# Generating fluctuating workload for cloud elasticity simulations

Simon Bihel

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

July 13, 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

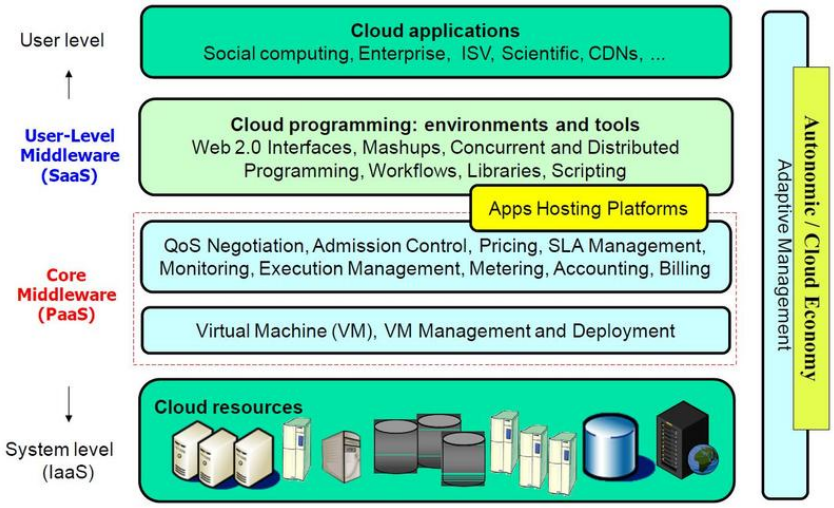
Nowadays clouds are used for a lot of server-based applications. From websites to scientific computing, it allows apps creators to avoid managing their own servers. Because of a well defined business model the cloud structure is composed of layers where each layer uses the precedent one and provides 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 where dynamic management is required to minimize costs at all time.

As clouds are complex structures these works have to be evaluated. 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 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

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




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:

- categorization and generalization of experimental needs for this domain, particularly workloads;
- implementation of an API to evaluate this way of seeing workloads.

## 2 Background

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...

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 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 negotiations are not satisfied easily and 100% availability is never a thing. On top of that every actor wants to meet the 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. 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.

[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 cost reduction, reproducible experiments, trust in results...

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. No clue have been found proving the opposite at the time of writing. As the proposals are on reacting to variating usage, simulators need a way to express this fluctuating workload. We worked on elastic tasks that model tasks that are triggered regularly and with a usage that fluctuates over time.

# 3 Contribution

The work was done on SimGrid [6]. 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.

An elastic task can repeat a certain task that we will 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.

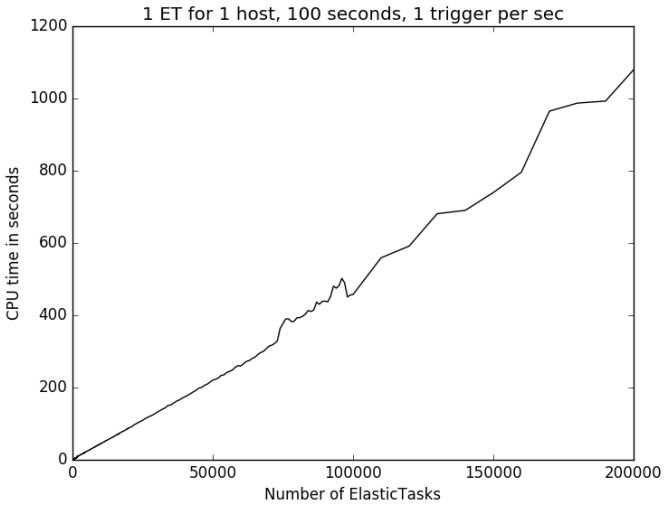
An output function can be provided to an elastic task and this function will be executed after each microtask that has ended. This has multiple usages. It allows the description of workflows of tasks. As microtasks only generate computing workload, output functions can be used to have different types 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.

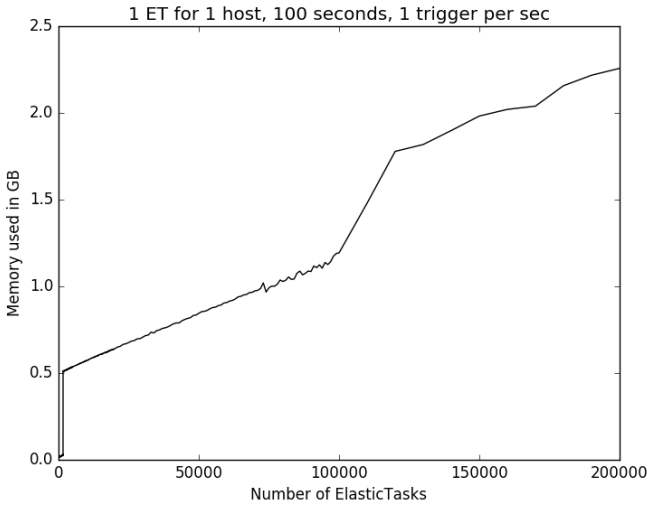
For detailed platforms description it is a core feature of SimGrid which allows to have multiple providers, topologies, hosts (VMs for us), bandwidths...

# 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 [7] 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

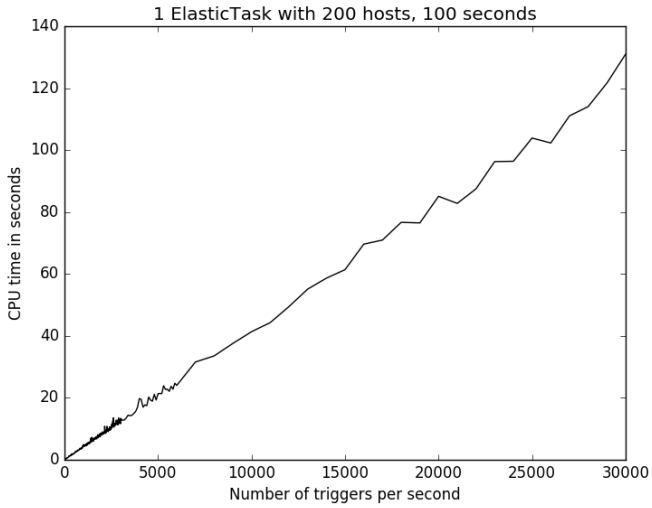
We evaluate raw performances to make sure that users can do significant experiments. Basically we just want to see if the time and memory used grow linearly with the number of micro tasks so that it can be used on a regular computer and allows multiple quick experiments.

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.

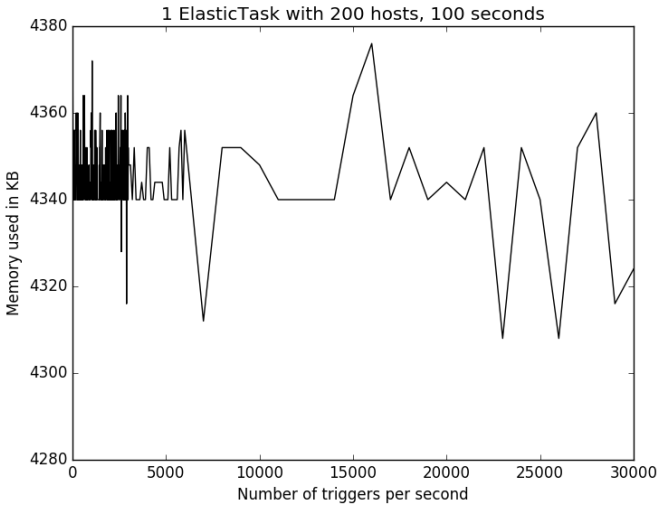
**Experiment 1** Figure 2a shows the CPU time (user + system time) while Figure 2b shows the maximum memory used. To get the $x$ axis with the total number of microtasks you just have to multiply by 100. The platform was upgraded two times, at 1,600 from 2,000 hosts to 100,000, and then at 100,000 from 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 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.




(a) Raw performances CPU time




(b) Raw performances Max Memory

Figure 3: Raw performances

## 4.2 Real traces

For that we used real traces from WorldCup 98 data access logs [7] with a platform of 2,000 host and enough flops. After translating the requests log as a timestamps file we tested with the test file of 1,000. It took 0.14 seconds of CPU time and 13,484 KB. Then with the same platform we used the days 10 and 20 with respectively 1,522,111 and 6,326,015 requests. The first one took 50.82 seconds of CPU time and 13,876 KB of memory and the later took 229.62 seconds and 14,472 KB. Adding more hosts changed except for the deployment of the platform as we have seen in 4.1.

| 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 |

## 4.3 Functionalities

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.

| | This API | [5] |
|---|---|---|
| Horizontal scaling | OK | ✓ |
| Vertical scaling | OK | ✓ |
| Workflow | OK | |
| Threshold | NO | |
| Traces | OK | |
| Constant rates | OK | |
| Generating law | NO | × |

# 5 Related work

According to the classification of the survey, a simulator should allow the manipulation of scopes, the evaluation of the different purposes, make possible the elastic actions and allow multiple providers.

At the moment no simulator article talks about dynamic workload. On the other hand in the code of DCsim [2] there was an interactive task and in the code of CloudSim [3] there was an host with dynamic workload. There are some tools to generate artificial workload like [4] 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.

On a less technical note, workloads are generally seen as a number of requests per a certain interval of time.

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.

# 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] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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

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