Specifying the Experimental Scenarios for Simulated Cloud Studies

Simon Bihel

Dept. of Computer Science, ENS Rennes
simon.bihel@ens-rennes.fr

July 15, 2016

Abstract

Cloud computing is a model that makes available infrastructures, platforms and software with a pay-as-you-go subscription. It aims to reduce the cost with a layer of visualization that allows virtual resources to be dynamically adjusted and occupied on-demand. The problem of using the minimal resources for the current demand/usage is still a research challenge that spans all layers and applications. This dynamic management of clouds is called cloud elasticity. To evaluate research work done on cloud elasticity simulation can be used and presents some advantages. While the simulation of cloud structures are already possible there is a lack of workload generation which is essential to evaluate works supposed to deal with fluctuating workload. This paper presents a way of describing workloads using tasks that are repeated over time with parameters that can be modified over time. It also shows that this proposition fits the needs of past works.

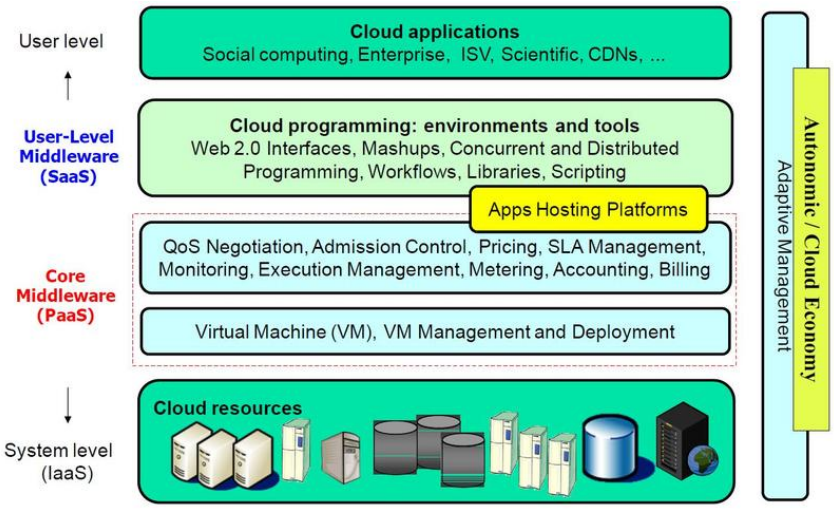
Index terms— Simulation; Workload; Cloud elasticity

1 Introduction

Nowadays clouds are used for a lot of server-based applications. From websites to scientific computing, it allows apps creators to avoid managing their own servers. Based a well defined and separated business model the cloud structure is composed of layers where each layer uses the precedent one and provides a service to the next one. The services provided by a layer is negotiated (e.g. to determine the pricing) and levels of quality have to be met. Of course the goal is to meet these levels of quality while minimizing the costs like the usage of the bottom layer, energy consumption... Research is being done to tackle this problem. The domain we are particularly interested in is the one that deals with fluctuating usage. In this cases a dynamic management is required to minimize costs at all time.

As clouds are complex structures these works have to be evaluated. Theory is not enough as so many problems are NP complete. One way is to use a real cloud and deploy the work that should be tested, and then generate somewhat artificial workload for the app. Another way of doing it would be to simulate the cloud and the workload. This kind of evaluation has often

Figure 1: Cloud architecture.
http://cloud-simulation-frameworks.wikispaces.asu.edu/



been used for grids, which can be seen as the ancestor of clouds. Among the different advantages of simulations, simplicity would come up on top.

For now it is possible to simulate clouds infrastructures but tasks are still seen as individual and independent computing task. For elastic clouds research it is essential to generate authentic workload and there is currently no way of manipulating naturally workloads for simulations. During this internship we worked on writing an API for SimGrid and this paper presents the work done.

In 2 we give a more detailed presentation of clouds and their uses. In 5 we study the work done on simulation, workload generation and define in more details the needs for a workload generator based on past work on cloud elasticity. 3 presents the actual contribution and its use. 4 provides an evaluation based on performances and expressiveness to make sure the contribution fits the needs. 6 concludes on whether this contribution will help future research.

The main contributions of this internship are: (i) a categorization and generalization of experimental needs for this domain; (ii) an implementation of an API to evaluate this categorization.

2 Background

2.1 Cloud Computing

Figure 1 describes the global architecture of a cloud. It is split in different layers and each layer has a specific role in the model of clouds. On the lowest part there is the physical resources (e.g. data centers) with Infrastructure as a Service model (IaaS). The pricing is based of the resources available. Then comes the layer of virtualization and performances negotiation called Platform as a Service (PaaS). Dealing with Virtual Machines (VMs also called hosts) allows a cleaner sharing of resources and makes it easier answering the users' demands (e.g. deploying more VMs). The pricing depends of multiple factors, like the Quality of Service (QoS) for the

quality of the network, the Service Level Agreement (SLA) for the faults rate, handling and responsibilities... On top of that is the layer for users' cloud tools with Software as a Service (SaaS). These tools will allow the user that writes cloud applications to manage resources, run their code... This architecture and organization is a real enhancement as before that users would have to run their own application stack (e.g. managing servers, databases, etc.). In addition to that the pay-as-you-go business models makes it even more convenient for the user to react, increase capacities, try new features, get more control on certain layers (e.g. for optimization purpose), etc.

Cloud applications will have fluctuating workload over time. For example with a website server the usage will be bigger during the day or during a short period of time because of a viral cultural event. Resources should thus be managed dynamically. Also, physical resources can encounter problems which makes them unavailable. Because of all these constraints the SLAs and QoS agreements are not satisfied easily and 100% availability is never a thing. On top of that every actor wants to meet their obligations with a minimal cost. All these questions of availability and cost are current research problems. In particular we are interested in the works that tackle problems related to fluctuating and dynamic usage of cloud applications and resources. The ability for a cloud infrastructure to adapt to a dynamic workload is called cloud elasticity.

There are some generic elastic actions that are used for common problems. The act of deploying more VMs (and thus having more resources overall) is called scaling up (and the convert is scaling down). The act of moving VMs to a different location is called scaling out. This is used for example when time passes by and users come from different countries/continents.

2.2 Typical Cloud Studies

Examples of research done on cloud elasticity can be a better scheduler algorithm, a caching system to react more precisely and faster to common workload variations, extracting characteristics of real-world workload to produce synthetic ones for experiments, etc. Practical evaluations have to be done because theory is not enough to study a cloud system/software as there are too many NP complete problems. Then there are multiple choice with running the product of the research work on a real cloud platform, emulating one, or simulating one.

[1] has categorized works on cloud elasticity and allows to see which elements of a cloud infrastructure (platform or application/software) are impacted. As it is for now most research works are evaluated on real clouds. It is interesting for a distributed systems simulator to search what is needed for simulating cloud elasticity. If it is shown that research works on cloud elasticity can be evaluated on a simulator they would benefit from simple setups, cost reduction, reproducible experiments, trust in results, etc. These last ones are particularly important when comparing two algorithms, which is a good part of this domain.

In this survey proposals are categorized as follows. The scope is about what elements of a cloud the proposals work on. It can be the management of VMs, allocation of resources... Then there is the purpose of the proposal. Enhancing the *performances* (to meet the SLA), reducing the *energy consumption* footprint, being *available* when needed and reducing the overall *cost*. Another dimension is the decision making. This is what a proposal add to an existing cloud to reach its goal. In addition to the scope there is the elastic actions performed by the proposals. As the scope is about what elements of a cloud are concerned, the elastic action is about what is done to them. Then there is the provider dimension that tells if there is only one provider or

multiple ones. At last there is the method used by the proposal to evaluate itself, through real cloud, simulation or emulation.

The survey gives a good overview on what elements of a cloud are manipulated to achieve cloud elasticity. At the time of writing no piece of evidence has been found to prove the opposite.

2.3 Typical Experimental Methodologies

Clouds experiments follow traditional experiments steps: what are they evaluating, setup (resources, platform), scenario (what actions are done during the experiment), the results and their analysis.

About simulators and particularly SimGrid. At first they are discrete event simulators. They have a queue of events and jump between following events thus not doing anything when the state of simulation remains constant. For cloud or grid simulator one of these events will be tasks (AKA jobs, gridlets, cloudlets, etc.) that will require a certain amount of flops (or another resource unit) to be completed. As they don't really execute these computing tasks, the completion of a task will be an event and if there is another task queued up it will start. If the simulated platform can handle easily the tasks then the simulator will have no problem to simulate this smooth execution. But if too much tasks are queued up the simulator will use a lot of memory to keep track of every element's state.

3 Contribution

As the proposals are on reacting to variating usage, simulators need a way to express this fluctuating workload. We worked on elastic tasks (ETs) that model tasks that are triggered regularly and with a usage that fluctuates over time. The work was done on SimGrid [2]. The code is available here: https://github.com/sbihel/internship_simgrid. It was written as a plug-in on top of the S4U interface which is intended to be the core API. Elastic tasks are objects and are the only things the user has to manipulate.

3.1 Scientific Needs

One key notion is that we use a discrete representation of workloads. Even if workloads are generated by a discrete amount it is easier for cloud elasticity researchers to see the workload as a continuous function of flops. If we still decided to describe workloads as continuous execution of tasks (or jobs...) is mainly because everything was already there in SimGrid and simulators are event-based. Also often times when visualizing the workload along side with the number of hosts for example, people chose a time interval and print the total number of requests for each interval. Finally talking about taskrates instead of floprates makes it more generic as there are more resources than just computing ones. With this representation the various elastic actions come naturally as we explain in the following paragraphs.

An elastic task can repeat a certain task that we call microtask. The user provides a rate of triggering per second and the flops required and then over time multiple identical microtasks will be created and executed. An elastic task can have multiple hosts to split the workload and

there will be a cycling shifting between hosts when creating microtasks, keeping one host for one microtask.

For horizontal and vertical scalings, they can be performed by modifying the list of hosts of an ET. To scale up you just have to add hosts and to scale out you have to replace this list with different hosts. Concerning workflows of tasks you can set the output function with a function that has access to ETs you want to trigger and just trigger them in the function, with possibly a multiplicative effect of the workload.

A way of evaluating the usage of hosts is also needed. There are two mechanisms, one which we will call online and another called offline. The online is a threshold. Detecting over-usage of an host is done by using the resources requirements of tasks and their completion time required. When the threshold is crossed a user-provided function is executed (to increase the number of hosts for example) and the extra queue up tasks can be dealt with. The offline mechanism is more passive as it is just a feedback to the user of the time that passed before a task has been executed.

3.2 Technical Needs

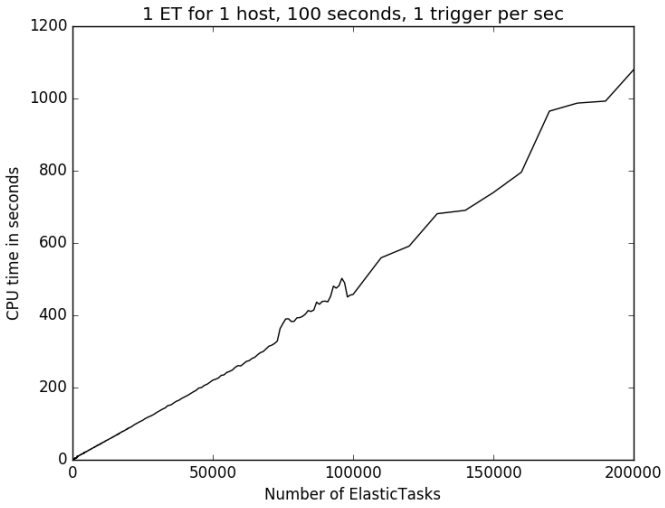
An output function can be provided to an elastic task and this function will be executed after each microtask that has ended. This has multiple usages. It allows the description of workflows of tasks. As microtasks only generate computing workload, output functions can be used to have different types of workloads like network usage, disk access (which can be simulated only by seeing it as a particular computing resource at the moment), and basically anything possible with SimGrid.

It is also useful to study the behavior of a system dealing with real workload. For that an elastic task can be given a file of timestamps and it will trigger/generate a microtask for each time stamp. Another way of simplifying the workload generation for the user is to use a statistical law. It is also called generating law as found in the network simulator NS-3. The timespan or delay between two triggers of the ET is determined through a common law as Poisson or through a user provided one.

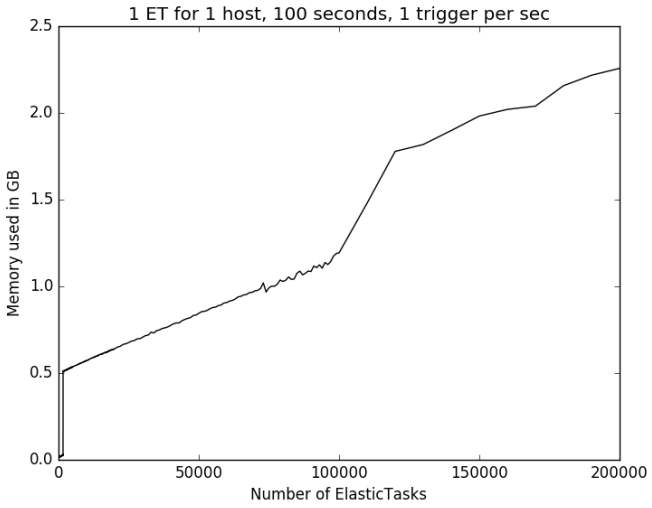
For detailed platforms description it is a core feature of SimGrid which allows to have multiple providers, topologies, hosts (understand VMs here), bandwidths, etc. Platforms are not specific to clouds and it shows again that as clouds are an extension of grids, our work on cloud simulations are an extension of grid simulations. In the end as we are working on cloud elasticity we are only interested in the dynamic part of simulations not the static one.

4 Evaluation

The contribution has been evaluated on the predefined criteria. We first did an experiment for raw performances. Then we used real traces from WorldCup 98 data access logs [3] which are often used. After that we evaluated the expressiveness and functionalities. All experiments have been executed on a MacBook Pro with an Intel Core i5 and 8GB of RAM.



(a) Raw performances CPU time.



(b) Raw performances Max Memory.

Figure 2: Raw performances.

4.1 Raw performances

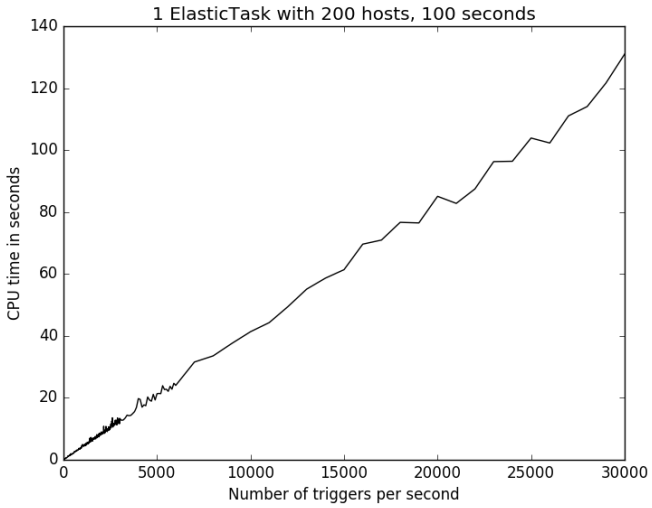
We evaluate raw performances to make sure that users can do significant experiments. Basically we just want to see if the time and memory used grow linearly with the number of micro tasks so that it can be used on a regular computer and allows multiple quick experiments.

Two different experiments were made. During first one the number of elastic tasks grows while keeping a ratio of one ET per host and one trigger per second. The sole purpose of this is to see the performances because a typical cloud app will use about 15 ETs. For the second experiment it's the number of triggering per second that grows while keeping only one ET with 200 hosts.

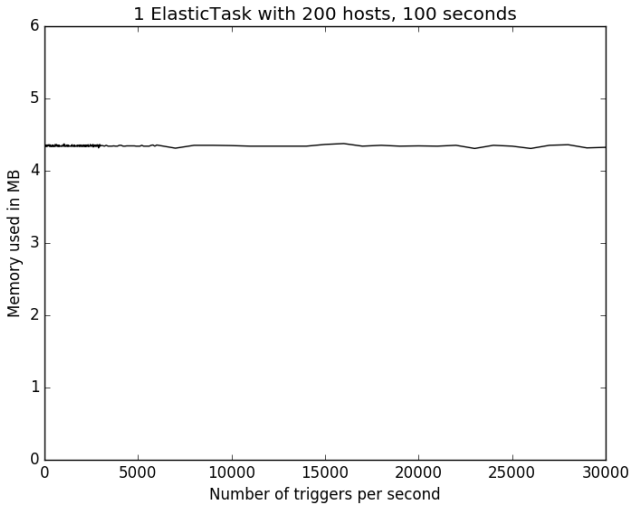
Experiment 1 Figure 2a shows the CPU time (user + system time) while Figure 2b shows the maximum memory used. To get the x axis with the total number of microtasks you just have to multiply by 100. The platform was upgraded two times, at 1,600 from 2,000 hosts to 100,000, and then at 100,000 from 100,000 to 200,000. We can see that the deployment of the platform has nearly no impact on the time but account for about a half of the memory used if there is as much ETs as hosts. Apart from that, time and memory grows linearly depending of the ETs amounts and operations done.

Experiment 2 Figure 3a shows the CPU time and Figure 3b shows the maximum memory used. To get the x axis with the total number of microtasks you just have to multiply by 100. As we don't touch the platform or application structure (we just touch its usage) the memory remains constant. As seen before the time is linear with the number of microtasks and with about the same growing speed.

What will create workload are the elastic tasks and the two way of increasing the workload (increasing the number of microtasks) are to add more elastic tasks or increase the ratio of triggering. For both this cases time is linear on the total number of microtasks and memory



(a) Raw performances CPU time.



(b) Raw performances Max Memory.

Figure 3: Raw performances.

Trace (# of requests)	Time	Max Memory
test (1,000)	0.14s	13,484 KB
day 10 (1,522,111)	50.82s	13,876 KB
day 20 (6,326,015)	229.62	14,472 KB

Figure 4: Execution of WorlCup 98 website traces.

will increase depending on the size of the platform and app architecture while always being moderate for an present-day computer. We did not focused on the amount of flops a microtask has because in the end the simulator is just doing a subtraction. Though you have to be careful to always have enough resources because then microtasks are delayed and start to stack up. At some point the simulator will have a hard and in the end it might crash your computer because of an excessive usage of memory.

4.2 Real Traces

Evaluating the use of traces feature has two goals. Show that it works and similarly to 4.1 we want to show that evaluations done in papers are possible in a simulation.

For that we used real traces from WorldCup 98 data access logs [3] with a platform of 2,000 host and enough flops. Figure 4 shows the results of a few executions. Again time and memory usage are reasonable for days of real time simulated (even though these traces are getting old and might not be as big as nowadays requests logs). Adding more hosts changed nothing except for the deployment of the platform as we have seen in 4.1.

	Implemented	[4]	[5]	[6]
Horizontal scaling	YES	✓	✓	✓
Vertical scaling	YES	✓	✓	✓
Threshold	NO			
Usage feedback	NO	×	×	×
Workflow	YES	✓		
Traces	YES	✓	✓	✓
Constant rates	YES			
Generating law	NO			

Figure 5: Functionnal requirements of several scientific studies.

4.3 Functionalities

As we have determined the needs for cloud elasticity works experiments, we have to evaluate which ones are met. Apart from performances which are addressed in 4.1 and 4.2 there are left elastic actions, workload generation features and system feedback. For various papers, Figure 5 shows which needs are met and which are not.

5 Related Work

At the moment no simulator article talks about dynamic workload. On the other hand in the code of DCsim [7] there was an interactive task and in the code of CloudSim [8] there was an host with dynamic workload. There are some tools to generate artificial workload like [9] and they generally follow the following steps. They have a thread that acts like clients/users and it makes request over time and simulate thinking times of the users.

Five papers in the survey used simulations. They used discrete event simulators (home-made or OMNeT++), used benchmarks like SPECjEnterprise2010 to have close-to-reality hosts, and run real traces.

6 Conclusion

During this internship we've studied which actions were taken by elastic clouds mechanisms. Then we searched what was done in simulations used for evaluations and other evaluations to come with a contribution that meets the needs of researchers. To prove that the contribution was good enough we evaluated it on some criteria.

For future works we would need a callback for the user to set a certain response time (a threshold) and allow him to react accordingly if this QoS level is not met. For that we would just need to either set an alarm and turn it off once the workload has been executed and if it is not then it stops it and execute a user's function. That would be an online solution and an offline one would be to just see how much time the workload took to be executed and let the user react accordingly. Another feature that would be really useful is a generating law. For an elastic task the date for the next triggering would be based on a statistical function. That would

get us even closer to simple and simplistic setup for experiments. Finally, the microtasks we have used are computing tasks. Users can set the flops amount to 0 and add more specific tasks in an output function but this is too hack-y. Right now in SimGrid there is only computing and network tasks and you can also simulate disk usage by seeing it as a computing resource.

On a more personal note, this internship has allowed me to discover what a simulator was, what a research code looked like along with its development, the life in a research environment, and many more.

Acknowledgment

Thanks to Martin Quinson and Anne-Cécile Orgerie for guiding and advising me during this internship. Also many thanks to Inria for hosting me.

References

[1] A. Naskos, A. Gounaris, and S. Sioutas, *Cloud Elasticity: A Survey.* Cham: Springer International Publishing, 2016, pp. 151–167. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-29919-8_12

[2] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter, "Versatile, scalable, and accurate simulation of distributed applications and platforms," *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 2899–2917, Jun. 2014. [Online]. Available: http://hal.inria.fr/hal-01017319

[3] "World cup 98 data access logs," http://ita.ee.lbl.gov/html/contrib/WorldCup.html, accessed: 2016-07-13.

[4] N. Vasić, D. Novaković, S. Miučin, D. Kostić, and R. Bianchini, "Dejavu: accelerating resource allocation in virtualized environments," in *ACM SIGARCH computer architecture news*, vol. 40, no. 1. ACM, 2012, pp. 423–436.

[5] L. R. Moore, K. Bean, and T. Ellahi, "A coordinated reactive and predictive approach to cloud elasticity," 2013.

[6] J. Hwang and T. Wood, "Adaptive performance-aware distributed memory caching," in *Proceedings of the 10th International Conference on Autonomic Computing (ICAC 13)*, 2013, pp. 33–43.

[7] M. Tighe, G. Keller, J. Shamy, M. Bauer, and H. Lutfiyya, "Towards an improved data centre simulation with dcsim," in *Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013).* IEEE, 2013, pp. 364–372.

[8] 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.

[9] P. Bodik, A. Fox, M. J. Franklin, M. I. Jordan, and D. A. Patterson, "Characterizing, modeling, and generating workload spikes for stateful services," in *Proceedings of the 1st ACM symposium on Cloud computing*. ACM, 2010, pp. 241–252.