Self-economy in Cloud Data Centers: Statistical Assignment and Migration of Virtual Machines

Carlo Mastroianni[1], Michela Meo[2], and Giuseppe Papuzzo[1]

[1] ICAR-CNR, Rende (CS), Italy
{mastroianni,papuzzo}@icar.cnr.it
[2] Politecnico di Torino, Italy
michela.meo@polito.it

Abstract. The success of Cloud computing has led to the establishment of large data centers to serve the increasing need for on-demand computational power, but data centers consume a huge amount of electrical power. The problem can be alleviated by mapping virtual machines, VMs, which run client applications, on as few servers as possible, so that some servers with low traffic can be put in low consuming sleep modes. This paper presents a new approach for the adaptive assignment of VMs to servers and their dynamic migration, with a twofold goal: reduce the energy consumption and meet the Service Level Agreements established with users. The approach, based on ant-inspired algorithms, founds on statistical processes: the mapping and migration of VMs are driven by Bernoulli trials whose success probability depends on the utilization of single servers. Experiments highlight the two main advantages with respect to the state of the art: the approach is self-organizing and mostly decentralized, since each server locally decides whether or not a new VM can be served, and the migration process is continuous and adaptive, thus avoiding the need for the simultaneous reassignment of many VMs.

1 Introduction

The need for on-demand computing, i.e., the possibility of using computational resources on a pay-as-you-go basis, was identified many years ago, but so far it is has been hindered by technological constraints. Recently, the availability of powerful data centers and high bandwidth connections have expedited the success of the Cloud computing paradigm, which is making on-demand computing a common practice for many enterprizes and scientific communities. The main advantage of this paradigm is that a company does not need to operate its own data center, with all the related costs and administration burdens, but can access to CPU power, storage facilities, software packages on the basis of current needs. For example, a Web server operated by a company can be hosted by a Cloud center, a choice that has many advantages in addition to money savings, among which higher security and availability guarantees (anti-hacker and back up procedures are managed by IT professionals), and much lower or even zero risks of under- or over-provisioning of resources. These advantages are particularly welcome by small companies, especially in their start up phase [3].

E. Jeannot, R. Namyst, and J. Roman (Eds.): Euro-Par 2011, LNCS 6852, Part I, pp. 405–416, 2011.

One of the main issues related to the success of Cloud computing is that the ever growing number of large data centers is causing a notable increase of electrical power consumed by hardware facilities and cooling systems. This increases the cost of computation itself and affects the carbon footprint of data centers, thus aggravating, on the global scale, the problem of global warming. It has been estimated that in 2006 the energy consumed by IT infrastructures in USA was about 61 billion kWh, corresponding to 1.5% of all the produced electricity, and these figures are expected to double by 2011 [2].

A major reason for this huge amount of consumed power is the inefficiency of data centers, which are often under-utilized: it has been estimated that only 20-30% of the total server capacity is used on average [1]. Despite the adoption of techniques that try to scale the energy consumption with respect to the actual utilization of a computer (for example, Dynamic Voltage and Frequency Scaling or DVFS), an idle server still consumes approximately 65-70% of the power consumed when it is fully utilized [8]. To cope with this problem, Cloud data centers exploit the *virtualization* paradigm: user processes are not assigned directly to servers, but are first associated with Virtual Machine (VM) instances, which, in turn, are run by servers. The use of virtualization allows heterogeneous platforms to be executed on any kind of hardware facility, which facilitates the *consolidation* of VMs, that is, their clustering on as few computers as possible.

Unfortunately, the optimal mapping of VMs to servers, so as to minize energy consumption, is an NP-hard problem and requires a full knowledge of the servers load. Centralized algorithms that explore sub-optimal solutions may be computationally costly, and do not scale well with the size of the system. This paper presents a self-organizing approach that is partly inspired by the basic ant algorithms used by Deneubourg et al. [6] to model the phenomenon of larval clustering in ant colonies. In our case, the approach aims at clustering VMs in as few servers as possible, using two types of statistical procedures, for the *assignment* and the *migration* of VMs. Specifically, a new VM is assigned to one of the available servers through statistical Bernoulli trials for which the success probability depends on the current utilization of the servers. The *assignment probability function* is defined so as to favor the assignment of a VM to a highly loaded server, in order to improve VM consolidation. On the other hand, the migration procedure fosters the migration of VMs from servers in which the current utilization is either too high or too low, that is, above or below two defined thresholds. In the first case, the migration of a VM helps to prevent a possible overload of the server, which may lead to Service Level Agreement (SLA) violations. In the second case, the objective of the migration is to take VMs away from lightly loaded servers, and then power off these servers. Migration is also driven by Bernoulli trials, for which the success probability is defined by appropriate *migration probability functions*.

The use of statistical processes has two important advantages: (i) assignment and migration processes are self-organizing and mostly decentralized. The data center manager coordinates the processes, but decisions are taken locally by each server on the basis of local information; (ii) the migration process is continuous and gradual, and the cost of migration (e.g., performance degradation) is smoothed over time, so that the quality of service perceived by users is hardly affected. Conversely, several approaches proposed in the literature (e.g., [12] and [2]) often require the simultaneous migration of many VMs. These properties, self-organization and gradual migration of VMs, favor the scalability of the approach and its capacity to adapt to the dynamic workload of client applications.

The rest of the work is organized as follows: Section 2 discusses the assignment and migration procedures; Section 3 reports the results of simulation experiments, which prove that the approach succeeds in efficiently consolidating VMs, and that power consumption is close to the theoretical minimum, while the number of SLA violations is minimized; Section 4 describes related work, and Section 5 concludes the paper and proposes some avenues for future work.

2 Assignment and Migration of Virtual Machines

This section describes the statistical procedures used for the assignment of VMs to the data center servers and for their dynamic migration. The examined scenario is pictured in Figure 1: an application request is transmitted from a client to the data center manager, which selects a VM that is appropriate for the application, on the basis of application characteristics such as the amount of required resources (CPU, memory, storage space) and the type of operating system specified by the client. Then, the VM is assigned to one of the available multi-core servers through the *assignment procedure*. The workload of the application is dynamic, that is, its demand for CPU varies with time, provided that it does not exceed the VM capacity. This is typical, for example, of Web servers, for which the CPU demand depends on the workload generated by Web users. Periodically, each server checks if its CPU utilization is between the specified upper and lower thresholds and, when this condition is violated, it activates the *migration procedure*, in order to move one VM to another server. The parameters λ and μ shown in Figure 1 are, respectively, the arrival rate of application requests and the service rate of a server core. They will be used in Section 3 for the performance analysis.

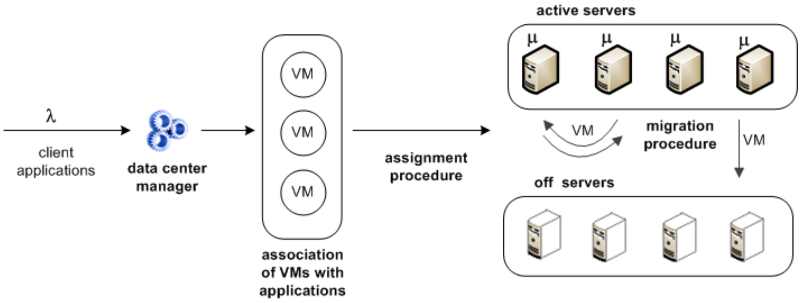

Fig. 1. Assignment and migration of VMs in a data center

2.1 Assignment Procedure

Once a client application is associated with a compatible VM, the latter must be assigned to a server for execution. The choice should take into account the following considerations: (i) it is preferable to assign the VM to a server with high CPU utilization, in order to enforce the consolidation of VMs and possibly allow idle servers to be powered off; (ii) the CPU utilization should not be too close to the server capacity: in such a case, if the VMs workload increases, the server may be unable to grant the amount of CPU required by the applications, and SLA violations may occur; (iii) the VM should be allocated on a powered off server only when strictly necessary, since switching on a server reduces consolidation and increases consumed power.

Given these objectives, the *assignment procedure* is defined as follows. The data center manager broadcasts the assignment request to servers[1]. Each active server executes a Bernoulli trial, whose success probability depends on its current CPU utilization, u (valued between 0 and 1), and on the maximum allowed utilization, T_a. The *assignment probabilistic function*, $f_{assign}(u)$, is null when $u > T_a$, otherwise it is defined as:

$$f_{assign}(u) = 1/M_p \cdot u^p \cdot (T_a - u) \qquad M_p = \frac{p^p}{(p+1)^{(p+1)}} \cdot T_a^{(p+1)} \tag{1}$$

Figure 2 shows the function graph for some values of the integer parameter p, and $T_a = 0.9$. The factor $1/M_p$ is used to normalize the maximum value to 1. The function definition ensures that the CPU utilization cannot exceed the threshold T_a (because no further VMs can be assigned when u reaches this threshold) and that VMs are preferably assigned to highly loaded servers, thus favoring consolidation. The value of u at which the function reaches its maximum - that is, the value at which assignment attempts succeed with the highest probability - is $p/(p+1) \cdot T_a$, which increases and approaches T_a as the value of p increases. Therefore, the value of p can be used to modulate the shape of the function and tune the consolidation effort.

Each server for which the Bernoulli trial succeeds, responds to the broadcast message declaring its availability to accommodate the VM. Then, the data center manager randomly selects one of these available servers, and assigns the VM to it. If all active servers are unavailable - because their utilization exceeds the threshold T_a, or because Bernoulli trials are unsuccessful - an inactive server will be switched on and will accommodate the VM. If this is not possible, because all the servers are already active, the VM will be forcedly assigned to any server that has some spare CPU fraction (such a server may be chosen after a second broadcast request), or it will be put in a waiting queue: this is a hint that the number of servers is too low to sustain the load.

[1] The broadcast strategy is the most reasonable when all the servers are located in a single high-speed network. If the servers are grouped in multiple clusters, a more efficient alternative can be to forward the request only to a subset of servers. The behavior is nearly equivalent.

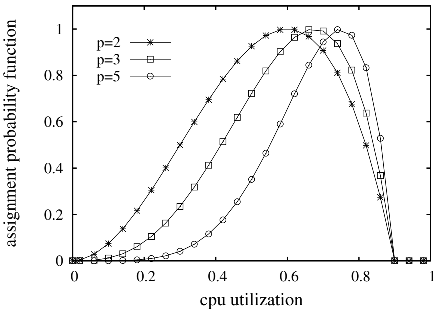

Fig. 2. Assignment probability function $f_{assign}(u)$ for different values of the parameter p. The value of the threshold T_a is set to 0.9.

One of the main advantages of the approach can now be appreciated, that is, its self-organizing and decentralized nature, since main decisions are taken locally. The manager is only required to know which servers are active and which are switched off, and to perform the random selection among the servers that are available to accommodate a new VM. The manager, though, is not requested to perform any algorithm to decide how to distribute the VMs to servers, nor to keep updated information about servers' state.

2.2 Migration Procedure

The assignment procedure allows VMs to be clustered in a reduced number of servers, as is shown in the performance evaluation section. Nevertheless, it can still happen that some servers are under-utilized and might be switched off. Indeed, even after an efficient allocation of VMs to computing resources, the VMs running in a server may terminate or may reduce their demand for CPU. Moreover, overload situations can also occur. In fact, the assignment of a VM to a server is made on the basis of its current CPU utilization, but the workload of other VMs in the same server may subsequently increase. This can cause SLA violations, thus affecting the degree of dependability of the data center and the quality of service offered to users. In both these situations, some VMs can be profitably migrated to other servers, either to switch off a server, or to alleviate its load.

Live migration of VMs is driven by the *migration procedure*. As opposed to other techniques recently proposed for VM migration (see the related work section), our approach is self-organizing and ensures a gradual and continuous migration process. At random time intervals, each server checks whether it is under-utilized or over-utilized and, when this occurs, evaluates the corresponding migration probability function, $f^l_{migrate}(u)$ or $f^h_{migrate}(u)$:

$$f^l_{migrate}(u) = (1 - u/T_l)^\alpha \tag{2}$$

$$f^h_{migrate}(u) = (1 + \frac{u-1}{1-T_h})^\beta \tag{3}$$

In either case, the server performs a Bernoulli trial and decides whether or not to request the migration of one of the local VMs. The functions, shown in Figure 3, are defined so as to trigger the migration of VMs when the CPU utilization is, respectively, below the threshold T_l or above the threshold T_h. When the utilization is in between, migrations are inhibited. The shape of the functions can be modulated by tuning the parameters α and β, which can therefore be used to foster or hinder migrations. A migration procedure completes when the VM is successfully assigned to another server, using the assignment procedure described in Section 2.1. In the case of the migration from an overloaded server, the threshold T_a of the assignment function is set to 0.9 times the CPU utilization of the current server. This ensures that the VM migrates to a less loaded server, and prevents situations in which a VM is continuously migrated from an overloaded server to another. The new value of T_a is sent to the other servers along with the migration request, and the VM is assigned to one of the available servers. If no server is available, the VM is kept by the original server.

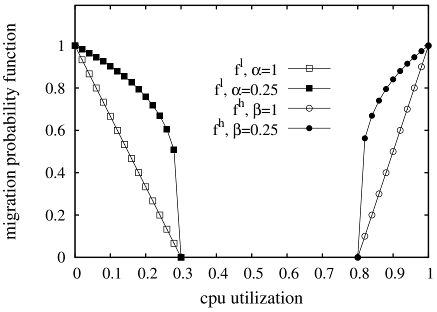

Fig. 3. Migration probability functions $f^l_{migrate}(u)$ and $f^h_{migrate}(u)$, labeled as f^l and f^h, for two values of α and β. The threshold T_l is set to 0.3, T_h is set to 0.8.

3 Performance Evaluation

The approach described in the previous section was tested with a Java simulator implemented at ICAR-CNR. The evaluated data center has N_s=100 servers, 33 of which have 4 cores, 34 have 6 cores and 33 have 8 cores. All cores have CPU frequency of 2 GHz. The Virtual Machines that host client applications have nominal CPU frequencies of 500 MHz, 1 GHz and 2 GHz. They are assigned to applications with the following probability distribution: 50% of applications are assigned to 500 MHz VMs, 25% to 1 GHz VMs, and 25% to 2 GHz VMs.

Each VM runs for a time interval generated with a Gamma distribution and average $1/\mu$ set to 100 minutes, where μ is the service rate of each server core. During its execution, the application hosted by the VM can require all the VM capacity or a fraction of it. This fraction can vary over time, as Cloud applications - in many cases Web servers - usually experience dynamic workload.

For each application, the average interval between workload changes is set to 20 minutes (with Gamma distribution), and after each interval the fraction of the VM capacity demanded by the application is extracted uniformly between 0 and 1. Requests for client applications are received by the data center manager at rate λ (see Figure 1), whose value ranges between 1.2 and 24 requests per minute. The average load of the data center, denoted as ρ, can be computed as $0.5\lambda/\mu_T$. Here, the arrival rate of requests is halved because applications ask on average for half the capacity of a VM, while $\mu_T = \mu \cdot N_s \cdot 6 \cdot 2$ is the overall service rate of the data center: indeed, the average number of cores per server is 6, and the capacity of each core (2 GHz) is twice the average frequency of a VM (1 GHz). The parameter ρ ranges between 0.05 (nearly idle data center) and 1 (data center loaded at its maximum CPU capacity), and will be used to analyze the system performance in different load conditions. Servers can be dynamically activated and powered off: an inactive server is switched on when it is asked to accommodate a new or a migrating VM; an active server is switched off (or hibernated) when all the running VMs terminate or when they are migrated to other servers.

To analyze the amount of consumed power, the model described by some recent studies (e.g., [8] and [2]) is adopted. Specifically, the power consumed by servers can be obtained with a simple relationship between CPU utilization and power consumption, assuming that an idle server consumes about 70% of the power consumed by a fully utilized server. The power consumption is expressed as $P(u) = P_{max} \cdot (0.7 + 0.3 \cdot u)$. In our tests, P_{max}, the power consumed at maximum utilization, is set to 250 W.

Figure 4 reports the average number of active servers, in steady condition, vs. the system load, when setting the threshold T_a of the assignment function (1) to 0.9, and with the following parameter setting for the migration functions (2) and (3): T_l=0.2, T_h=0.95, α=0.25 and β=0.25. The migration procedure is evaluated by each server every 10 minutes. The figure reports results obtained with different values of the parameter p of the assignment function. For comparison purposes, the results are shown together with three other curves. The first is the average number of servers activated when each VM is randomly assigned to one of the servers, regardless of their current utilization. The related curve is by far the highest, and has a typical negative exponential trend. The VM mapping problem can be formulated in terms of the *bin packing problem*, i.e., the NP-hard problem of allocating objects of heterogeneous sizes in as few bins as possible [12]. The second curve corresponds to the optimal solution of this problem, i.e., when the minimum number of servers is used to accommodate the VMs. The curve labeled as *BFD* corresponds to the performance achievable when the bin packing problem is solved with the Best Fit Decreasing algorithm, which has quadratic complexity and guarantees to use at most 11/9 MIN+1 servers, where MIN is the minimum number of servers [13].

Figure 4 shows that our approach performs better than the BFD, especially when the load is high, and that the number of active servers is very close to the optimal curve. Of course, reducing the number of active servers allows the data

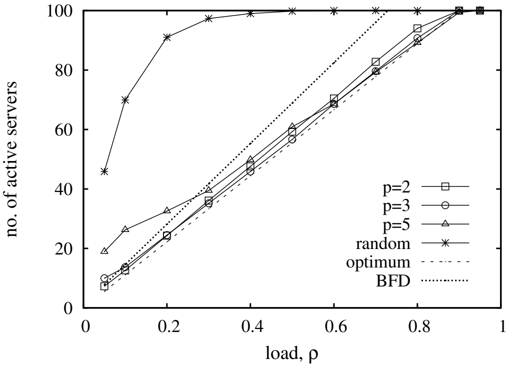

Fig. 4. Average number of active servers for different values of the parameter p of the assignment function. The meaning of the other curves is explained in the text.

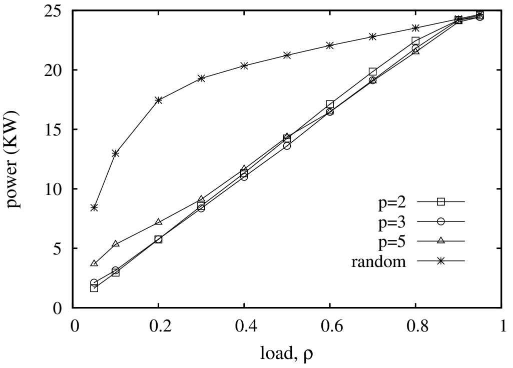

Fig. 5. Average power consumed by the data center for different values of the parameter p of the assignment function

center to save power, as appears in in Figure 5. The “green” behavior of our approach is testified by the fact that consumed power increases almost linearly with load. The two figures also show that the performance is not very sensitive to the value of p, which is a sign of robustness. Nevertheless, larger values of p can be used to improve consolidation (and reduce power consumption) in high load conditions, because they increase the probability of allocating a VM to a highly loaded server. Conversely, a low value of p is preferable when the load is low. The tuning of p can be done dynamically by the data center manager, by estimating the overall system load. In the next experiments, the parameter p is set to 3, as this value ensures a good behavior for all load conditions.

As explained before, VM migrations can be performed either because the utilization of a server is too low or too high. In the following, the migration events of the two kinds are referred, respectively, as *l_migrations* and *h_migrations*. Any migration causes a slight performance degradation of the application hosted by the VM for the time necessary to migrate, which in general is estimated in the

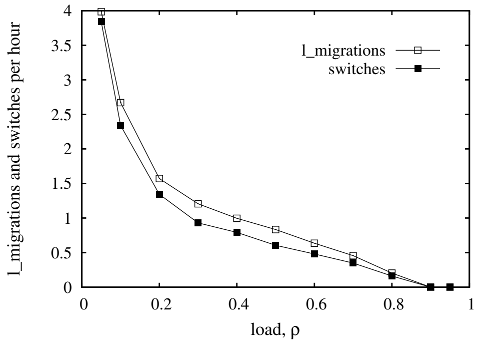

Fig. 6. Frequency of l_migrations and server switches vs. load.

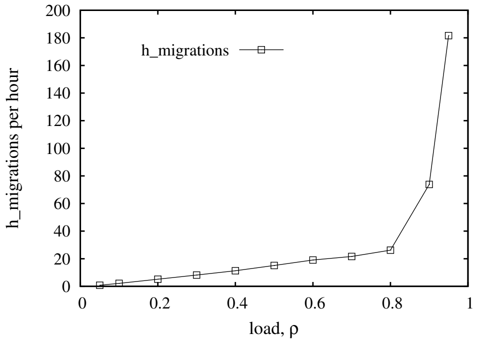

Fig. 7. Frequency of h_migrations vs. load.

order of 60 seconds [9]. Similarly, the activation of an off server needs a start up time and additional power. Therefore, it is important to limit the frequency of migrations and switches, though a certain number is essential for VM consolidation and power reduction. Figure 6 reports the frequencies of l_migrations and server activations experienced in the whole data center. Both frequencies are inversely proportional to the load. In fact, with high load, most servers are always active and highly loaded, so both events are impossible or rare. With low load, many servers are off, and the assignment procedure has more chances to assign a VM to an inactive server, which is then switched on. Since this server is initially under-utilized, it will likely attempt a migration procedure in the near future, which explains the higher migration frequency. Both frequencies are always lower than 4 events per hour in the whole data center, which is an easily sustainable burden.

Conversely, the frequency of h_migrations, reported in Figure 7, is directly proportional to the load. The trend is nearly linear, but becomes exponential when the load approaches the data center capacity: this suggests that new servers should be acquired when the load exceeds 0.8. Finally, Figure 8 reports the percentage of time in which the VMs allocated to a server demand more CPU than what the server can provide, which may lead to SLA violations. This index, in accordance with recent studies [2], is used to measure the QoS level offered to users. Conditions for potential SLA violations are rare when ρ is lower than 0.8, then their frequency increases rapidly. The figure reports the index values obtained with and without the use of h_migrations, which clearly testifies the beneficial impact of these events.

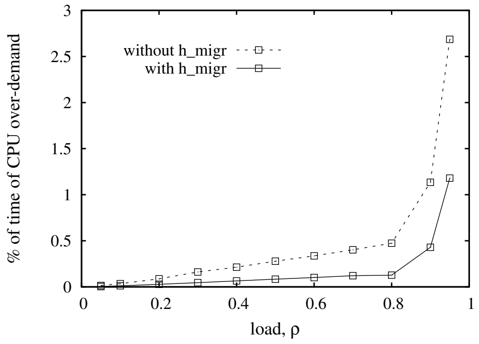

Fig. 8. Percentage of time in which the VMs require more CPU than that offered by a server, with and without the use of h_migrations

4 Related Work

As the Cloud computing paradigm rapidly emerges, a notable amount of studies focus on algorithms and procedures that aim at improving the "green" characteristics of Cloud data centers. A common aspect is the use of virtualization as a means to consolidate applications on as few servers as possible and in this way reduce power consumption. Some approaches - e.g., [4] and [10] - try to forecast the demand and aim at determining the minimum number of servers that should be switched on to satisfy the client requests, so as to reduce energy consumption and maximize data center revenue. However, even a correct setting of this number is only a part of the solution: algorithms are needed to decide how the VMs should be mapped to servers in a dynamic environment, and how live migration of VMs can be exploited to unload servers and switch them off when possible, or to avoid SLA violations.

As mentioned in Section 3, the problem of dynamically mapping VMs to servers is in some way similar to the *bin packing problem*, and the analogy is indeed exploited in recent research, for example in [12] and in [2]. Live migration of VMs between servers is adopted in [2] and by the VMWare Distributed Power Management system, using lower and upper utilization thresholds to enact migration procedures. All these approaches represent important steps ahead for the deployment of green-aware data centers, but still they share a couple of notable drawbacks: (i) the centralized manager is required to execute complex algorithms and solve a problem that is inherently NP-hard, and must always be aware of the state of all the servers, which becomes an issue in large and highly dynamic data centers; (ii) mapping strategies may require the concurrent migration of many VMs, which can cause considerable performance degradations during the reassignment process. Conversely, the approach presented here is self-organizing, decentralized for the most part (assignment and migration decisions are taken autonomously by each server), and uses a gradual migration process.

Bio-inspired algorithms and protocols are emerging as a useful means to manage distributed systems. Assignment and migration procedures presented here are partly inspired by the *pick* and *drop* operations performed by some species of ants that cluster items in their environment [6]. The pick and drop paradigm, though very simple and easy to implement, has already proved to be surprisingly powerful: for example, it was used to cluster and order resources in P2P networks, in order to facilitate their discovery [7]. Another ant-inspired mechanism was proposed in [5]: in this study, the data center is modeled as a P2P network, and ant-like agents explore the network to collect information that can later be used to migrate VMs and reduce power consumption. Since the mapping of VMs to servers is essentially an optimization problem, evolutionary and genetic algorithms can also represent a valid solution. In [11], a genetic algorithm is used to optimize the assignment of VMs, and minimize the number of active servers. The main limitations of this kind of approach are the need for a strong centralized control and the difficulties in the setting of key parameters, such as the population size and the crossover and mutation rates.

5 Conclusion and Future Work

This paper presents an approach that aims at minimizing the number of active servers and reducing power consumption in Cloud data centers. The core of the proposal stands in the statistical and self-organizing procedures that are used to assign Virtual Machines to servers, and to migrate them when this helps either to power off under-utilized computers or to prevent possible SLA violations in highly loaded servers. Simulation experiments show that the adopted techniques succeed in the combined objective of reducing power consumption and ensuring a good level of the QoS experienced by users, but the novelty of the approach requires further research to better assess its performance and explore its potentialities. Some of the avenues are: (i) a deeper analysis of the sensitivity to parameter values; (ii) a study of scalability properties; preliminary evaluations are promising, as performance seems to improve with the data center size, which is not surprising given the statistical nature of the algorithms; (iii) adapt assignment and migration procedures to take into account not only the CPU utilization of servers, but also other aspects such as the necessity of assigning several VMs to the same server, when they need to cooperate with each other; (iv) the definition of mathematical models, which may help in giving a more formal foundation to the approach.

References

1. Barroso, L.A., Hölzle, U.: The case for energy-proportional computing. IEEE Computer 40(12), 33–37 (2007)
2. Beloglazov, A., Buyya, R.: Energy efficient allocation of virtual machines in cloud data centers. In: 10th IEEE/ACM Int. Symp. on Cluster Computing and the Grid, CCGrid 2010, pp. 577–578 (2010)
3. Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Compututer Systems 25(6), 599–616 (2009)
4. Chen, Y., Das, A., Qin, W., Sivasubramaniam, A., Wang, Q., Gautam, N.: Managing server energy and operational costs in hosting centers. SIGMETRICS Perform. Eval. Rev. 33(1), 303–314 (2005)
5. Dubois, D.J., Mirandola, R., Barbagallo, D., Di Nitto, E.: A bio-inspired algorithm for energy optimization in a self-organizing data center. In: Self-Organizing Architectures. Springer, Heidelberg (2010)
6. Deneubourg, J.L., Goss, S., Franks, N., Sendova-Franks, A., Detrain, C., Chrétien, L.: The dynamics of collective sorting: robot-like ants and ant-like robots. In: First International Conference on Simulation of Adaptive Behavior on From Animals to Animats, pp. 356–363. MIT Press, Cambridge (1990)
7. Forestiero, A., Mastroianni, C., Spezzano, G.: So-grid: A self-organizing grid featuring bio-inspired algorithms. ACM Transactions on Autonomous and Adaptive Systems 3(2) (May 2008)
8. Greenberg, A., Hamilton, J., Maltz, D.A., Patel, P.: The cost of a cloud: research problems in data center networks. SIGCOMM Comput. Commun. Rev. 39(1), 68–73 (2009)
9. Hirofuchi, T., Ogawa, H., Nakada, H., Itoh, S., Sekiguchi, S.: A live storage migration mechanism over wan for relocatable virtual machine services on clouds. In: 9th IEEE/ACM Int. Symp. on Cluster Computing and the Grid, CCGrid 2009 (2009)
10. Mazzucco, M., Dyachuk, D., Deters, R.: Maximizing cloud providers' revenues via energy aware allocation policies. In: 10th IEEE/ACM Int. Symp. on Cluster Computing and the Grid, CCGrid 2010, pp. 131–138 (2010)
11. Mi, H., Wang, H., Yin, G., Zhou, Y., Shi, D., Yuan, L.: Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers. In: 2010 IEEE Int. Conference on Services Computing, SCC 2010, Miami, Fl, USA, pp. 514–521 (July 2010)
12. Verma, A., Ahuja, P., Neogi, A.: pMapper: Power and migration cost aware application placement in virtualized systems. In: Issarny, V., Schantz, R. (eds.) Middleware 2008. LNCS, vol. 5346, pp. 243–264. Springer, Heidelberg (2008)
13. Yue, M.: A simple proof of the inequality FFD (L) $\leq$ 11/9 OPT (L) + 1, for all L for the FFD bin-packing algorithm. Acta Mathematicae Applicatae Sinica 7(4), 321–331 (1991)