of input using a sample of 480 queries.



(a) (b)

Figure 10: (a) QL Real distribution (b) QL Simulated distribution.

6 Conclusions

We have presented a methodology to simulate a Search Engine Service in a small scale in order to evaluate the performance of promising scenarios. We simulated the interaction among main components on the system based on a real trace. The output is a useful tool to create experiments in order to analyze and design new configurations of the services and network. As

future work, this model also allow us introduce new algorithms to deal with issues like network congestion and to analyse overload of services.

7 Acknowledgments

This research has been supported by the MINECO Spain under contract TIN2014-53172-P, SENESCYT[1] under contract 2013-AR7L335. Authors would like to thank to Veronica Gil-Costa, Mauricio Marin and Centro de Biotecnología and Bioinformática under Basal Project.

References

[1] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 63–74. ACM, 2008.

[2] Net Applications. Market Share Statistics for Internet Technologies, 2016. `https://www.netmarketshare.com/`.

[3] J.J.P. Arias, A.S. González, and R.P.D. Redondo. *Teoría de colas y simulación de eventos discretos*. Pearson Educación, 2003.

[4] Jerry Banks, John S Carson, Barry L Nelson, and David M Nicol. *Discrete-Event System Simulation: Pearson New International Edition*. Pearson Higher Ed, 2013.

[5] Joe Carríon, Daniel Franco, and Emilio Luque. Application-aware routing policy based on application pattern traffic. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, page 142. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2015.

[6] William James Dally and Brian Patrick Towles. *Principles and practices of interconnection networks*. Elsevier, 2004.

[7] Veronica Gil-Costa, Jair Lobos, Alonso Inostrosa-Psijas, and Mauricio Marin. Capacity planning for vertical search engines: An approach based on coloured petri nets. In *International Conference on Application and Theory of Petri Nets and Concurrency*, pages 288–307. Springer, 2012.

[8] Joel Hestness, Boris Grot, and Stephen W Keckler. Netrace: dependency-driven trace-based network-on-chip simulation. In *Proceedings of the Third International Workshop on Network on Chip Architectures*, pages 31–36. ACM, 2010.

[9] Alonso Inostrosa-Psijas, Gabriel Wainer, Veronica Gil-Costa, and Mauricio Marin. Devs modeling of large scale web search engines. In *Simulation Conference (WSC), 2014 Winter*, pages 3060–3071. IEEE, 2014.

[10] Nan Jiang, James Balfour, Daniel U Becker, Brian Towles, William J Dally, George Michelogiannakis, and John Kim. A detailed and flexible cycle-accurate network-on-chip simulator. In *Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium on*, pages 86–96. IEEE, 2013.

[11] Nan Jiang, George Michelogiannakis, Daniel Becker, Brian Towles, and William J Dally. Booksim 2.0 users guide. *Standford University*, 2010.

[12] Mauricio Marin and volume=1 pages=1–1 year=2017 publisher=IEEE Gil-Costa, Vernica journal=Computing in Science and Engineering. Simulating search engines.

[13] John Wilkes. Keynote: Cluster management at Google. In *Federated Computing Research Conference*, FCRC '15, pages 1–, New York, NY, USA, 2015. ACM.

[1]SENESCYT: Secretaría Nacional de Educación Superior, Ciencia, Tecnología e Innovación, http://www.senescyt.gob.ec/