Towards a Cost Model for Scheduling Scientific Workflows Activities in Cloud Environments

Vitor Viana, Daniel de Oliveira, and Marta Mattoso
Federal University of Rio de Janeiro, Brazil
{vitorgv, danielc, marta}@cos.ufrj.br

Abstract

Cloud computing has emerged as a new computing model that enables scientists to benefit from several distributed resources such as hardware and software. Clouds are an opportunity for scientists that need high performance computing infrastructure to execute their scientific experiments. Most of the experiments modeled as scientific workflows manage the execution of several activities and produce a large amount of data. Since scientists work with large amounts of data, parallel techniques are often a key factor on the experimentation process. However, parallelizing a scientific workflow in the cloud environment is far from trivial. One of the complex tasks is to define the configuration of the environment, i.e. the number and types of virtual machines and to design the parallel execution strategy. Due to the number of options for configuring an environment it is a hard task to be done manually and it may produce negative impact on performance. This paper proposes a lightweight cost model that is based on concepts of quality of service (QoS) in cloud environments to help determining an adequate configuration of the environment according to restrictions imposed by scientists.

Keywords: Scientific Workflows, Cloud Computing

1. Introduction

Over the last years, in the scientific experimentation process [1,2], due to the effective need of analyzing an increasing amount of information and complex scientific data, scientists use several programs and toolkits in their experiments. These experiments make use of several computing resources and are commonly modeled as scientific workflows [1]. In scientific workflows, a specific set of programs is organized in a coherent flow of activities in which the outcome of an activity is transferred to the next activity of the flow as an input to be consumed. Scientific workflows are normally handled by mechanisms, called Scientific Workflow Management Systems (SWfMS) [3].

The activities of a scientific workflow can also be re-executed many times as needed, varying the input data of the workflow or its parameters in order to analyze the data products generated by the workflow execution. This situation is commonly named parameter sweep [4] and it occurs when the same workflow is exhaustively executed using different input data files and/or different configurations until the exploration finishes. In addition, in most scenarios, these workflow variations are time-consuming and cannot be processed in a feasible time by a single computer or a small cluster. Due to this characteristic, scientific workflows require some degree of parallelism to reduce total execution time.

Although workflow parallelism is a key issue, it is not the only impact factor on total execution time. The workflow execution may be unfeasible if scientists do not run their experiments in high performance (HPC) environments associated to parallel techniques such as parameter sweep. Clouds [5] have emerged as an alternative HPC environment where Web-based services allow for different kinds of users to obtain a large variety of resources. Clouds have demonstrated applicability to a wide-range of problems in several domains, including scientific ones [6]. An important advantage provided by clouds is that scientists are not required to assemble expensive computational infra-structure to execute their experiments or even configure many pieces of software. Images can be pre-configured to be instantiated on demand. An average scientist is able to run experiments by allocating the necessary resources in a cloud [7]. In addition, cloud resources are available via Web services, and can be dynamically re-configured to fulfill a variable demand (*i.e.* scalability). The use of these services is typically operated by a pay-per-use model [5].

Although clouds represent a step forward, the management of a parallel execution of scientific workflows in clouds is an open issue [8]. In addition, cloud-based configuration services are not available in the existing SWfMS. A specific infrastructure for running parallel scientific workflows in clouds is necessary. One of the recently proposed solutions to provide this infrastructure is named SciCumulus [8].

SciCumulus is a middleware designed to distribute, manage and monitor parallel execution of scientific workflows activities (or even entire scientific workflows) using a SWfMS in the cloud, such as Amazon EC2 [9]. SciCumulus orchestrates the execution of workflow activity in a distributed set of Virtual Machines (VM). SciCumulus configures the cloud environment for the workflow execution, thus creating VMs to execute activities, setting up parameters and staging data in and out.

However, in its current version, SciCumulus creates the VMs according to the scientist definition. This way, scientists have to inform the number of necessary VMs and its configuration. However, there may be several combinations to choose when configuring an environment and it can be tedious and error-prone according to Juve and Deelman [8].

This way, given a set of types of VMs, the decision of which one will perform an activity is not a simple task to accomplish. This paper aims at developing a lightweight cost model to help determining an adequate environment configuration before scheduling workflow activities in cloud environments. The main idea is to determine an effective configuration of the environment according to some restrictions informed by scientists (total workflow execution time and monetary cost). This configuration is produced based on information collected from the cloud environments and the types of VMs available.

By using this lightweight model and some important information provided by scientists we are able to develop a new component to couple to SciCumulus scheduler – the optimizer. With this component we can determine the best possible configuration of the environment that can be used in the workflow scheduling. By setting up the environment with the best possible configuration, the responsibility of the scheduler is focused on selecting the specific VM that will perform a certain activity in a given time. The cost model presented in this paper aims at being QoS-oriented, meaning that it is targeted on improving the satisfaction of scientists, maintaining a good Quality of Service (QoS). Restrictions are commonly specified by scientists and used in the approach proposed in this paper to provide some differentiation between the VMs to be used. The approach proposed in this paper is based on two criteria (total execution time and monetary cost), with the priority ordering chosen by the scientist, initially optimizing the first criterion and based on the optimization tries to optimize the next criterion.

This paper is organized as follows. Section 2 discusses related work and background. Section 3 briefly describes the SciCumulus architecture. Section 4 introduces the proposed cost model for supporting dynamic resource scheduling in the cloud. Section 5 presents some experimental results. Finally, Section 6 concludes and presents future work.

2. Related Work and Background

The term “Quality of Service” (QoS) [10] refers to all non-functional features of a service (in our case the parallel workflow execution in a cloud) that can be used by a client (or a scientist in the context of this paper) to evaluate the quality of service. Important features related to QoS are: (i) Performance; (ii) Reliability; (iii) Capacity and; (iv) Cost. However, in this paper we focus only on Performance and Cost. Initially, the term "Quality of Service" refers to some performance aspects of a network, not including any control or interference by the end user. However, nowadays the specification of quality criteria is widely used and currently covers various domains of computer science such as distributed and parallel computing, and in the context of this paper allowing the definition of quality criteria by scientists, to specify the level of quality you want in running scientific workflows.

Following we provide a brief description of some existing work in the literature that relate in some way to the issue that we are dealing in this paper. Furtado *et al.* [11], presents a comparison between scheduling in distributed databases. The authors analyze the centralized and hierarchical architecture according to the throughput and QoS constraints for distributed services, similarly to this paper. About the QoS constraints, the authors state that the time constraints (*e.g.* deadlines) are among the mostly used QoS criteria. Although the authors use QoS constraints in a cost-model, their objective is different from the ones presented in this paper because they do not intend to configure the environment. Furtado *et al.* intend to schedule distributed queries in a pre-configured environment (distributed database).

Dias *et al.* [12] proposes Heracles, an approach that uses Peer-to-Peer (P2P) techniques to manage distributed scientific workflows on huge clusters based on QoS criteria such as deadlines. However, different from the approach proposed in this paper, Heracles is focused on clusters and does not address cloud configuration issues.

According to Juve and Deelman [8] Amazon cloud provides several good alternatives to HPC systems for workflow computation such as communication and storage services, yet they discuss on the virtualization overhead as one of challenges to overcome. We did not find any proposal to address the issue of cloud configuration for scientific workflows.

3. SciCumulus Architecture

SciCumulus is a cloud middleware designed and implemented to manage the parallel execution of scientific workflow activities (or even entire scientific workflows) from a SWfMS into a cloud environment, such as Amazon EC2. SciCumulus architecture is composed by three layers that are also distributed: (i) Desktop - dispatches workflow activities to be executed in the cloud environment using a local SWfMS such as VisTrails [13], (ii) Distribution - manages the execution of activities in cloud environments, and (iii) Execution - executes programs from workflows. The conceptual architecture was proposed in details by Oliveira *et al.* [14] in a previous paper.

The Desktop layer is responsible for starting parallel execution of workflow activities in the cloud. The components of the Desktop layer are deployed in an existing Scientific Workflow Management Systems (SWfMS) such as VisTrails [13]. The local components installed in the local SWfMS starts the parallel execution in the cloud. There are three main components: Uploader, Dispatcher, and Downloader. The Uploader moves data to the cloud while Downloader gathers the final results from cloud VMs. The Dispatcher launches the execution process of cloud activities [14] in the cloud and communicates directly to the execution broker in the distribution layer.

The Distribution layer manages the execution of parallel activities in cloud environments by creating cloud activities that contain the program to be executed, its parallelism strategy, parameters values and input data to be consumed. The layer has six components: Execution Broker, Parameter Sweeper, Encapsulator, Scheduler, Data Summarizer and Distribution Controller. The Execution Broker makes the connection between the desktop layer and the distribution layer. The Parameter Sweeper handles the combinations of parameters received by the desktop layer for a specific workflow activity that is being parallelized. The Encapsulator generates all cloud activities to be transferred to the virtualized instances. The Scheduler defines which VMs receive a specific cloud activity to execute. The Distribution Controller transfers cloud activities to the available virtual machines and monitors its execution. The Data Summarizer is combines the final results to organize the final results to be transferred to the client layer (SWfMS).

Finally, the Execution layer is responsible for invoking executable codes in many VMs available for use. It has three main components: instance controller, configurator and executor. The Instance Controller makes the connection between the distribution layer and the execution layer. It is also responsible for controlling the execution flow in the instance when the cloud activity is associated to two or more applications to be executed. The Configurator sets up the entire environment. The Configurator unpacks the cloud activity, creates replicas of the program to a specific location; create workspaces to store parameter files, if needed, and stores data to be consumed. Finally, the Executor invokes the specific application and stores provenance data to a repository also stored in the cloud.

This way, SciCumulus provides a computational infrastructure to support workflow parallelism with provenance gathering of the cloud environment. SciCumulus architecture is simple and may be deployed and linked to any existing SWfMS, diminishing effort from scientists.

Although SciCumulus has three layers with several complex components, the approach proposed in this paper is in the scope of the distribution layer. The proposed cost model and the component that determines the best cloud configuration based on this cost model can be coupled to SciCumulus scheduler.

4. Dynamic Selection of Resources in Cloud Environments

With the evolution of cloud computing, several computing resources are already available for use. For example, Amazon EC2 [9] provides more than 10 different types of VMs. Each one of them has a different memory capacity and CPU power. Choosing the type of VM to be used in the parallel execution of a scientific workflow is not a trivial task and can impact directly in the performance of the experiment. In addition, given this wide range of available resources, we can commonly find services offered by different cloud providers such as Amazon EC2[1], IBM[2] and GoGrid[3], but with equivalent functionality. Moreover, in the same cloud environment, we can have multiple VM instantiated.

This way, it is necessary to have cloud services able to determine which of these computing resources to instantiate and use by following a specific criterion such as total workflow execution time or overall monetary cost. To make this type of choice possible, functionally equivalent resources should be searched, cataloged and a cost model has to be applied in order to

[1] aws.amazon.com/ec2/
[2] www.ibm.com/ibm/cloud/
[3] www.gogrid.com/

determine the best possible environment configuration to use.

The proposed model aims at presenting a two criteria cost model with QoS support for clouds, enabling better resources usage and compliance with the requirements of the workflow (informed by scientists). Before starting the execution of a distributed workflow the scientist has to inform the restrictions for the execution as a simplified Service Level Agreement (SLA). This way, scientists describe important (and commonly found) requirements for the experiment execution (total execution time and execution cost). These criteria are based on those described in the papers presented in the related work presented in section 2.

Thus, it would be under the responsibility of an optimizer component (coupled to SciCumulus scheduler) to determine which cloud resource should be instantiated in a given time. In other words, this optimization component would be in charge of optimizing the dynamic selection of VMs before scheduling cloud activities. For an optimizer be able to make the selection of an appropriate form, two requirements must be met. First, cost parameters applicable to scalable and distributed resources and its operations should be established; and second, a cost model that takes into account these parameters should be developed so that resources can be chosen based on these parameters.

The next sub-sections present the parameters considered in the development of the cost model and the cost model itself. This cost model is a basis of the optimizer in order to simulate and achieve the best possible configuration of the environment according to the number and type of VMs.

4.1 Formalization of the Parameters

In this sub-section, we present the parameters related to the cloud environment and used to create a lightweight cost model for determining the best possible configuration of the environment. Let us consider $VM = \{VM_0, VM_1, VM_2, \ldots, VM_{n-1}\}$ as the set of n available types of VMs identified by an agent Ag. For each machine VM_i, also consider the set $P_i = \{P_0, P_1, P_2, \ldots, P_{m-1}\}$, set of m programs installed and configured by each machine VM_i.

This way, we are able to define $C_{Ti,j}$ as the total monetary cost of running a particular program P_j in a virtual machine VM_i in time $T_{i,j}$. In a simplified way, we can define $C_{Ti,j}$ as the sum of the average cost of initializing the virtual machine VM_i, the average cost of executing the program P_j in VM_i for a specific time $T_{i,j}$ and the average cost of data stage in and out involved in executing the program P_j in VM_i. The $C_{Ti,j}$ is defined as the longest initialization time of the VMs (since all VMs initialize in parallel) plus the time required for a particular program to run in a VM plus data transfer. Following in this sub-section we explain each one of the considered parameters and its formalization:

(i) Availability: The availability of a VM, denoted by $D(VM_i)$ in the proposed cost model, corresponds to the percentage of time that a type of VM is available for use, considering a time interval *T*. It is a fundamental criterion; however, the responsible for guaranteeing the availability is the service provider. For example, Amazon EC2 guarantees an availability of 99%.

(ii) Reliability: A VM may be available but may not be able to address a specific request by several problems in a network, such as network congestion or problems in implementation. We define reliability of a VM as $R(VM_i)$ as the ratio between the number of requests serviced by this VM and the number of requests made.

(iii) Initialization Cost: Cost for a given VM to initialize. The initialization cost is estimated by analyzing the historical initialization times of each type of VM and using the VM configuration, *e.g.*, its capacity, the operating system, and so on.

(iv) Transfer Cost: Cost needed to stage in and stage out a certain amount of data from the client machine to the provider.

(v) Storage Cost: Cost based on space needed for storage of information in the cloud.

(vi) Runtime Cost: Cost of execution of a program $P_{,j}$ of a virtual machine VM_i includes the cost / time spent since the beginning of the program execution until its end, ignoring the costs of startup and transfer involved.

4.2 The Proposed Cost Model

Cost models have been used successfully to estimate the costs involved in diverse areas of computing, as in database systems [10], among others. A cost model consists of a set of formulas that estimates the costs involved in a particular computing environment [15], in our case the cloud environment. It can be used for a quantitative simulation of each possible configuration of the environment, thereby allowing that the best possible choice that can be identified objectively, avoiding the actual

implementation of all possible combinations present in the model, which can be costly.

A cost model is usually implemented in a system that performs several simulations. This system is usually called optimizer. In the context of this paper, the optimizer initially receives the activities of the workflow to execute and the combination of data and parameters to be explored. It identifies the set of cloud activities [14,17] produced from the execution of these explorations of parameters and input data. It separates these cloud activities into several subsets to be executed in several VMs. After identifying each cloud activity involved in the execution, the optimizer uses the cost model to simulate several executions and to decide whether to choose a type and number of VM based on one of these two criteria. The approach proposed in this paper is based on a two criteria (performance – overall execution time and monetary cost), with the priority ordering chosen by the scientist, initially optimizing the first criterion and based on the optimization tries to optimize the next criterion: (i) Execution Time: In this case, the choice of VMs will follow the combination of VMs that produce the smallest execution time. However, we also try to optimize the monetary cost, since it cannot exceed the limit previously set by scientists. (ii) Monetary Cost: Here, the situation is the opposite one; the choice of VMs to instantiate to perform a set of cloud activities is prioritized by the monetary cost involved. However, we also try to optimize the overall execution time of the cloud activities that shall not exceed the predetermined limit of total execution time.

The performance cost involved on using a cost model for an optimizer may not be high, because although negligible in most of the published literature, it can have an important role in distributed environments where high communication costs may be involved in obtaining information about the environment. Therefore, we need a model that takes into account the characteristics of VMs, the cloud environment and their programs, and that is able to be simulated several times under different conditions without impacting on the performance of workflow that is being executed by SciCumulus.

Given the configuration of the machines offered by the cloud provider, we developed a cost model that aims at determining which configuration and combinations of VMs to use to execute a particular workflow following some restrictions and taking into account the parameters discussed in the previous sub-section. Thus, we propose a dynamic model to estimate the costs involved in the execution of a workflow in a cloud environment and based on this estimative determine the best environment configuration to be used by an orchestrator such as SciCumulus. Following we detail each one of the parameters involved in the cost model.

4.2.1 Total Execution Cost

The total cost of running a program P_j of a virtual machine VM_i, denoted by C_T is defined in Equation 1,

$$C_T = C_I + C_E + C_R \quad \text{(Eq. 1)}$$

where C_I is the cost of initializing the virtual machines, C_E is the cost of executing the programs P^j in the virtual machines and C_R is the transfer cost involved in executing the programs P^j in the virtual machines.

We can extend Equation 1 so that it encompasses also the quality criteria of "availability" and "Reliability" as defined in the previous section. Thus, we are ensuring that the cost model is not static, because, after repeated executions of scientific workflows, the parameters "Availability" and "Reliability" can vary, making instances constantly unavailable or unreliable. Thus, Equation 2 defines the total cost of implementation.

$$C_T = \frac{C_I + C_E + C_R}{D(VM_i) \times R(VM_i)} \quad \text{(Eq. 2)}$$

The values of $D(VM_i)$ and $R(VM_i)$ are integer numbers varying between 0 and 1, to divide the total execution time for these quality criteria, we will be increasing the execution time of VMs less reliable or constantly unavailable.

Ideally, each provider should provide this information to client applications that wish to use their resources. Thus, we can define initialization cost as in Equation 3.

$$C_I = \sum_{i=0}^{n-1} I(VM_i) \quad \text{(Eq. 3)}$$

The execution cost is the sum of the costs of executing the programs P_j of a particular virtual machine VM_j $(t_{i,j}^e)$.

$$C_E = \sum_{j=0}^{m-1} \sum_{i=0}^{n-1} t_{i,j}^e \quad \text{(Eq. 4)}$$

The transfer cost is calculated based on the volume of data that we stage in and stage out to and from the cloud environment and the price charged by the service provider. We define it as Equation5,

$$C_R = C_U + C_D + C_S \quad \text{(Eq. 5)}$$

where C_U is the cost of uploading, the C_D is the cost of downloading and C_S is the cost of storing data in the provider.

4.2.2 Total Execution Time

The total execution time of running all programs P_j in VM_i, denoted by T_T is defined in Equation 6,

$$T_T = T_I + T_E + T_R \quad \text{(Eq. 6)}$$

where T_I is the higher startup time of the virtual machines, T_E is the cost of implementing the programs P^j in the virtual machines and T_R is the transfer time involved in executing the programs P_j in the virtual machines.

The initialization time should be calculated beforehand by analyzing initialization times t_i^I of each VM_i and taking the highest one because all VMs instantiate in parallel, as defined by Equation 7.

$$T_I = \max_{0 \le x \le n-1} t_i^I \quad \text{(Eq. 7)}$$

The runtime is the sum of execution times of programs P_j in a given virtual machine VM_i varying with the number of parameters of the activity. Equation 8 defines execution time.

$$T_I = \sum_{j=0}^{m-1} \sum_{i=0}^{n-1} t_{i,j}^e \quad \text{(Eq. 8)}$$

The transfer time is calculated based on the volume of data that we capture and send the server and available bandwidth. We define it as follows:

$$T_I = \sum_{i=0}^{n-1} t_i^D + \sum_{i=0}^{n-1} t_i^U \quad \text{(Eq. 9)}$$

Where t_i^U is the time to upload and t_i^D is the time to download from and to a specific VM_i.

4.3 Using the Model in SciCumulus

In order to evaluate the model in a real scientific workflow cloud environment we introduce an optimization component in SciCumulus architecture. The optimization component is a simulator implemented in Java that simulates several combinations of types of VMs, transferring data and so on to produce the best possible environment configuration for SciCumulus.

The optimizer simulates the execution of SciCumulus components with several environment configurations according the criteria informed by scientists. To simulate the execution and the environment, the optimizer uses several information captured from the cloud environment by an autonomous agent. This agent is independent from SciCumulus and crawlers the several cloud providers capturing information about types of VMs, pricing, VM capacity, and transferring rates. All of this information is stored in a specific DB schema of SciCumulus and can be considered important provenance data [9,17]. The information about the experiment and environment definition is fundamental to scientific experimentation process. This data is called Provenance Data [18].

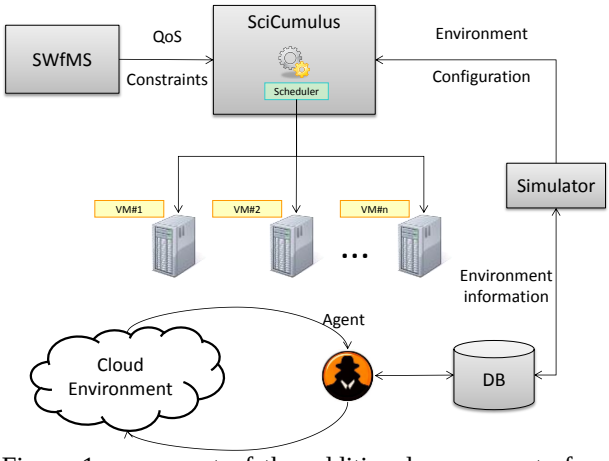

Figure 1 an excerpt of the additional components for the SciCumulus architecture

5. Experimental Results

We performed several tests to validate our cost model in defining the environment for SciCumulus. In this Section we present the most relevant experimental simulated results achieved by the optimizer component and the real results to evaluate if the environment setup proposed is indeed the best possible choice for the experiment being executed. For the simulation, we used an extended version of CloudSim [18]. Scientists have to inform the primary and secondary criteria to optimize (performance or cost in order). According to the QoS data provided by scientists the cost model was then calibrated taking into account the cloud activities to be executed, the size of input data and the size of output data.

We used the X-ray crystallography scientific workflow [19] results to calibrate the cost model with real information about program execution time. This workflow is a very representative one and due to this characteristic, it was chosen as use case for the next Provenance Challenge[4]. X-ray crystallography experiments concern on determining the three-dimensional structure of the molecules by analyzing

[4] twiki.ipaw.info/bin/view/Challenge/FourthProvenanceChallenge

the diffractions patterns obtained from X-ray scattering experiments. The X-ray crystallography scientific workflow is composed by several programs and toolkits such as CCP4. The information about the cloud environment was captured from Amazon EC2 environment using Amazon API tools encapsulated in the agent. The workflow used to test the cost model and the optimizer processes 2,000 images in groups of 3 and it is composed by 667 cloud activities to be distributed among the VMs in the cloud environment. This workflow executes in approximately 45,876 seconds in a single machine.

The optimizer first simulates the instantiation of one VM for each cloud activity. Intuitively we can visualize that this is not an appropriate policy to follow. This way, the optimizer starts to reduce the number of VMs and analyzes if the total execution time reduces proportionally until we reach the restriction values for performance or cost (depending on the priority choice of the scientist). This process is executed for each one of VM types analyzing performance and monetary cost. The restrictions imposed in this simulation were total execution time of 1 hour and 10 dollars of cost maximum.

Since we are executing embarrassingly parallel experiments, the performance varied the type of VM was not significant when the number of VMs involved in the execution increases. This behavior can be explained since embarrassingly parallel experiments are the most scalable patterns for clouds [20,21]. In our simulation, the execution time restriction (1 hour = 3,600 seconds) was reached using from 16 up to 256 virtual cores as presented in Figure 2.

Although the performance is quite similar using different configurations, the monetary cost was significantly different. For example, the price paid varies from US$ 2,448 (16 cores) to US$ 39,168 (256 cores). In this simulation we defined as the primary priority the monetary cost instead of execution time. This way, the optimizer fixed the number of VMs in 16 (because the restriction of total execution time was fully satisfied) and started to vary the type of VM in order to reduce cost. As result, the high CPU extra-large instance was considered the suitable one from the monetary point of view.

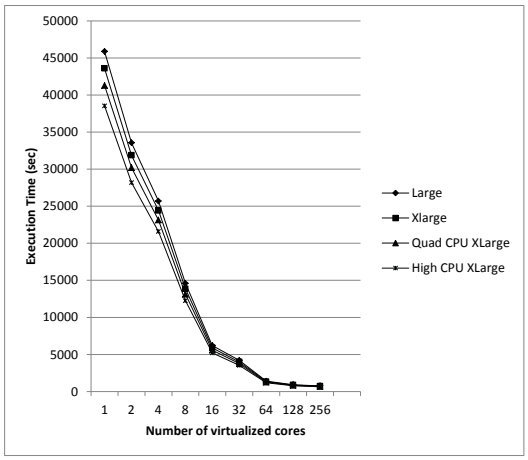

Figure 2 Simulated results for the execution of the workflow

Since we have chosen the high CPU extra-large instances in Amazon EC2 to use, we simulated the monetary cost of running the experiment using from 2 cores up to 16 cores to evaluate the cost model and the optimizer. All simulations produced a monetary prediction lower than the real cost, the results were only up to 8% lower. It indicates that a fine tuning in the optimizer is indeed needed. However, the behavior of the simulator follows the real execution.

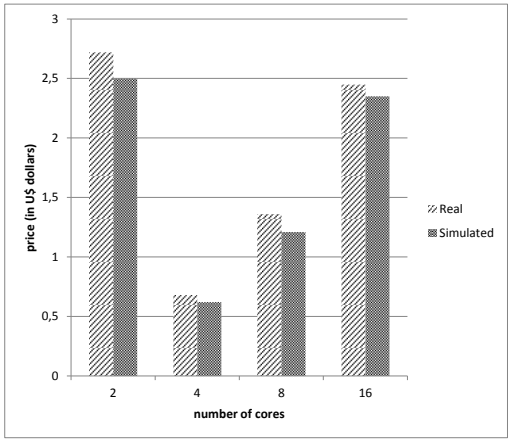

Figure 3 Comparison between the simulated monetary cost and the real cost.

6. Conclusion

Large scale scientific workflows usually present several executions using different parameters and input data. In order to produce results in a feasible time, scientists use parallel techniques to improve performance. For running these experiments in cloud environments, specific infrastructure such as SciCumulus is used. However, in its first version, SciCumulus creates the VMs following the decisions

of the scientist. Scientists have to inform the total number and type of necessary VMs which is not appropriate. There may be several combinations of types of VM to choose depending on the provider and this choice can be error-prone.

This work presents a cloud-based cost model used to help determining the best possible configuration of the cloud environment before scheduling cloud activities into those environments. This lightweight cost model aims at optimizing two criteria: total execution time of the workflow and monetary cost. The simulated results show that it is possible to use this model for determining an effective configuration following some QoS restrictions imposed by scientists. By optimizing the environment configuration we showed a reduction on the monetary cost up to 94% (using 16 VMs instead of 256) just reducing the number of VMs and satisfying the total execution time restriction.

Acknowledgements

The authors thank CNPq and CAPES for partially sponsoring our research.

7. References

[1] M. Mattoso, C. Werner, G.H. Travassos, V. Braganholo, L. Murta, E. Ogasawara, D. Oliveira, S.M.S.D. Cruz, and W. Martinho, 2010, Towards Supporting the Life Cycle of Large Scale Scientific Experiments, *International Journal of Business Process Integration and Management*, v. 5, n. 1, p. 79–92.

[2] R.D. Jarrard, 2001, *Scientific Methods*. Online book, Url.: http://emotionalcompetency.com/sci/booktoc.html.

[3] I.J. Taylor, E. Deelman, D.B. Gannon, M. Shields, and (Eds.), 2007, *Workflows for e-Science: Scientific Workflows for Grids*. 1 ed. Springer.

[4] E. Walker and C. Guiang, 2007, Challenges in executing large parameter sweep studies across widely distributed computing environments, In: *Workshop on Challenges of large applications in distributed environments*, p. 11-18, Monterey, California, USA.

[5] D. Oliveira, F. Baião, and M. Mattoso, 2010, "Towards a Taxonomy for Cloud Computing from an e-Science Perspective", *Cloud Computing: Principles, Systems and Applications (to be published)*, Heidelberg: Springer-Verlag

[6] T. Hey, S. Tansley, and K. Tolle, 2009, *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Online book, Url.: http://emotionalcompetency.com/sci/booktoc.html.

[7] L. Wang, J. Tao, M. Kunze, A.C. Castellanos, D. Kramer, and W. Karl, 2008, Scientific Cloud Computing: Early Definition and Experience, In: *10th IEEE HPCC*, p. 825-830, Los Alamitos, CA, USA.

[8] G. Juve and E. Deelman, 2010, Scientific workflows and clouds, *Crossroads*, v. 16 (Mar.), p. 14–18.

[9] Amazon EC2, 2010. Amazon Elastic Compute Cloud (Amazon EC2). *Amazon Elastic Compute Cloud (Amazon EC2)*. URL: http://aws.amazon.com/ec2/. Access: 5 Mar 2010.

[10] A. Campbell, G. Coulson, and D. Hutchison, 1994, A quality of service architecture, *ACM SIGCOMM Computer Communication Review*, v. 24 (Apr.), p. 6–27.

[11] R.L.D.C. Costa and P. Furtado, 2008, Scheduling in Grid Databases, In: *Advanced Information Networking and Applications Workshops, International Conference on*, p. 696-701, Los Alamitos, CA, USA.

[12] J. Dias, E. Ogasawara, D. Oliveira, E. Pacitti, and M. Mattoso, 2010, Improving Many-Task Computing in Scientific Workflows Using P2P Techniques, In: *MTAGS 2010*, p. 31-40, New Orleans, LA, USA.

[13] S.P. Callahan, J. Freire, E. Santos, C.E. Scheidegger, C.T. Silva, and H.T. Vo, 2006, VisTrails: visualization meets data management, In: *Proc. SIGMOD 2006*, p. 745-747, Chicago, Illinois, USA.

[14] D. Oliveira, E. Ogasawara, F. Baião, and M. Mattoso, 2010, SciCumulus: A Lightweigth Cloud Middleware to Explore Many Task Computing Paradigm in Scientific Workflows, In: *Proc. 3rd IEEE International Conference on Cloud Computing*, Miami, FL.

[15] M.T. Ozsu and P. Valduriez, 1999, *Principles of Distributed Database Systems*. 2 ed. Prentice Hall.

[16] R. Elmasri and S.B. Navathe, 2006, *Fundamentals of Database Systems*. 5 ed. Addison Wesley.

[17] D. Oliveira, E. Ogasawara, F. Baiao, and M. Mattoso, 2010, An Adaptive Approach for Workflow Activity Execution in Clouds, In: *International Workshop on Challenges in e-Science - SBAC*, p. 9-16, Petrópolis, RJ - Brazil.

[18] J. Freire, D. Koop, E. Santos, and C.T. Silva, 2008, Provenance for Computational Tasks: A Survey, *Computing in Science and Engineering, v.10*, n. 3, p. 11-21.

[19] R. Buyya, R. Ranjan, and R. Calheiros, 2009, Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges, In: *Proc. of HPCS 2009HPCS 2009*, Leipzig, Germany.

[20] G.C. Terstappen and A. Reggiani, 2001, In silico research in drug discovery, *Trends in Pharmacological Sciences*, v. 22, n. 1 (Jan.), p. 23-26.

[21] M. Duke, M. Day, R. Heery, L.A. Carr, and S.J. Coles, 2005, Enhancing access to research data: the challenge of crystallography, In: *Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*, p. 46–55, New York, NY, USA.

[22] Q. He, S. Zhou, B. Kobler, D. Duffy, and T. McGlynn, 2010, Case study for running HPC applications in public clouds, In: *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, p. 395–401, New York, NY, USA.