An Analysis of Metaheuristic to SLA Establishment in Cloud Computing

Leonildo J. de M. de Azevedo[1], Julio C. Estrella[1], Claudio F. M. Toledo[1] and Stephan Reiff-Marganiec[2]

[1] Institute of Mathematics and Computer Sciences, University of Sao Paulo, SP, Brazil
leonildo.azevedo@usp.br, {jcezar, claudio }@icmc.usp.br
[2] University of Leicester, Leicester, UK
srm13@leicester.ac.uk

Abstract. Nowadays the access to a cloud computing environment is provided on-demand offering transparent services to customers. Although the cloud allows an abstraction of the behavior of the service providers in the infrastructure (involving logical and physical resources), it remains a challenge to fully comply with the Service Level Agreements (SLAs), because, depending on the service demand and system configuration, the providers may not be able to meet the requirements of the customers. This paper proposes algorithms to address the need for optimization when handling computational resources before establishment the SLA.

Keywords: Cloud computing, SLA, optimization, IaaS

1 Introduction

In recent years, cloud computing has been one of the most widely discussed areas of Information Technology (IT). Cloud computing can also be regarded as an extension of other paradigms such as standard grade and utilitarian computing; this enables business applications to be viewed as sophisticated services which can be accessed by means of a network (the Internet) [5].

Cloud Computing is conceptually divided between three basic services models: **Software as a Service (SaaS)**, **Platform as a Service (PaaS)**, and **Infrastructure as a Service (IaaS)**. These models set out the capacities of the cloud services and how such services can be rendered to clients. This approach can be viewed as a division of rendered services in abstract layers [5].

Cloud Computing is a vast and complex computing paradigm that is closely bound up with the business dealings of its clients. It has always been a challenging task to provide clients with a suitable infrastructure for rendering services. Different clients require different resources depending on their hosted applications or demands made by users for these applications. For example, a particular client may have a CPU-Bound application while another client has a Memory-Bound application. One client might have an application with a large number of simultaneous accesses, whereas another client might have an application with only a few random accesses.

O. Kopp, J. Lenhard, C. Pautasso (Eds.): 9th ZEUS Workshop, ZEUS 2017, Lugano, Switzerland, 13-14 February 2017, published at http://ceur-ws.org/

The Cloud providers generally offer several configurations of virtualized resources (combinations of CPU, Memory and Disc space based on Virtual Machines (VMs)). Some configurations of virtual machines are predefined. Providers such as Amazon EC2 and Microsoft Azure employ a methodology in which the clients themselves are responsible for giving a precise estimate of the necessary resources and selecting the VM to be contracted [8].

It should be remembered that the users do not always have the technical knowledge to handle the provisioning of resources, not to mention the fact that this task can be burdensome for them. Therefore, the application of some kind of optimization technique for the provisioning of resources can make this an automated task. This can ensure the maximum use of computational resources and hence, the user need not to pay for more than she/he really needs. They will pay a fair price for the contracted service.

In this paper, our concern is with the IaaS service model and thus we have set out to create and evaluate optimization algorithms that fully comply with SLAs. We propose two optimization algorithms aimed at overcome the highlighted challenges, however, it is not a business evaluations. First, we describe a deterministic algorithm that is able to search exhaustively the problem solution space. Next, a micro-Genetic Algorithm (μGA) is proposed aiming to search more efficiently the solution space.

The paper presents two algorithms providing novel solutions for automating resource allocation in cloud computing and initial results of their performance.

2 Related Work

The cloud is a highly scalable environment in which the demand for services can change unexpectedly. Thus, the automatic allocation of resources to meet this demand is becoming an issue of great interest in both the academic and industrial world.

Correct provisioning allows for a better use of available computational resources and the whole infrastructure which comprises cloud, since it carries out a more efficient mapping between the loading and resources [2]. For example, the task of providing more computational resources to a client becomes more feasible and it can be made available more easily since the resources are virtualized. Moreover, while from the standpoint of the clients, the resources can be regarded as unlimited, it should be remembered that the allocation of more resources has an influence on the final cost. This cost has to be passed on to the client and it requires effective mechanisms on the the provider side.

There are several studies that analyze mechanisms for managing resources in a cloud environment. The suggestion of Amazon, that there should be an automatic reconfiguration of the infrastructure of the clients, is based on monitoring by means of Cloud-Watch Alarms and Scaling Policies. The scaling of Amazon Web Services (AWS) can be carried out by employing metrics which are generally based on the consumption of CPU time or on policies that are essentially divided between Manual Scaling, Dynamic Scaling and Scheduled Scaling [1].

In [11], a "central load balancer" is proposed for a large-scale cloud computing environment. However, there was a need to investigate how to implement other algorithms to spread the load in accordance with the use of resources, obtaining a more dynamic and robust load distribution. [12] proposes a scaling model of virtual resources based on the selection of a Non-dominated Sorting Genetic Algorithm (NSGA II). However, it is not evaluated failures found in a real environment, the needs of the client as opposed to those of the SLA and the available resources, or the costs incurred by the client.

There are many complex problems within the area of cloud computing that can be addressed by solutions that are found through optimization. For example, a) the problems of allocating virtual machines in a real machine [10]; b) energy saving [3]; c) scaling and load balancing of applications in virtual machines [9]; d) the use of resources aimed at reducing costs, while still guaranteeing a satisfactory performance [6]; and e) ensuring the QoS is in compliance with the SLA.

All these problems require optimal or sub-optimal solutions which can be achieved through techniques that are conceptualized within the area of optimization. This study seeks to tackle the problem of allocating resources in a suitable way by employing a heuristic and a meta-heuristic.

3 Methods

Some experiments were conducted and carried out in the CloudSim Simulator 3.0.3 3 version [3], with the aid of a computer with an AMD Phenom(tm) II X6 1090T Processor, with 16 GB of RAM memory, 1.5 TB of disc storage and the Ubuntu 14.04.3 LTS operational system with a kernel version 3.13.0. CloudSim was configured for the execution of three types of VM requests based on ***m3.medium, m3.large*** **and** ***m3.xlarge*** from Amazon EC2[4].

When the experiments were conducted, an SLA was stipulated for a particular client, the QoS attributes consisting of Capacity (C^c), Time (T^c), Availability (A^c) and Cost/hour (C/h^c). The simulation involves executing the client application in the capacity described in the SLA. As a result of the execution, the response time, availability and cost/h returned by CloudSim are obtained. If the parameters returned by the CloudSim are not compatible with the descriptions in the SLA, a meta-heuristic is applied to find a solution (Capacity (C^*), Time (T^*), Availability (A^*) of cost/h (C/h^*)) which can be best adapted to the requirements of the client. Thus, the proposed method will try to define for the provider the best arrangement of VMs ($s*, m*, l*$).

Two algorithms were designed to solve this problem: a deterministic algorithm (vide Algorithm 1) and a μGA (vide Algorithm 2). These methods will optimize the number of virtual machines contracted by the client. Thus, the representation of the solution (encoding) is defined by (s, m, l), which are the number of Virtual Machines (VMs) of type Small, Medium and Large, respectively, that will better satisfy client requests.

[3] http://www.cloudbus.org/cloudsim/

[4] https://aws.amazon.com/pt/ec2/instance-types/

The representation of solutions (s, m, l) are evaluated by CloudSim Simulator 3.0.3 3 version, that is configured for the execution of three types of VMs. As a result of its execution, the capacity (C^*), response time (T^*), availability (A^*) and cost/h (C/h^*) returned by CloudSim are obtained. If these parameter values are not compatible with those defined in the SLA (C^c, T^c, A^c, C/h^c), another representation of solution (s', m', l') must be evaluated. This compatibility is estimated by Equation(1).

$$f(P[i]) = \left|\frac{C^* - C^c}{C^c}\right| + \left|\frac{T^* - T^c}{T^c}\right| + \left|\frac{A^* - A^c}{A^c}\right| + \left|\frac{C/h^* - C/h^c}{C/h^c}\right| \quad (1)$$

The Manhattan Distance [4] is employed to estimate how close the solution is to the values stipulated in the SLA. There are four objectives with scale of values, so a standardized system is necessary based on Gap values.

The search space is defined from the number of machines adjusted by the client, where an amount 50% above or below the desired number of VMs are evaluated. For example, if the client contracted 50 VMs, the possible interval to explore remains between 25 and 75 VMs. In this case, representation of solutions as (s, m, l)=$(1, 20, 4)$ or (s, m, l)=$(0, 0, 75)$ are allowed since $25 \leq (s+m+l) \leq 75$. Thus, all possible combinations of VMs within this range of 50% are evaluated aiming to obtain the best value for Equation (1).

Algorithm 1 describes the deterministic algorithm, where all possible combinations for (s, m, l) are exhaustively evaluated and the best one is returned. Algorithm 2 shows the proposed μGA. This method is a genetic algorithm (GA) version which operates with a smaller-sized population and employs a convergence criterion that allows reinitialization.

Algorithm 1: Deterministic algorithm

```
   Input: SLA: C^c, T^c, A^c, C/h^c
1  //2. Sweep all the search space
2  repeat
3  |   //generate a capacity to be
   |     evaluated
4  |   Generate(s, m, l)
5  |   Evaluate(s, m, l)
6  until Until all (s, m, l)
     possibilities within the interval
     have been tested;
7  //return a better configuration
     found to satisfy the SLA of the
     client
8  return SLA^* : C^*, T^*, A^*, C/h^*
```

Algorithm 2: μGA algorithm

```
   Input: SLA: C^c, T^c, A^c, C/h^c
1  //1. Generate a population with 5 individuals
2  InitializePopulation(P)
3  //Assess the fitness of each individual in the
     population
4  Evalutate(P)
5  repeat
6  |   Selection(P)
7  |   Crossover(P)
8  |   Mutation(P)
9  |   Evaluate(P)
10 |   if the best individual is not updated after
   |     10 iterations then
11 |   |   Reinitialize(P)
12 |   end
13 until time limit has been reached;
14 //return the best individual
15 return SLA^* : C^*, T^*, A^*, C/h^*
```

The population was adjusted until it settled at only 5 individuals, each of which represents a possible VM configuration (s, m, l) for the client. The initialization generates randomly individuals (s, m, l) such as $min \leq (s+m+l) \leq max$, where the possible range is defined by $[min, max]$.

A tournament is applied to select between two individuals one to take part in crossover. The BlX-α crossover [7] is applied to generate new individuals. However, if $(s + m + l) \leq min$ or $max \leq (s + m + l)$ for the new individual,

the necessary amount is added or deduced from the last position (l). If it is not enough, the amount can be also reduced from position m.

4 Computational Results

The computational tests were divided into 10 scenarios compounded by different number of VMs available in a data center as shown in Table 1. We report the range to explore solutions $[min, max]$, the Execution Time spent to both algorithms and the total Number of combination evaluated by the deterministic algorithm for each scenario.

For a data center with 10 VMs, it was assumed that the client requested 5 VMS, so the range to be explored is $[2, 8]$ in this case. It can be observed that for 20 VMs, a fairly high execution time is already obtained in a scale of minutes (2.19 minutes). The time increases exponentially for the others scenarios as a result of the large number of VMs. The execution takes longer than 20 minutes for $VMs > 50$. The deterministic algorithm is able to find the optimal solution for this problem once it explores completely the solution space. However, these results indicates that the method is not viable for practical purposes even for a small number of VMs.

The μGA was executed 100 times for each scenario with 3 minutes as stop criterion by execution. It was assumed that 3 minutes is a reasonable time to reply a client request. After all executions, the success rate achieved by μGA was estimated by comparing the best solution of μGA with the optimal solution found by the deterministic algorithm. A success is computed if μGA finds the optimal solution in some run. The proposed μGA was able to find the optimal solutions for all scenarios in all 1000 executions. Table 1 reports the average Time with standard deviations to find the optimal solution.

This correctness rate was obtained in less than 1 minute for all scenarios by μGA, in contrast with the deterministic method that took more than 1 minute in all cases, except by VMs=10. The mean of μGA time was computed based on 100 executions. The standard deviation in some cases is greater than the average, due the significative discrepancy among the data.

Table 1. Response time to the μGA and Deterministic model of 10 scenarios scenario from 10 until 100 available VMs.

Number of VMs	$[min, max]$	**Execution Time (minutes)**	**Time μGA (minutes)**	**Number of combinations**
10	$[2, 8]$	0.35	0.13 ± 0.15	63
20	$[5, 15]$	2.19	0.54 ± 0.32	342
30	$[7, 22]$	6.28	0.19 ± 0.19	1330
40	$[10, 30]$	14.83	0.32 ± 0.21	2743
50	$[12, 37]$	27.08	0.56 ± 0.24	4912
60	$[15, 45]$	47.53	0.17 ± 0.19	9260
70	$[17, 52]$	72.38	0.14 ± 0.17	13823
80	$[20, 60]$	108.56	0.17 ± 0.20	19682
90	$[22, 67]$	150.25	0.19 ± 0.21	29790
100	$[25, 75]$	213.86	0.16 ± 0.16	39303

5 Conclusion

It is not a trivial task to provide a cloud computing user with an efficient infrastructure, that respects the SLA and its QoS attributes, while at the same time seeking to reduce costs. Within the domain of cloud computing, the most wide-ranging problems can be mapped out in solutions that generally involve optimization based on their complexity and the large number of resources that can be scalable.

This article addresses one of these challenges which was to provide the client with a self-manageable infrastructure with the provision of SLA agreed between the client and provider. Our proposal mapped some of the QoS attributes that determine the criteria for the SLA and we also designed and analyzed algorithms that allow an optimized reconfiguration of the infrastructure based on these criteria. The results provide evidence that the μGA algorithm is efficient and applicable to the solution of the problem.

In future work, we intend to carry out new tests with a bigger number of VMs and clients. We are also going to design new meta-heuristic algorithms so that a comparison can be made between them, and at the monitoring phase. Other QoS attributes will be added to the SLA and new constraints to the problem such as defining a maximum cost threshold that the client is able to afford.

Acknowledgments The authors thank CNPq, CAPES and FAPESP (processes IDs: 16/14219-2 and 15/11623-4) for the financial support funding this research work.

References

1. Amazon: Scaling the size of your auto scaling group (2015), available at: http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/scaling_plan.html. Last Access: 01/20/2015
2. Batista, B.G., Estrella, J.C., Ferreira, C.H.G., Leite Filho, D.M., Nakamura, L.H.V., Reiff-Marganiec, S., Santana, M.J., Santana, R.H.C.: Performance evaluation of resource management in cloud computing environments. PloS one 10(11), 21 (2015)
3. Beloglazov, A., Abawajy, J., Buyya, R.: Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Generation Computer Systems 28(5), 755 – 768 (2012), http://www.sciencedirect.com/science/article/pii/S0167739X11000689, special Section: Energy efficiency in large-scale distributed systems
4. Black, P.E.: Manhattan distance. Dictionary of Algorithms and Data Structures 18, 2012 (2006)
5. Buyya, R., Broberg, J., Goscinski, A.M.: Cloud Computing Principles and Paradigms. Wiley Publishing (2011)
6. Byun, E.K., Kee, Y.S., Kim, J.S., Maeng, S.: Cost optimized provisioning of elastic resources for application workflows. Future Generation Computer Systems 27(8), 1011 – 1026 (2011), http://www.sciencedirect.com/science/article/pii/S0167739X11000744

7. Eiben, A., Smith, J.: Introduction to Evolutionary Computing. Natural Computing Series, Springer Berlin Heidelberg (2007), https://books.google.com.br/books?id=7IOE5VIpFpwC
8. Huu, T.T., Montagnat, J.: Virtual resources allocation for workflow-based applications distribution on a cloud infrastructure. In: Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on. pp. 612–617. IEEE (2010)
9. L.D., D.B., Krishna, P.V.: Honey bee behavior inspired load balancing of tasks in cloud computing environments. Applied Soft Computing 13(5), 2292 – 2303 (2013), http://www.sciencedirect.com/science/article/pii/S1568494613000446
10. Masdari, M., Nabavi, S.S., Ahmadi, V.: An overview of virtual machine placement schemes in cloud computing. Journal of Network and Computer Applications pp. – (2016), http://www.sciencedirect.com/science/article/pii/S1084804516000291
11. Soni, G., Kalra, M.: A novel approach for load balancing in cloud data center. In: Advance Computing Conference (IACC), 2014 IEEE International. pp. 807–812. IEEE (2014)
12. Zhao, J., Zeng, W., Liu, M., Li, G.: Multi-objective optimization model of virtual resources scheduling under cloud computing and it's solution. In: Cloud and Service Computing (CSC), 2011 International Conference on. pp. 185–190. IEEE (2011)