Simulation on Cloud Computing Infrastructures of Parametric Studies of Nonlinear Solids Problems

Elina Pacini[1], Melisa Ribero[1,2], Cristian Mateos[3], Anibal Mirasso[2], Carlos García Garino[1,2]

[1] Instituto para las Tecnologías de la Información y las Comunicaciones (ITIC) – UNCuyo, Mendoza Argentina {epacini, cgarcia}@itu.uncu.edu.ar
[2] Facultad de Ingeniería, UNCuyo, Mendoza, Argentina, melisaribero@yahoo.com.ar, aemirasso@uncu.edu.ar
[3] ISISTAN - CONICET. Tandil, Buenos Aires, Argentina, cmateos@conicet.gov.ar

Abstract. Scientists and engineers are more and more faced to the need of computational power to satisfy the ever-increasing resource intensive nature of their experiments. Traditionally, they have relied on conventional computing infrastructures such as clusters and Grids. A recent computing paradigm that is gaining momentum is Cloud Computing, which offers a simpler administration mechanism compared to those conventional infrastructures. However, there is a lack of studies in the literature about the viability of using Cloud Computing to execute scientific and engineering applications from a performance standpoint. We present an empirical study on the employment of Cloud infrastructures to run parameter sweep experiments (PSEs), particularly studies of viscoplastic solids together with simulations by using the CloudSim toolkit. In general, we obtained very good speedups, which suggest that disciplinary users could benefit from Cloud Computing for executing resource intensive PSEs.

Keywords: Parameter Sweep, Viscoplastic Solids, Cloud Computing

1 Introduction

Parameter Sweep Experiments, or PSEs for short, is a very popular way of conducting simulation-based experiments among scientists and engineers through which the same application code is run several times with different input parameters resulting in different outputs [1]. From a software perspective, most PSEs are cluster friendly since individual inputs of an experiment can be handled by independent tasks. Therefore, using a software platform such as Condor [2], which is able to exploit the distributed nature of a computer cluster, allows these tasks to be run in parallel. In this way, not only PSEs execute faster, but also more computing intensive experiments can be computed, and hence more complex simulations can be performed. This idea has been systematically applied to execute PSEs on Grid Computing [3], which are basically infrastructures that connect clusters via wide-area connections to increase computational power.

A recent distributed computing paradigm that is rapidly gaining momentum is Cloud Computing [4], which bases on the idea of providing an on demand computing infrastructure to end users. Typically, users exploit Clouds by requesting from them one or more machine images, which are virtual machines running a desired operating system on top of several physical machines (e.g. a datacenter). Among the benefits of Cloud Computing is precisely a simplified configuration and deployment model compared to clusters and Grids, which is extremely desirable for disciplinary users.

In this paper, we will show the benefits of Cloud Computing for executing PSEs through a case study. The application domain under study involves PSEs of viscoplastic solids, which explore the sensitivity of solid behavior in terms of changes of certain model parameters (viscosity parameter η, sensitivity coefficient, and so on). In this sense parametric studies previously discussed for imperfections [5] are extended for material parameters case, which were computed on Clouds by using the CloudSim simulation toolkit [6]. Results show that by executing our experiments in our simulated Clouds, depending on the configured computational capabilities and the scheduling policy being used, near-to-ideal speedups can be obtained.

The next Section provides more details on Cloud Computing and the motivation behind considering this distributed computing paradigm for executing PSEs. The Section also explains CloudSim. Section 3 describes our case study. Later, Section 4 presents the results obtained from processing these problems on Cloud Computing. Finally, Section 5 concludes the paper and describes prospective future works.

2 Background

Running Parameter Sweep Experiments (PSE) [1] involves many independent jobs, since the experiments are executed under multiple initial configurations (input parameter values) several times, to locate a particular point in the parameter space that satisfies certain criteria. Interestingly, PSEs find their application in diverse scientific areas like Bioinformatics, Earth Sciences, High-Energy Physics, Molecular Science and even Social Sciences.

When designing PSEs, it is necessary to generate all possible combinations of input parameters, which is a time-consuming task. Besides, it is not straightforward to provide a general solution, since each problem has a different number of parameters and each of them has its own variation interval. Another issue, which is in part a consequence of the first issue, relates to scheduling PSEs on distributed environments, which is a complex activity. Therefore, it is necessary to develop efficient scheduling strategies to appropriately allocate the workload and reduce the computation time.

In recent years Grid Computing [3] and even more recently Cloud Computing technologies [4] have been increasingly used for running such applications. PSEs are well suited for these environments since they are inherently parallel problems with no or little data transfer between nodes during computations. Since many applications require a great need for calculation, these applications have been initially addressed to dedicated High-Throughput Computing (HTC) infrastructures such as clusters or pools of networked machines, managed by some software such as Condor [2]. Then, with the advent of Grid Computing new opportunities were available to scientists, since Grids offered the computational power required to perform large experiments. Despite the widespread use of Grid technologies in scientific computing, some issues still make the access to this technology not easy for disciplinary users. In most cases scientific Grids feature a prepackaged environment in which applications will be executed. Then, specific tools/APIs have to be used, and there could be limitations on the hosting operating systems or the services offered by the runtime environment. On the other hand, although Grid Computing favors dynamic resource discovery and provision of a wide variety of runtime environments for applications, in practice, a limited set of options are available for scientists, which are not in addition elastic enough to cover their needs. In general, applications that run on scientific Grids are implemented as bag of tasks applications, workflows, and MPI (Message Passing Interface) [7] parallel processes. Some scientific experiments could not fit into these models and therefore have to be redesigned to exploit a particular scientific Grid.

2.1 Cloud Computing: Overview

Cloud Computing [4], the current emerging trend in delivering IT services, has been recently proposed to address the aforementioned problems. By means of virtualization technologies, Cloud Computing offers to end users a variety of services covering the entire computing stack, from the hardware to the application level, by charging them on a pay per use basis. This makes the spectrum of options available to scientists, and particularly PSEs users, wide enough to cover any specific need from their research. Another important feature, from which scientists can benefit, is the ability to scale up and down the computing infrastructure according to the application requirements and the budget of users. They can have immediate access to required resources without any capacity planning and they are free to release resources when no longer needed.

Central to Cloud computing is the concept of virtualization, i.e. the capability of a software system of emulating various operating systems. By means of this support, scientists can exploit Clouds by requesting from them machine images, or virtual machines that emulate any operating system on top of several physical machines, which in turn run a host operating system. Usually, Clouds are established using the machines of a datacenter for executing user applications while they are idle.

Interaction with a Cloud environment is performed via Cloud services [4], which define the functional capabilities of a Cloud, i.e. machine image management, access to software/data, security, and so forth. Cloud services are commonly exposed to the outer world via Web Services [8], i.e. software components that can be remotely invoked by any application. By using these services, a user application can allocate machine images, upload input data, execute, and download output (result) data for further analysis. To offer on demand, shared access to their underlying physical resources, Clouds dynamically allocate and deallocate machines images. Besides, and also important, Clouds can co-allocate N machines images on M physical machines, with $N \geq M$, thus concurrent user-wide resource sharing is ensured.

A Cloud gives users the illusion of a single, powerful computer in which complex applications can be run. The software stack of the infrastructure can be fully adapted and configured according to user’s needs. This provides excellent opportunities for scientists and engineers to run applications that demand by nature a huge amount of computational power. Precisely, for parametric studies such as the one presented in this paper or scientific applications [9] in general, Cloud Computing has an intrinsic value.

2.2 The CloudSim Toolkit: Simulation of Cloud Computing environments

CloudSim [6] is an extensible simulation toolkit that enables modeling, simulation and experimentation of Cloud Computing infrastructures and application provisioning environments. CloudSim supports both system and behavior modeling of Cloud system components such as data centers, virtual machines (VMs) and resource provisioning policies. A virtual machine (VM) is a software implementation of a machine (i.e. a computer) that executes programs like a physical machine. The core hardware infrastructure services related to Clouds are modeled by a Datacenter component for handling service requests. A Datacenter is composed by a set of hosts that are responsible for managing VMs during their life cycle. Host is a component that represents a physical computing node in a Cloud, and as such is assigned a pre-configured processing capability, memory, storage, and scheduling policy for allocating processing elements (PEs) to VMs.

CloudSim supports scheduling policies at the host level and at the VM level. At the host level it is possible to specify how much of the overall processing power of each PE in a host will be assigned to each VM. At the VM level, the VMs assign a specific amount of the available processing power to individual task units -called cloudlet by CloudSim- that are hosted within its execution engine. At each level, CloudSim implements the *time-shared* and *space-shared* allocation policies. When employing the space-shared policy only one VM can be running at a given instance of time. This policy takes into account how many PEs will be delegated to each VM. The same happens for provisioning cloudlets within a VM, since each cloudlet demands only one PE. If there are other cloudlets ready to run at the same time, they have to wait in the run queue (because one PE is used exclusively by one cloudlet). Last but not least, with the time-shared policy, the processing power of hosts is concurrently shared by the VMs. Therefore, multiple cloudlets can simultaneously multi-task within the same VM. With this policy, there are no queuing delays associated with cloudlets.

3 Case Study: A PSE for nonlinear solids problems

In order to assess the effectiveness of Cloud Computing environments for executing PSEs, we have processed a real experiment by using different Cloud infrastructures simulated via CloudSim. The case study chosen is the problem proposed in [10], in which a plane strain plate with a central circular hole is studied. The dimensions of the plate are 18 x 10 m, R = 5 m. Material constants considered are E = 2.1 10^5 Mpa; ν = 0:3; σ_y = 240 Mpa; H = 0. A linear Perzyna viscoplastic model with m = 1 and n = ∞ is considered. The large strain elasto/viscoplastic Finite Element code SOGDE [11] is used in this study. A detailed presentation of viscoplastic theory, numerical implementation and examples can be found in the works [12], [13].
We have previously studied parametric problems where a geometry parameter of imperfection was chosen [5]. In this case a material parameter is selected as a parameter. Different viscosity values of η parameter are considered: 1.10^4, 2.10^4, 3.10^4, 4.10^4, 5.10^4, 7.10^4, 1.10^5, 2.10^5, 3.10^5, 4.10^5, 5.10^5, 7.10^5, 1.10^6, 2.10^6, 3.10^6, 4.10^6, 5.10^6, 7.10^6, 1.10^7, 2.10^7, 3.10^7, 4.10^7, 5.10^7, $7.10^7$and 1.10^8 Mpa. Here, a mesh of 1,152 elements and Q1/P0 elements was used. Imposed displacements (at y=18m)

are applied until a final displacement of 2000 mm is reached in 400 equals time steps of 0.05 mm each one. $\delta = 1$ has been set for all the time steps.

4 Experimental Results

This section presents the results obtained from our experimental study, which aims to evaluate the viability of using Cloud Computing to perform PSEs. First, in a single machine we run the PSE of the previous section by varying the elasticity parameter η and measuring the execution time for 25 different experiments (resulting in 25 input files with different input configurations). The PSE were solved using the SOGDE solver. The characteristics of the machine on which the experiments were carried out are shown in Table 1. The machine model is AMD Athlon(tm) 64 X2 3600+, running Ubuntu 11.04 kernel version 2.6.38-8.

The obtained real information (execution times, input/output file sizes) was then used to feed CloudSim. The information regarding processing power was obtained from the benchmarking support of Linux and as such is expressed in bogomips. Bogomips (from bogus and MIPS), is a metric used by Linux operating systems that indicates how fast a computer processor runs. After that, we performed a number of simulations involving executing the PSE on Cloud infrastructures by using CloudSim. The simulations have been carried out by taking into account the bogomips metric. This is, once the execution times have been obtained from the real machine, we calculated for each experiment the number of executed instructions by the following formula: $NI_i = bogomipsCPU * T_i$, where NI_i is the number of million instructions to be executed by or associated to a task i, bogomipsCPU is the processing power of our real machine measured in bogomips and T_i is the time that took to run a task i on the real machine. Here is an example of how to calculate the number of instructions of a task that took 117 seconds to be executed. The machine where the experiment was executed has a processing power of 4,008.64 bogomips. Then, the resulting number of instructions for this experiment was 469,011 MI (Million Instructions). CloudSim was configured as a data center composed of a single machine –or "host" in CloudSim terminology– with the same characteristics as the real machine where the experiments were performed. The characteristics of the configured host are shown in Table 2.

Table 1. Machine used to execute the PSE

Feature	Characteristic
CPU power	4,008.64 bogoMIPS
Number of CPUs	2
RAM memory	2 Gbytes
Storage size	400 Gbytes
Bandwidth	100 Mbps

Table 2. Host characteristics

Host Parameters	Value
Processing Power	4,008
RAM	4,096
Storage	409,600
Bandwidth	100
PE	2

Processing power is expressed in MIPS (Million Instructions Per Second), RAM memory and Storage capacity are in MBytes, bandwidth in Mbps, and finally, PE is the number of processing elements (CPUs/cores) of a host. Once configured, we checked that the execution times obtained by the simulation coincided or were close to real times for each independent task performed on the real machine. The results were successful in the sense that one experiment (i.e. a variation in the value of η) took 117 seconds to be solved in the real machine, while in the simulated machine the elapsed time was 117.02 seconds. Once the execution times have been validated for a single machine on CloudSim, a new simulation scenario was set, which consisted of one datacenter with 10 hosts, each with the same hardware capabilities as the real single machine, and 40 VMs, each with the characteristics specified in Table 3. A summary of this simulation scenario is shown in Table 4.

Table 3. VM characteristics

VM Parameters	Value
MIPS	4,008
RAM	1,024
Image Size	102,400
Bandwidth	25
PE	1
Vmm	Xen

Table 4. CloudSim configuration

Parameter	Value
Number of Hosts	10
Number of VMs	40
Number of Cloudlets	from 25 to 250

With this new scenario, we performed several experiments to evaluate the performance of our PSE in a simulated Cloud Computing environment as we increase the number of tasks to be performed, i.e. 25 * i tasks with i = 1, 2, …, 10. This is, a base subset comprising 25 tasks was obtained by varying the value of η, while the extra tasks were obtained not by further varying this value but cloning the base subset. The reason of this was to stress the various experimental Cloud scenarios.

Each task, called cloudlet by CloudSim, is described by its Length, required PEs, and Input File and Output File sizes. The Length parameter is the number of instructions to be executed by a cloudlet in MI (Million Instructions). PE is the number of processing elements required to perform a task (CPUs). Input File and Output File sizes are expressed in bytes. The values used in the simulation were between 244,127 and 469,011 (Length), 1 (PEs), 93,082 bytes (Input File size) and 2,202,010 bytes (Output File size).

To perform the simulation we have considered, on one hand, that PSE cloudlets have similar processing times. The processing times are similar because both input and output files have the same size. The size of the input files is equal because only one parameter is varied within them. One cloudlet corresponds to execute an instance of a PSE of viscoplastic solid. On the other hand, the goal is to assign tasks to Cloud hosts so that the total completion time, also known as makespan, is minimized. Finally, the order in which cloudlets are processed on a particular host is not relevant, since we assume they are completely independent and do not share data.

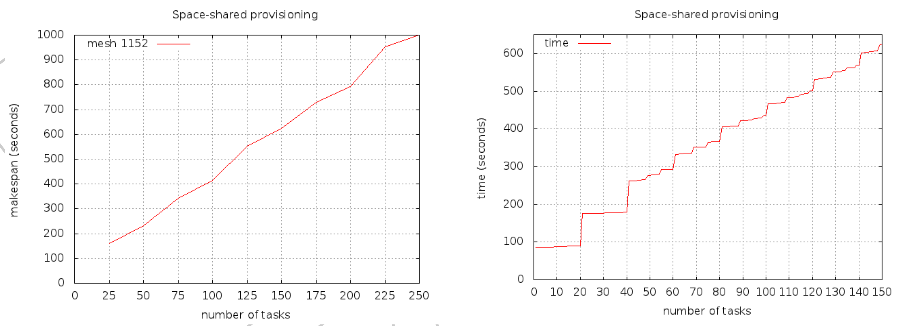
In CloudSim, the amount of available hardware resources to each VM is constrained by the total processing power and system bandwidth available within the associated host. Therefore, scheduling policies must be applied in order to assign the VMs to the host and get a maximum use of resources. On the other hand, cloudlets must also be scheduled with some scheduling policy for a maximum resource performance and minimize the makespan. In the next subsections we report the obtained results when executing the 25 experiments of our PSE using the scheduling policies described in subsection 2.2. In addition, we have considered two types of environments, i.e. homogeneous and heterogeneous, which are explained below.

4.1 Without resource heterogeneity

In this subsection we analyze how each scheduling policy responds when Cloud hosts and VMs follow the specifications described in Table 2 and 3.

4.1.1 Space-shared provisioning

Fig. 1a presents the provisioning scenario where the space-shared policy is applied for both VMs and tasks (cloudlets). Here, the makespan of the whole cloudlets is shown.



(a) Makespan (b) Execution times

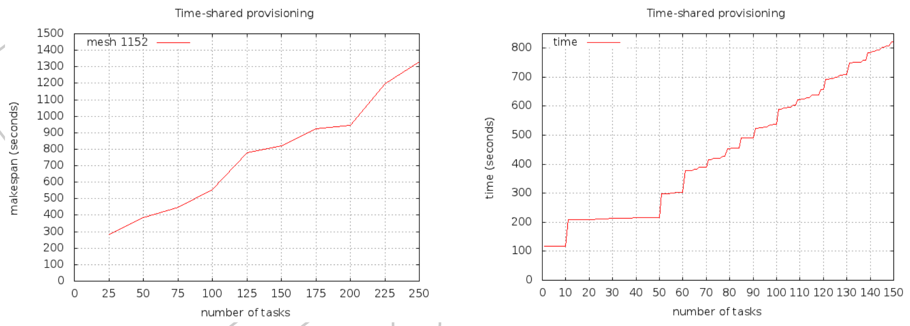
Fig. 1. Space-shared: Results

The makespan has shown a linear growth with respect to the number of cloudlets. After creating VMs, cloudlets were incrementally sent to VMs in groups of 25 to measure the makespan as we increase the workload on the VMs. The makespan rose from 160.25 to 1,000.86 seconds when the number of cloudlets went from 25 to 250. As each VM requires one PE for processing (see Table 3), with the space-shared policy only two VMs can actually run in a host at a given instant of time, because each host has two PEs as shown in Table 2. Therefore, given a scenario consisting of a total of 10 hosts and 40 VMs, at a given instant of time may be assigned 20 VMs to the hosts, i.e. one VM by each PE, and the rest of the VMs can be assigned once the former set complete their execution. As the number of PSEs and hence cloudlets in regard to the available amount of resources increases, the estimated start time of each cloudlet depends on the position of the cloudlet in the execution queue, since each PE is used exclusively by one cloudlet under the space-shared policy. Remaining cloudlets are queued when there are not free processing elements to use for execution.

Fig. 1b presents the progress of execution times when we sent to execute a group of 150 cloudlets, i.e. a group of 150 PSEs as described in the previous section. Since, under this policy, each cloudlet had its own dedicated PE, the queue size (cloudlets waiting to be run) did not affect execution time of individual cloudlets. As shown, the execution times were increasing linearly approximately every 20 tasks. This is because as mentioned above, only 20 VMs were created with the space-shared policy, so the cloudlets are sent to the VMs to run in groups of 20 until they finish their execution. When the first submitted group of cloudlets finishes their execution, 20 more are sent, and so on until all cloudlets are executed.

4.1.2 Time-shared provisioning

In this scenario a time-shared allocation is applied. Fig. 2a shows the makespan as the number of cloudlets increases from 25 to 250. Here, the processing power within a host is concurrently shared by its associated VMs, and the PEs of each VM are simultaneously divided among its cloudlets. As a consequence, there are no queuing delays associated with cloudlets. CloudSim assumes that all the computing power of PEs is available for VMs and cloudlets, and it is divided equally among them. In this scenario, the makespan rose from 280.94 to 1,328.13 seconds when the number of cloudlets was increased from 25 to 250.



(a) Makespan (b) Execution times

Fig. 2. Time-shared provisioning: Results

Fig. 2b illustrates the progress of execution times when we sent to execute a group of 150 cloudlets, or a group of 150 PSEs. Since using the time-shared policy in the hosts the processing power available is concurrently shared by VMs, here 40 VMs have been created in the 10 available hosts. Here, the execution times were increasing gradually over the first 50 cloudlets. The remaining 100 tasks took considerably longer than the 50 first tasks. This latter effect occurs because the VMs available for processing within hosts begun to be switched between their PEs, which takes time.

4.2 With resource heterogeneity

In this subsection we analyze how each scheduling policy responds when using a Cloud with heterogeneous hosts. To analyze the performance of the scheduling algorithms, one characteristic that is of importance in real world scenarios is how the algorithms perform in the presence of resource heterogeneity. In this analysis, we have considered hosts with a random number of PEs between 1 and 6, while the other specifications are the same shown in Tables 2 and 3. Until now, each VM had only one PE. Next, we discuss the same scenarios of the previous section, and perform a comparison of task assignment with respect to homogeneous and heterogeneous infrastructures.

4.2.1 Space-shared provisioning

Fig. 3 presents a scenario where the space-shared policy is applied. After creating VMs with a random number of PEs, cloudlets were incrementally sent to VMs in groups of 25 to measure the makespan as the workload on the VMs increased. The number of cloudlets to be performed ranges from 25 to 250 as in the previous subsection. In the Figure, the allocation of cloudlets to heterogeneous resources is illustrated by the curve in blue. The red curve shows the same scenario that was discussed in subsection 4.1.1 for the case of homogeneous resources.

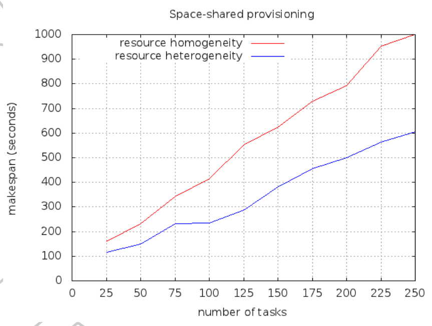


Fig. 3. Space-shared provisioning using resource heterogeneity: Results

Due to the fact that in this scenario the entire Cloud had more number of PEs available to run the experiments (between 1 and 6 per resource), the runtimes were reduced significantly with respect to the homogeneous scenario. The makespan of the first group of 25 cloudlets was very close to the makespan of the homogeneous scenario. This makespan was 160.25 seconds for the homogeneous scenario and 117.12 seconds when using heterogeneous resources. The makespan is close because in the worst case (homogeneous scenario) the number of PEs available to execute the cloudlets is nearby to the number of executed cloudlets (20 VMs to execute 25 cloudlets). Then, each cloudlet is executed in one PEs until the former finishes. For the following groups of cloudlets –between 50 and 250- the makespan was always lower when using heterogeneous resources. The makespan was 117.12, 150.24, 231.38, 235.43, 288.54, 380.74, 454.94, 499, 563.07, and 604.22 seconds.

4.2.2 Time-shared provisioning

In this subsection we present the results for a heterogeneous scenario where the time-shared policy is applied. Fig. 4 illustrates the progress curve in blue of execution times as the number of cloudlets increase from 25 to 250.

In this heterogeneous scenario, the makespan was 146.37 seconds and 677.64 seconds when the number of cloudlets was equal to 25 and 250, respectively. In the figure, we have performed the comparison with the scenario composed of homogeneous resources (curve in red). Overall, we obtained that, with this heterogeneous scenario, the makespan for the entire set of cloudlets was significantly reduced.

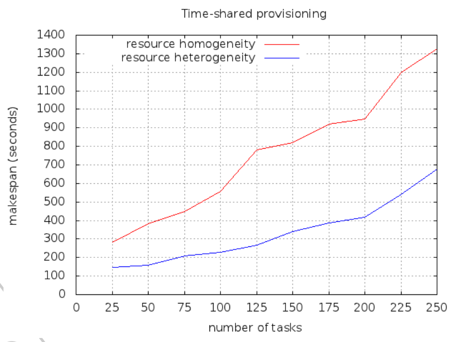


Fig. 4. Time-shared provisioning using resource heterogeneity. Results

4.3 Summary and discussion

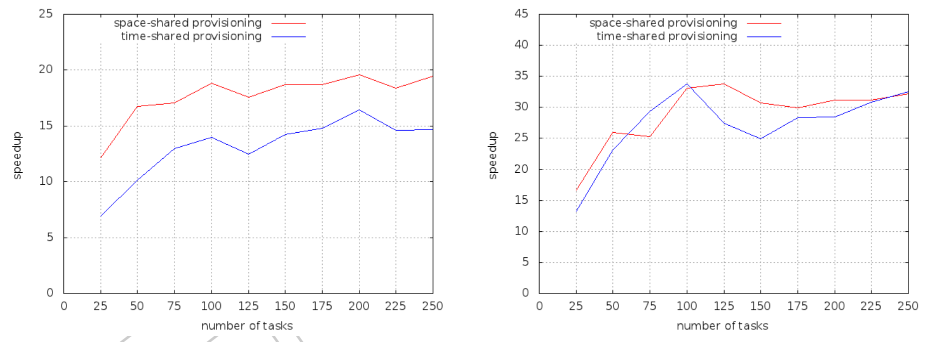
In the previous sections we have reported the results obtained from executing our PSE on a simulated Cloud under two different scenarios and also taking into account whether the resources are homogeneous or heterogeneous. When the space-shared policy is used to assign VMs to hosts, the scheduler assigns as many VMs as PEs have available on the hosts (20 VMs in the proposed scenario). Instead, when time-shared policy is used, the PEs share their time slots among all VMs to be created (40 VMs). On the other hand, the cloudlets are sent to be executed something similar happens. When using the space-shared policy each cloudlet is assigned to a VM until it completes its execution. With the time-sharing policy processing power must be shared among several cloudlets, generating a lot of exchanges for completing their execution, which makes each cloudlet to take longer to finish.

The better performance obtained by the space-shared policy is mainly because each VM can allocate and get all the processing power that needs to execute assigned cloudlets from the host where the VM executes. Instead, with the time-shared policy, each VM receives a time slice on each processing element, and then distributes the slices among the PSEs to be executed. Due to the fact that the VMs have less processing power (time slices) the experiments took longer to complete. As a result, the space-shared policy was more appropriate to this type of PSEs.

Then, we performed the analysis of the same scenarios, but in a heterogeneous environment. The behavior of the different combinations of scheduling techniques was exactly the same as the case of homogeneous resources. Although the scheduling criteria to assign VMs to hosts and cloudlets to VMs were the same, the makespan was reduced for all scenarios. This improvement was because in the heterogeneous environment as many PEs as needed to run experiments were available.

Finally, we performed a speedup analysis to measure the performance of each technique (space-shared and time-shared) to execute the PSEs on the Cloud with respect to the sequential execution on a single machine (see Fig. 5).

The speedup is calculated as $S_p = T_1 / T_p$, where p is the number of processing elements, T_1 is the completion time of the sequential execution in a single machine, and T_p is the completion time of the parallel execution with p processing elements.



(a) Without resource heterogeneity (b) With resource heterogeneity

Fig. 5. Speedup achieved by space-shared and time-shared policies: Results

We conclude that a scenario in which the space-sharing policy is used for the VMs allocation to hosts enables a better speedup than the time-sharing policy in both scenarios (homogeneous and heterogeneous). While for the experiments we have conducted in this paper a space-share policy for the allocation of VMs to hosts yields better results, due to the fact that the employed cloudlets are sequential –i.e. they have no inner parallelism to exploit–, a time-share policy to assign the VMs to hosts would be more appropriate for other types of applications (not batch or sequential) and also could be good to improve not only the makespan but also the perceptible response time to the user, since incoming tasks could be periodically scheduled and then executed in small groups, thus giving sign of progress.

5 Conclusions

Cloud Computing is a new paradigm that offers the means for building the next generation parallel computing infrastructures. Although the use of Clouds finds its roots in IT environments, the idea is gradually entering scientific and academic ones. Even when Cloud Computing is popular, little research has been done with respect to evaluating the benefits of the paradigm for scheduling and executing resource intensive scientific applications. Through a real case study and simulations, we have reported on the speedups obtained when running PSEs on Clouds. Results are quite encouraging and support the idea of using Clouds in the academia.

We are extending this work in several directions. First, we are conducting studies with other kind of PSEs, such as tension tests in metals [13], to further support our claims. Second, one of the key points to achieve good performance when using Clouds concerns task scheduling. In particular, there is an important amount of work in this respect in the area of Cloud Computing and distributed systems in general that aim at building schedulers by borrowing notions from Swarm Intelligence (SI), a branch of Artificial Intelligence that comprise models that resemble the collective behavior of decentralized, self-organized systems like ants, bees or birds. Moreover, a recent survey of our own [14] shows that there is little work regarding SI-based schedulers for Cloud Computing. Therefore, we aim at designing a new SI-based scheduler that is capable of efficiently run PSEs. We are also planning to embed the resulting scheduler into CloudSim in order to provide empirical evidence of its effectiveness. Eventually, we could implement the scheduler on top of a real (not simulated) Cloud platform, such as Eucalyptus (http://www.eucalyptus.com).

References

1. Youn C. and Kaiser T. Management of a parameter sweep for scientific applications on cluster environments. Concurrency and Computation: Practice and Experience, vol. 22, pp. 2381–2400, (2010)
2. Thain D., Tannenbaum T., and Livny M. Distributed computing in practice: The Condor experience. Concurrency and Computation: Practice and Experience, vol. 17, pp. 323–356, (2005)
3. Foster I. and Kesselman C. The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Inc., San Francisco, CA, USA, (2003)
4. Dikaiakos M.D., Katsaros D., Mehra P., Pallis G. and Vakali, A. "Cloud Computing: Distributed Internet Computing for IT and Scientific Research," IEEE Internet Computing, vol.13, No.5, pp.10-13, (September 2009)
5. Careglio C., Monge D., Pacini E., Mateos C., Mirasso A., and García Garino C. Sensibilidad de resultados del ensayo de tracción simple frente a diferentes tamaños y tipos de imperfecciones. Mecánica Computacional, vol. XXIX, pp. 4181–4197, (2010)
6. Calheiros R.N., Ranjan R., Beloglazov A., De Rose C., and Buyya R. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software - Practice and Experience, vol. 41, pp. 23–50, (2011)
7. Gropp W., Lusk E., and Skjellum A. Using MPI: Portable Parallel Programming with the Message Passing Interface. MIT Press, (1994)
8. Erickson J. and Siau K. Web Service, Service-Oriented Computing, and Service-Oriented Architecture: Separating hype from reality. Journal of Database Management, vol. 19, pp. 42–54, (2008)
9. Wang L., Tao J., Kunze M., Castellanos A.C., Kramer D., and Karl W. Scientific cloud computing: Early definition and experience. In: 10th IEEE International Conference on High Performance Computing and Communications, pp. 825–830, (2008)
10. Alfano G., Angelis F.D., and Rosati L. General Solution procedures in elasto-viscoplasticity. Computer Methods in Applied Mechanics and Engineering, vol. 190, pp. 5123–5147, (2001)
11. García Garino C. and Oliver J. Un modelo constitutivo para el análisis de sólidos elastoplásticos sometidos a grandes deformaciones: Parte i formulación teórica y aplicación a metales. Revista internacional de métodos numéricos para cálculo y diseño en ingeniería, vol.11, pp. 105–122, (1995)
12. Ponthot J., García Garino C., and Mirasso A. Large strain viscoplastic constitutive model. Theory and numerical scheme. Mecánica Computacional, vol. XXIV pp.441–454, (2005)
13. García Garino C., Gabaldón F., and Goicolea J.M. Finite element simulation of the simple tension test in metals. Finite Elements in Analysis and Design, vol. 42, pp. 1187–1197, (2006)
14. Pacini E., Mateos C., and García Garino C. Planificadores basados en inteligencia colectiva para experimentos de simulación numérica en entornos distribuidos. In: Sexta Edición del Encuentro de Investigadores y Docentes de Ingeniería ENIDI'11. (2011)