-ORIGINAL ARTICLE-

Experimental Framework to Simulate Rescue Operations after a Natural Disaster

Framework Experimental para Simular Operaciones de Rescate luego de un Desastre Natural

Luis Veas-Castillo[1], Gabriel Ovando-Leon[1], Gabriel Astudillo[2], Veronica Gil-Costa[3], and Mauricio Marin[1]

[1]*DIINF, CITIAPS, CeBiB, Universidad de Santiago de Chile, Santiago, Chile*
{ luis.veasc, juan.ovando, mauricio.marin }@usach.cl
[2]*Universidad de Valparaíso, Valparaíso, Chile*
gabriel.astudillo@uv.cl
[3] *Universidad Nacional de San Luis, San Luis, Argentina*
gvcosta@unsl.edu.ar

Abstract

Computational simulation is a powerful tool for performance evaluation of computational systems. It is useful to make capacity planning of data center clusters, to obtain profiling reports of software applications and to detect bottlenecks. It has been used in different research areas like large scale Web search engines, natural disaster evacuations, computational biology, human behavior and tendency, among many others. However, properly tuning the parameters of the simulators, defining the scenarios to be simulated and collecting the data traces is not an easy task. It is an incremental process which requires constantly comparing the estimated metrics and the flow of simulated actions against real data. In this work, we present an experimental framework designed for the development of large scale simulations of two applications used upon the occurrence of a natural disaster strikes. The first one is a social application aimed to register volunteers and manage emergency campaigns and tasks. The second one is a benchmark application a data repository named MongoDB. The applications are deployed in a distributed platform which combines different technologies like a Proxy, a Containers Orchestrator, Containers and a NoSQL Database. We simulate both applications and the architecture platform. We validate our simulators using real traces collected during simulacrums of emergency situations.

Keywords: Experimental framework, Simulation, Benchmark

Resumen

La simulación computacional es una poderosa herramienta para evaluar el rendimiento de sistemas. Resulta útil para realizar el planeamiento de capacidad de clusters de Centros de Datos, para obtener perfiles de aplicaciones y detectar cuellos de botella. Se ha utilizado en diferentes áreas de investigación como buscadores web a gran escala, evacuaciones por desastres naturales, biología computacional, comportamiento y tendencia humana, entre otros. Sin embargo, ajustar correctamente los parámetros de los simuladores, definir los escenarios de simulación y recopilar los rastros de datos no es una tarea fácil. Es un proceso incremental que requiere contrastar constantemente las métricas estimadas y el flujo de acciones simuladas con datos reales. En este trabajo, presentamos el diseño de un marco experimental para el desarrollo de simulaciones a gran escala de aplicaciones sociales utilizadas después de un desastre natural. La primera es una aplicación social destinada a registrar voluntarios y gestionar campañas en emergencias y tareas. La segunda aplicación es un repositorio de datos llamado MongoDB. Las aplicaciones se depliegan en una plataforma distribuida que combina diferentes tecnologías como Proxy, Orquestador de Containers, Containers y una Base de Datos NoSQL. Simulamos ambas aplicaciones y la plataforma computational. Validamos nuestros simuladores utilizando trazas reales recopiladas durante simulacros.

Palabras claves: Framework Experimental, Simulación, Benchmark

1 Introduction

Performance evaluation by means of discrete-event simulation of large scale social software applications is a topic that has deserved little attention even in well-established research areas such as Web search [1, 2], whereas has deserved none in much less developed areas such as emergency management of natural disasters. Yet performance is a relevant topic to be taken into consideration when designing social applications intended to scale to thousands/millions of users. This

is especially relevant when the target recipients are people affected by a disaster or volunteers that must be properly coordinated to be effective and efficient in relief operations. In these cases, it is desirable to be able to anticipate critical issues when operating the application at large scale and demanding user dynamics such as the amount of hardware resources required to support the workload and points of potential bottlenecks. In general, these issues are indeed relevant in many other application domains when they are required to be prepared for success coming from an exponential growth in new users and activity. In practice, testing the application may be too expensive or not feasible under large scale scenarios at software development time, which gives place to the need for tools able to predict performance. During production operation, performance evaluation is also relevant in the form of capacity planning studies devoted to determine the economical use of hardware resources deployed at the data center [3].

The complexity of these applications/systems placed in their operational contexts is evident. Performance is featured by both the dynamics of user behavior and the multiple software and hardware platforms where the applications are expected to be run. We mean platforms ranging from clusters of processors to large collections of smartphones, with different layers of data communication in between, and multiple software platforms providing services to the applications [4]. This combination makes it difficult (maybe impossible) to properly evaluate performance with analytical methods in a meaningful practical sense.

In this work, we present an experimental framework for large scale piloting-based simulations aimed to manage the rescue operations and tasks after a natural disaster strikes. Figure 1 shows the general scheme of our proposed framework. The "Data Traces" box represents traces of user requests that show the users behavior in each application, like the requests sent to each application, its timestamp and the GPS position. We obtain this information during a simulacrum of a natural disaster where different agencies and institutions, such as ONEMI[1] and MovidosxChile[2], instruct people to use social applications specially designed to manage information and coordinate the human resources (volunteers). We call this piloting. We generate a larger synthetic trace of requests by adapting the real trace based on its empirical distribution. Then we use the synthetic trace to perform an application-specific benchmark on each system/app through REST services. This allows us to obtain the real execution time for each task executed in the apps. For example, for the "Register New User" task we obtained an average execution time of 442.1 ms and a standard deviation of 25.8 ms on the platform described in Section 3.

[1]https://www.onemi.gov.cl/noticia/crean-software-y-aplicaciones-para-organizar-ayuda-en-caso-de-catastrofes/

[2]https://movidosxchile.cl/

In addition, we developed hardware benchmarks [5] to obtain runtime distributions, e.g. for the network communication. The simulator takes as input parameters the execution times obtained with the benchmarks and simulates computational infrastructure which includes the Bot (Front-End) used to access the application, the communication network and the distributed cluster of multi-core computers (Back-End) where the applications are executed and receive requests according to the synthetic trace.

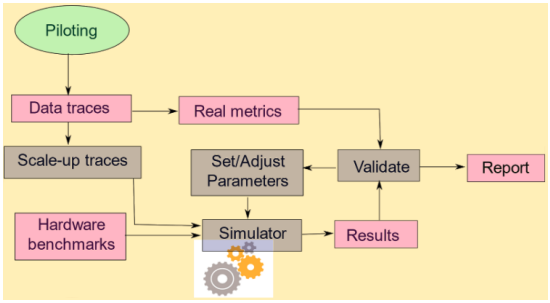


Figure 1: General scheme of our experimental design.

As a case study, we simulate two applications. The first one is a social application aimed to register volunteers and manage emergency campaigns. The second application is a data repository named MongoDB [6]. We validate our simulations results against real data obtained from the piloting. The parameters of the simulator have to be adjusted to reduce the error of the simulation. Results show that our simulators are highly correlated with the real data and we can estimate the execution times of the applications with a small relative error of 1,37 at most.

The remaining of this paper is organized as follows. Section 2 presents the experimental design. Section 3 presents the technologies used to design our distributed platform and the applications. In Section 4 we experimentally validate our simulators and Section 5 concludes.

2 Experimental Framework for Large Scale Piloting-Based Simulations

2.1 Experimental Framework

The deployment of the experimental scenario consists mainly of a series of activities, which aim to obtain valid data, close to reality, which will allow the resulting models and simulators to be developed, adjusted and validated. Figure 2 shows the different components that are part of the experimental framework, where different agencies and institutions participate during the piloting of social applications and in the simulacrum of natural disasters in the north and south of Chile. They also give access to distributed computing architecture in which the computing platform can be deployed.

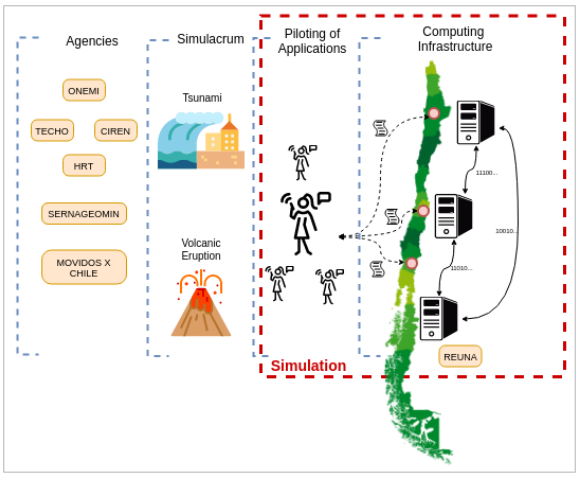


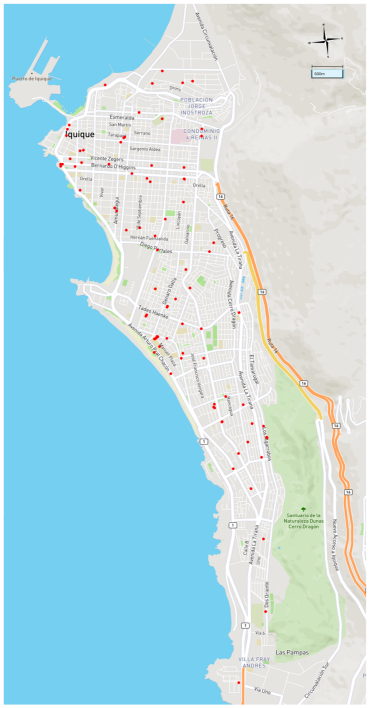
Figure 2: Experimental framework. At left, the Chilean agencies involved during a natural disaster strikes. At right, the distributed computational infrastructure provided by REUNA and national universities. Each computer image represents a cluster of distributed multi-core computers.

During the piloting of applications, the people who participate in the simulacrum access the apps through their mobile phones. There are some applications such as Volunteer Recruitment, used to coordinate the dynamic registration of users to different types of missions generated after a natural disaster, and the application Map of Needs, which reads the constant flow of information published in Twitter, classifies and groups these tweets and then creates a map with the statistics collected. The piloting of applications was carried out in collaboration with agencies and government institutions such as ONEMI, CIREN[3], SERNAGEOMIN[4], HRT[5], different ONGs and nonprofit foundations MovidosxChile and TECHO[6].

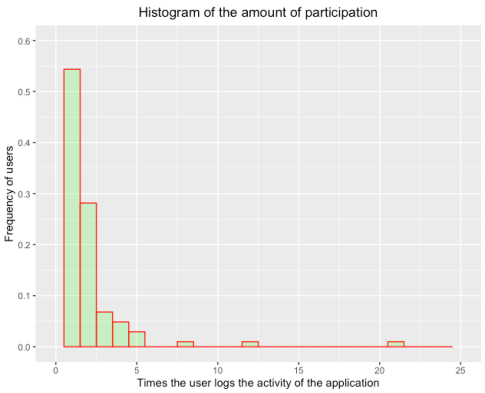
2.2 Data and Traces

We obtained data from two simulacrums of natural disasters coordinated by the ONEMI in two cities: Iquique and Valparaiso. During the first simulacrum of a tsunami in Iquique, we obtained traces containing information about the use of the applications with about 200 operations and the position of the people. These traces gave us a perspective of the behavior of the users and the use of the applications. The second simulacrum was in Valparaiso. We tested our complementary GPS location-system for Android, called "Traza de Evacuación", which allows us to record the geographical location and battery level of 20 users for every 5 minutes. Most of the data collected belonged to students of the University of Valparaiso, moving to security points. Figure 3 shows the data obtained during the first simulacrum. Figure 3(a) shows the initial position of the people using the applications and Figure 3(b) shows the percentage of users accessing the applications, where the x-axis is a simulacrum window time and the y-axis is the percentage of users requests sent to the system.

To increase the number of operations to be tested, we create synthetic traces of requests using an empirical distribution according to Figure 3(b). Therefore, the new trace of requests follows a similar distribution to the real trace of user requests obtained during the simulacrum.



(a)



(b)

Figure 3: (a) Initial position of people at the beginning of the simulacrum. (b) Frequency of users accessing to the application. The x-axis shows the number of times the application was used and the y-axis shows the percentage of users.

[3]https://www.ciren.cl/
[4]http://www.sernageomin.cl/
[5]https://www.hospitaldetalca.cl/
[6]http://www.techo.org/paises/chile/

We also collected data about the computation and communication costs of a computing platform using self-developed application-specific benchmark programs which use as input the synthetic trace of requests. To this end, we instrumented the codes of the social applications to measure the time required to process data in the CPU, to access secondary memory (disk) and to send messages through the network (for more details on benchmarks programs see [2, 7]). We executed those benchmarks on a set of multi-core computers as described in Section 4. We run tests to analyze the behavior of the apps (bottlenecks, delay times, etc.) on high demand. The results of these benchmarks are used to feed the simulators.

2.3 Model and Simulation

We implemented two simulators that describe the right part of Figure 2. First, we implemented an agent-based simulator to simulate the piloting of applications and the behavior of the people in a natural disaster situation. Then, we implemented a discrete event simulator to evaluate the computing infrastructure performance and capacity. Figure 4 shows the general scheme of the performance simulator with the configuration parameters and the outputs. This simulator allows us to estimate the computational cost of the parallel and concurrent tasks executed by the apps. We set the simulator parameters [8] based on the values obtained with the benchmarks.

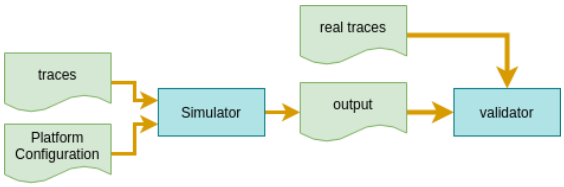


Figure 4: Main components of the simulator used to measure the performance of social apps.

Figure 4 shows that the input parameters are "Platform Configuration" and "traces". These are JSON files read by the simulation program. The file named "Platform Configuration" describes the principal hardware components that will be represented in the simulator. It includes the multi-core processors and their capacity, a subdivision of these multi-cores into virtual machines, a detailed description of the systems and their principal dominant cost operations. The dominant costs are the execution times of the simulated components - Front-End, Back-End and Database (DB). Figure 5 shows all of these components. The second input parameter file named "traces" is also a JSON file divided in two parts. The first part includes the requests sent by the users, that are based on the applications piloting and the natural disasters experts comments. The second part includes the events timeline that allows to trigger particular events based on computing infrastructure experts' comments and the natural disasters simulacrums that we have participated in. Figure 6 shows an example of a crash of a server in a particular zone of the country. It shows all of the "traces" data.

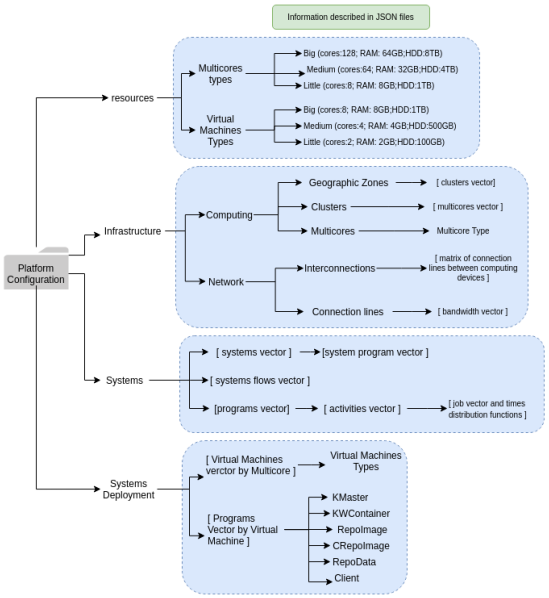


Figure 5: Input data to the simulator: Configuration Platform file.

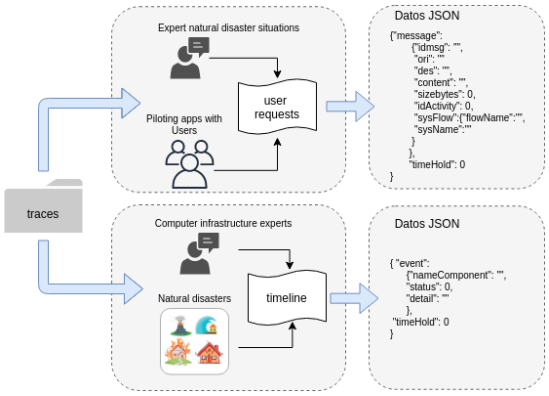


Figure 6: Input data to the simulator: Traces file.

The simulation model uses a processes and resources approach. Processes represent threads in charge of processing high cost operations executed in computational infrastructure. Resources are shared artifacts such as posting lists, data structures for partial results, global variables, RAM memory, cores,and the communication interfaces represented by a tube network that simulates the point-to-point transfer of messages. Our simulator programs are implemented on top of the LibCppSim library [9]. This library manages the creation/removal of co-routines as well as the future event list. The library ensures that the simulation kernel grants execution control to co-routines in a mode of one co-routine at a time. Co-routines are

activated following the sequential occurrence of events in chronological order.

Co-routines represent processes that can be blocked and unblocked at will during simulation by using the operations passivate(), hold() and activate(). When a $hold(\Delta t)$ operation is executed, the co-routine is paused for a given amount of Δt units of simulation time representing the dominant cost of a task. Once the simulation time Δt has expired, the co-routine is activated by the simulation kernel. The dominant costs come from tasks related to ranking of documents, intersection of posting lists and merge of partial results. These costs are determined by benchmark programs implementing the same operations executed on single processors. Additionally, a co-routine executes a passivate() operation to stop itself, indicating it has paused its work. Finally, a co-routine in passivate state can be activated by another co-routine using the activate() operation. Figure 7 shows the class diagram with the simulator main classes.

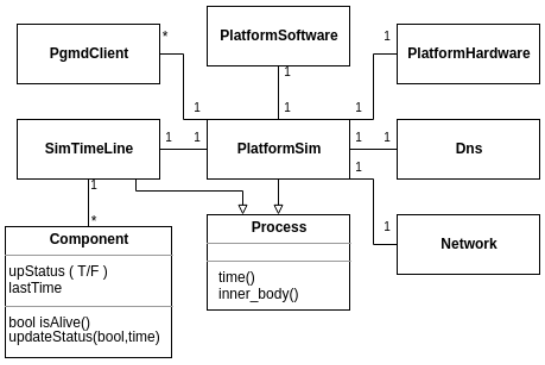


Figure 7: Class diagram of our Discrete event Simulator.

Typically, the implementation of large-scale databases enforces the scheduling policy of one single thread per core to prevent from saturation at processor level. Thus, the incoming requests (operations like search, update, insert and delete) are queued in the assigned thread to receive service which is also reflected in the corresponding simulation.

To simulate the network, the messages are sent over a "tube" network. Each message includes the data, a header with sender and receiver identifiers, and the number of packages forming the message. All input packages go to the same input queue. Benchmark programs are devised to evaluate the cost of different communication patterns such as multicast, broadcast, and point-to-point messages [2].

3 Platform Architecture

In this work we model and simulate a platform composed of three main components: (1) a Front-End, (2) a Back-End and (3) a Data Repository. Each component can be replicated, distributed and partitioned depending on its workload and communication requirements. The Front-End is deployed on the user device, directly interacts with the user and communicates with the Back-End. The Back-End is the host of the applications and communicates with the Data Repository component. It performs data searches and executes transactions. The Data Repository stores the data in non-relational databases like MongoDB or Cassandra. Both, the Back-End and the Data Repository can be deployed on different clusters of servers.

Our platform has a container orchestrator [10] and the container technology [11], allowing the deployment of applications based on microservices, to achieve self-scalable, fault-tolerant and stateless systems, making the infrastructure configuration transparent to the programmer. The specific technologies selected to deploy the social applications are Kubernetes [12] and Docker [13], due to its wide dissemination in the technological community. They also have support in the software communities and they can efficiently deploy the applications in separated geographical areas [14]. In Figure 8 we show the general scheme of the architecture, where Nginx [15] is the entry point, which is the application server and serves as a proxy. The container orchestrator manages and redirects the requests to a container. Finally, the figure shows the interaction between different Back-Ends and different Data Repositories like MongoDB, Cassandra, MariaDB, etc. The orchestrator also starts new instances of the applications that are running in the containers. New instances are started when some applications fails or when more resources are needed to support users demand. To this end, the ImageRepo keeps an undeployed image of all orchestrated containers. As a particular example, Figure 9 shows the deployment of a complete platform distributed among clusters of computers and coordinated by a container orchestrator.

Figure 10 summarized the virtualization technologies used to deploy the social application into the platform which -as mentioned before- is composed of one or more cluster of servers. Our platform supports different types of configurations: No-sharing, virtual machine monitor (VMM), Cloud-computing and Container. In the no-sharing configuration, we install the operating system (O.S.) which manages the resources, the runtime libraries and then we deploy the applications. Another configuration is when the hardware is directly connected to the VMM (hypervisor) regardless of the operating system, indicated by the star symbol (*) in Figure 10. Cloud-computing is not strictly a virtual technology but it provides tools to manage virtual machines and their resources, making tasks such as Live Migration easier, and resizing of virtual machines based on workload. Container virtualization allows creating an architecture based on isolated and scalables microservices deployed in the clusters, but by itself requires manual intervention to manage the containers. However, an orchestrator allows the auto-

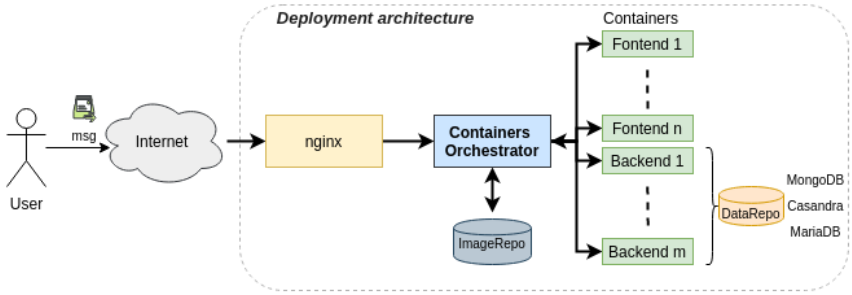


Figure 8: Deployment of the general architecture.

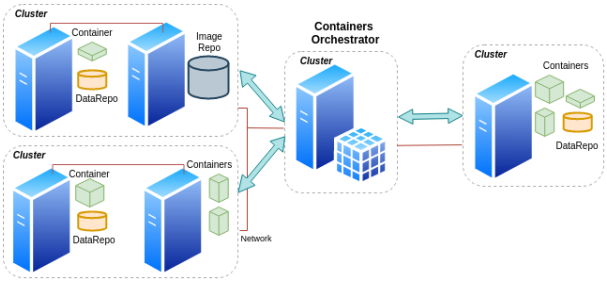


Figure 9: Distributed Platform deployment: an example.

matic management of the containers and the efficient distribution of the workload between them [16].

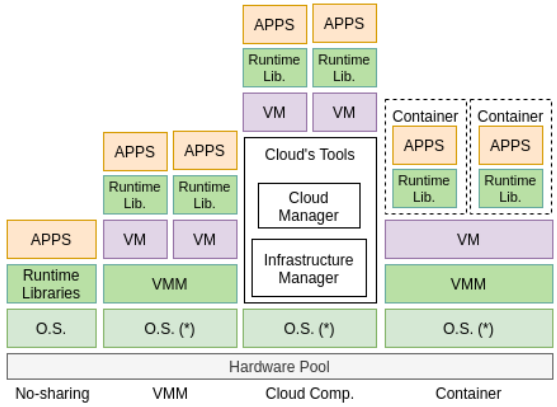


Figure 10: Virtualization technologies used to consolidate a platform composed of servers.

3.1 Social Applications

In this section we describe two applications used during the simulacrum. The application named Ayni[7] is in charge of managing the actions executed by groups of volunteers. The volunteers are gathered through a "BOT" of Telegram (Front-End), implemented with the API available by Telegram. These volunteers sign-in in the ongoing emergency and report their abilities or skills (physics, emotional, knowledge, etc.). Then, an automatic process is used to select the volunteers. Finally, the volunteers can participate in the tasks created as a result of the emergency. The architecture of this system is presented in Figure 11(a). It is composed of two Front-Ends, one Back-End and a Data Repository.

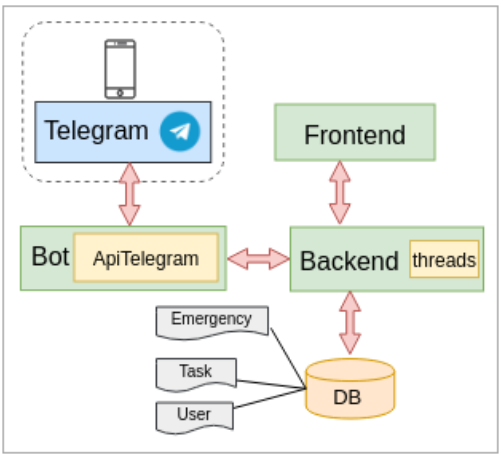
The second application is MongoDB. It is a data repository (NoSql database), based on documents (JSON type data), and collections of these documents. It is scalable, it can be replicated and it supports fault tolerance. Its main components are MongoD, MongoS and the ConfigServer. These components and their connections are shown in Figure 11(b).

The Ayni application is composed of different tasks (like "Register a New User" or "Create Emergency"), and each task executes different activities (such as "Create Profile", "Insert Emergency", "Notify User", etc.). The execution of each task in each application is represented with a DAG (Directed Acyclic Graph). The nodes of the DAG are the processing elements which represent the use of a resource (internal or external) of each application and the edges represent the connections, which can be an internal such as Connection Bus when it comes to applications deployed on the same server, or the Network, when it comes to communication between applications deployed on different servers. Figure 12 shows the DAGs of the "Create Emergency" task executed in the Ayni application. The activities and their set of tasks represent the vertices, and the blue arrows represent the edges or instance of communication. The flow chart of the first row, corresponds to activities executed in the Front-End. The flow chart of the second row, correspond to activities executed in the Back-End and flow chart of the third row correspond to activities executed in the Data Repository.

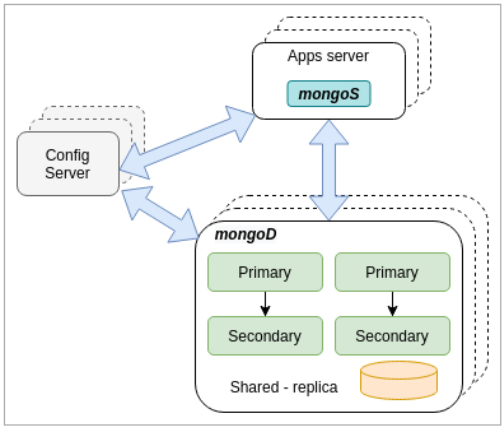
4 Results

We simulate the platform and the applications described in Section 3 by applying the experimental design described in Section 2. Experiments were performed on a AMD Opteron(TM) Processor 6272 with 32 cores and 64 GB of RAM. Table 1 shows some of the tasks executed by the Ayni application at the Front-End (Bot) and at the Back-End (distributed platform) during the simulacrum. For each task we show the

[7]https://citiaps.usach.cl/portafolio/ayni



(a)



(b)

Figure 11: (a) System architecture for the Ayni application. (b) Components of the MongoDB data repository.

number of times it was invoked by the users. The *X* symbol represents that the task is not executed in the Front-End.

Table 2 shows the results (average (avg) running time in milliseconds and the standard deviation (std)) obtained with the benchmark programs executed in the Bot -which includes the Api Telegram (Api TelA)- and in the Back-End -including the database access (DB). In addition, in the Back-End only the Task number 7 used the Api Telegram, obtaining an average time of 1400.896 ms and standard deviation of 522.141. These results were used to set the simulated work in the hold() function of the Libcppsim library.

In Figure 13 we show the execution time in milliseconds of different tasks executed in the real platform and in our simulator. We show that our simulator is capable of reporting similar results to the real distributed platform. The lines in the plot are almost overlapped. We obtained a Pearson correlation of 0.91. A Person correlation value of 1 is total positive linear correlation, 0 is no linear correlation, and -1 is total negative linear correlation. Therefore, our simulator reports values highly correlated with the real ones. We also

Table 1: Tasks executed in the Front-End and the Back-End

Tasks	Front-End	Back-End
(1) Help	162	162
(2) Cancel	194	194
(3) Register New User	136	136
(4) Accept Register	109	107
(5) Cancel Registration	60	60
(6) Get User Description	65	65
(7) Create Emergency	X	137
(8) Create Task	X	132
(9) End Emergency	X	163
(10) End Task	X	150
(11) Get Active Emergencies	140	140
(12) Get Detail My Emergencies	118	115
(13) Get Detail My Tasks	141	141

calculated the root mean square error of the deviation which is a measure of the differences between values obtained by the sequential simulation and the values reported by the parallel simulations. It is defined as Eq (1):

$$\varepsilon_m = \sqrt{\frac{\sum(x_i - \bar{x})^2}{(n(n-1))}} \tag{1}$$

For the values presented in Figure 13 we obtained a root mean square error of 2.1. Then we calculated the relative error (e_r) as Eq (2), which give us a value of 0.04. Therefore, these results confirm that our simulator is capable of reproducing the times of a real application with a very small error.

$$\frac{\varepsilon_m}{\bar{x}} = 0.04 \tag{2}$$

The lines in Figure 14 show the execution time in milliseconds reported by a real execution of the MongoDB and the values reported by our simulator. We executed the insert operations with different size of data ranging from 100 bytes to 16 Mb. The insertions were performed on four different collections. In the first collection we inserted data with sizes between 100 bytes and 4Mb. In the second collection we inserted data with sizes between 4Mb and 8Mb, in the third collection data sizes between 8Mb and 12Mb and in the last collection data sizes ranging from 12Mb to 16Mb. There are peaks representing insert operations when the collection does not exist, in this case MongoDB creates a new collection. We also show results when executing insert operations with a primary key error. That is, we insert a document with a primary key that already exists and therefore the document is not inserted in the collection.

Results presented in Figure 14 show that the simulation achieves good agreement with the results from the actual implementation of MongoDB for insert operations. We computed $e_r = 1.02$ and a Pearson correlation of 0.86.

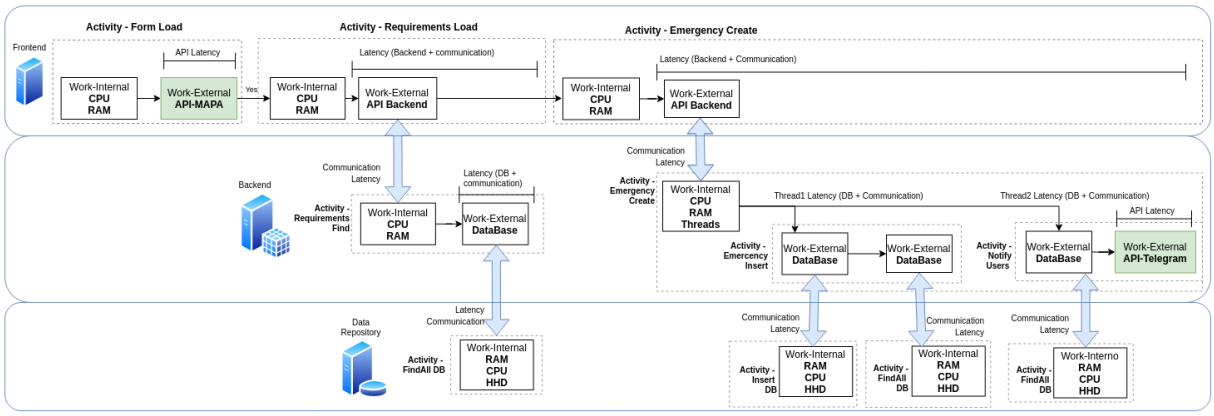


Figure 12: The Create Emergency task: sequence of operations executed in the Front-End, Back-End and in the Data Repository.

Table 2: Average execution time in milliseconds and the standard deviation obtained with benchmark programs running in the Front-End (Bot) and in the Back-End (distributed platform).

	Front-End(Bot)		Back-End	
(Task)		Api		
Metric	Time	TelA	Time	DB
(1)-avg	7.071	419.118	0.022	2.931
(1)-std	-	16.610	0.003	5.267
(2)-avg	419.102	416.266	0.020	-
(2)-std	39.499	39.519	0.004	5.861
(3)-avg	442.138	807.447	8.788	1.850
(3)-std	25.896	478.831	6.000	4.390
(5)-avg	485.891	1153.298	0.206	32.083
(5)-std	89.136	754.766	0.034	19.503
(7)-avg	-	-	0.161	87.846
(7)-std	-	-	0.063	355.488
(8)-avg	-	-	0.073	21.210
(8)-std	-	-	0.015	0.087
(11)-avg	475.070	1428.465	0.586	31.011
(11)-std	29.494	408.329	0.117	18.170
(12)-avg	58.636	1590.329	0.520	58.636
(12)-std	28.623	523.968	0.087	28.623
(13)-avg	30.194	915.439	0.291	11.700
(13)-std	25.640	288.175	0.050	15.039

In Figure 15 we executed a trace composed of 600 tests for each basic CRUD operation (insert, select, update and delete). Results show that our simulator is capable of reproducing the behavior of a real implementation of MongoDB. We obtained a relative error $e_r = 1.37$ and a Pearson correlation of 78%.

5 Conclusions

In this work, we have presented an experimental design for large scale simulations. In particular, we focus on a social application and a data repository application. We have presented the hardware design which uses virtualization technologies to enable scalability, fault tolerance and to make the infrastructure configuration transparent to the programmer. Our approach is based on the piloting of the applications during the simulacrum of a natural disaster scenario. With the collaboration of Chilean agencies, we collected data traces during a simulacrum which were later used to feed our simulators.

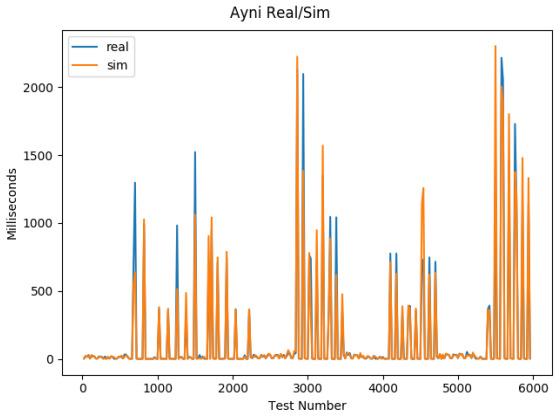


Figure 13: Execution time in milliseconds reported by a real execution of the tasks and the simulator, one example out of twenty examples of 5960 tests

As a case of study, we evaluated two applications. The first one named Ayni, which is used to manage the volunteers and the emergency campaigns. The second application is the MongoDB used to store the statistics and information about the tasks assigned and completed by the volunteers. We validated our simulators against real data. The results showed that our simulators are able to estimate the execution times of the tasks executed in each application with a minimum error. Moreover, the results obtained by our simulators are highly correlated with the real data.

As future work, we intend to combine our simulation framework with data mining and machine learning

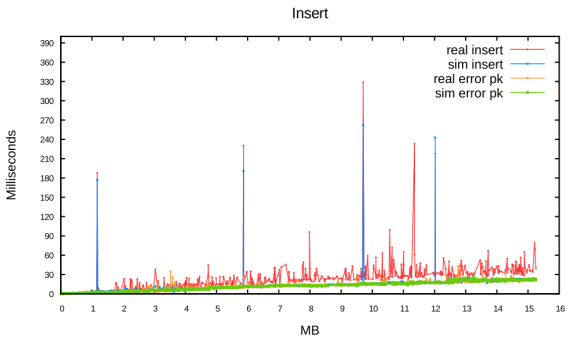


Figure 14: Insert task: Time in milliseconds obtained with a real execution of MongoDB and our simulator.

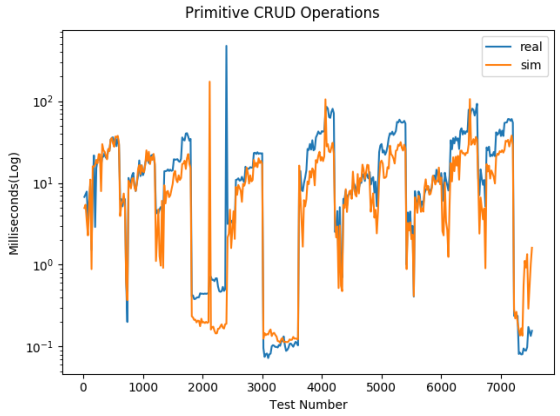


Figure 15: Execution times in milliseconds reported by a real implementation of MongoDB running on a single computer and our simulator for a trace with 7520 CRUD operations

techniques to develop a capacity planning model. We also plan to include agent-based simulation combined with Generative Adversarial Networks to study the behavior of people during the evacuation and obtain situation-maps of the affected zones.

This work was developed as part of a larger project that aims to deliver an Emergency Management Support Platform through Social Applications. To this end, different tasks and developments (like different agent-based and discrete event-based simulators, the design, implementation and testing of applications) are performed in parallel by different research groups in the context of large-scale disaster simulations, involving a methodology for Fault Tolerant deployment, among others.

Competing interests

The authors have declared that no competing interests exist.

Authors' contribution

LVC and GOL deployed the computational platform, they programmed the discrete event simulator, executed the tests and obtained the experimental results. GA implemented the agent-based simulator to analyze the simulacrum traces. Finally, VGC and MM conducted the research project. All authors read and approved the final manuscript.

Acknowledgements

This work has been partially funded by CeBiB and project FB00001.

References

[1] V. Gil-Costa, J. Lobos, A. Inostrosa-Psijas, and M. 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*, pp. 288–307, Springer, 2012.

[2] V. Gil-Costa, M. Marin, A. Inostrosa-Psijas, J. Lobos, and C. Bonacic, "Modelling search engines performance using coloured petri nets," *Fundamenta Informaticae*, vol. 131, no. 1, pp. 139–166, 2014.

[3] J. Allspaw, *The art of capacity planning: scaling web resources*. " O'Reilly Media, Inc.", 2008.

[4] J. Rogstadius, M. Vukovic, C. A. Teixeira, V. Kostakos, E. Karapanos, and J. A. Laredo, "Crisistracker: Crowdsourced social media curation for disaster awareness," *IBM Journal of Research and Development*, vol. 57, no. 5, pp. 4–1, 2013.

[5] M. Alaniz, S. Nesmachnow, B. Goglin, S. Iturriaga, V. G. Gosta, and M. Printista, "Mbspdiscover: An automatic benchmark for multibsp performance analysis," in *Latin American High Performance Computing Conference*, pp. 158–172, Springer, 2014.

[6] R. Copeland, *MongoDB Applied Design Patterns: Practical Use Cases with the Leading NoSQL Database*. " O'Reilly Media, Inc.", 2013.

[7] A. Inostrosa-Psijas, V. Gil-Costa, R. Solar, and M. Marín, "Load balance strategies for devs approximated parallel and distributed discrete-event simulations," in *2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pp. 337–340, IEEE, 2015.

[8] M. Marin, V. Gil-Costa, C. Bonacic, and A. Inostrosa, "Simulating search engines," *Computing in Science & Engineering*, vol. 19, no. 1, pp. 62–73, 2017.

[9] M. Marzolla *et al.*, "libcppsim: a simula-like, portable process-oriented simulation library in c++," in *Proc. of ESM*, vol. 4, pp. 222–227, Citeseer, 2004.

[10] G. Osborne and T. Weninger, "Ozy: a general orchestration container," in *2016 IEEE International Conference on Web Services (ICWS)*, pp. 609–616, IEEE, 2016.

[11] T. Adufu, J. Choi, and Y. Kim, "Is container-based technology a winner for high performance scientific applications?," in *2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 507–510, IEEE, 2015.

[12] K. Hightower, B. Burns, and J. Beda, *Kubernetes: up and running: dive into the future of infrastructure*. " O'Reilly Media, Inc.", 2017.

[13] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux journal*, vol. 2014, no. 239, p. 2, 2014.

[14] F. Rossi, V. Cardellini, F. L. Presti, and M. Nardelli, "Geo-distributed efficient deployment of containers with kubernetes," *Computer Communications*, 2020.

[15] W. Reese, "Nginx: the high-performance web server and reverse proxy," *Linux Journal*, vol. 2008, no. 173, p. 2, 2008.

[16] H. Falatiuk, M. Shirokopetleva, and Z. Dudar, "Investigation of architecture and technology stack for e-archive system," in *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T)*, pp. 229–235, IEEE, 2019.

Citation: L. Veas-Castillo, G. Ovando-Leon, G. Astudillo, V. Gil-Costa and M. Marín. *Experimental Framework to Simulate Rescue Operations after a Natural Disaster*. Journal of Computer Science & Technology, vol. 20, no. 2, pp. 62-71, 2020.
DOI: 10.24215/16666038.20.e07.
Received: March 31, 2020 **Accepted:** August 30, 2020.