Design and Implementation of a Multi-objective Optimization Mechanism for Virtual Machine Placement in Cloud Computing Data Center

Soichi Shigeta, Hiroyuki Yamashima, Tsunehisa Doi,
Tsutomu Kawai, and Keisuke Fukui

Cloud Computing Research Center, Fujitsu Laboratories Ltd.
4-1-1, Kamikodanaka, Nakahara-ku, Kawasaki 211-8588, Japan
{shigets,yama,micky,tkawai,kfukui}@labs.fujitsu.com

Abstract. Cloud computing is becoming a popular way of supplying and using computing resources. A cloud-computing data center is equipped with a large number of physical resources and must manage an even larger number of virtual machines (VMs). The center's VM placement strategy affects the utilization of physical resources, and consequently, it influences operational costs. Our goal is to develop a multi-objective optimization mechanism for VM placement that satisfies various constraints and results in the lowest operational cost. The number of possible combinations of VMs and hosts can be extremely large. For the mechanism to be practical, the number of possible combinations must be reduced. We reduced computational overheads by classifying VM hosts into a relatively small number of equivalent sets. Simulation results show that expected operational costs can be significantly reduced by applying the proposed mechanism.

Keywords: cloud computing, VM placement, multi-objective optimization.

1 Introduction

Cloud computing is becoming an increasingly popular way of supplying and using computing resources. A number of commercial cloud services, such as Amazon EC2 [1] and S3 [2], Google AppEngine [3], Salesforce CRM [4], and Fujitsu Global Cloud Platform [5] are presently being used to run business systems. Cloud services can be classified into three types: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). For example, Fujitsu Global Cloud Platform and Amazon EC2 are examples of IaaS. Google AppEngine is an example of PaaS, and Salesforce CRM is an example of SaaS. In this paper, our main focus is on an IaaS data center with particular emphasis on reducing operational costs.

Reducing operational costs is a key to achieve high cost-benefit performance in cloud computing data centers. Lower operational costs are also important for price competitiveness because they will be reflected in the price of a service.

Accordingly, the aim of this research is to develop a multi-objective mechanism to determine the lowest cost for virtual machine (VM) placement that considers various components of cost efficiency (such as electrical power consumption, availability, and load balancing).

Traditional systems have been designed and built to satisfy the peak load that was estimated in advance. However, although periodic fluctuations for business workloads may be predictable, it is difficult to predict fluctuation of demand for a public cloud. Load balancing among physical servers is a typical criterion to increase the efficiency in cloud computing data centers. Therefore, many existing resource schedulers determine VM placement on the basis of utilization of physical servers (typically CPU and memory usage) or VM hosts. However, efficient and effective VM placement involves many other factors, and existing resource schedulers do not give these additional factors adequate consideration because multi-objective optimization is an extremely complex problem. If this problem is addressed in a straightforward way, an astronomical number of possible combinations will be generated because of the large number of physical servers and VMs in a cloud computing data center; i.e., a huge number of time-consuming computations are required to obtain an optimum result.

Consequently, timely optimization is a challenge. A VM deployment request from a user should be completed within several minutes (including time to boot up VM). Therefore, a very limited amount of time is available to determine an optimum VM placement.

2 Various Constraints to Virtual Machine Placement

In addition to the physical capacity of each host (i.e., CPU and memory), resource scheduling often has to consider various other constraints. These constraints are related to policies established by the operator of the cloud. Moreover, it is common for the constraints to conflict with each other. Examples of constraint policies are given below:

- Efficient use of electricity (towards a green environment): This policy allocates as many VMs as possible to a host. Electricity can be saved by powering-off idling hosts.
- Availability of a virtual system: This policy distributes VMs to different hosts (redundancy) to guard against VMs going down if a single host fails. This policy may conflict with the goal of saving electricity.
- Affinity: This policy allocates compatible or similar VMs to the same host. For example, network traffic via switches and routers can be reduced by allocating VMs that routinely communicate with each other to the same host. This is a typical situation for multiple VMs owned by a single tenant in a multi-tenant data center.
- Repulsion: VMs that compete for resources should not be placed on the same host. For example, VMs requiring higher network bandwidth should be placed repulsively to avoid network congestion. Firewalls are a typical example.

- Minimum migration cost: This policy determines VMs that should be migrated from one host to another host. Migration costs can be a function of factors memory size, I/O rate, and the number of hops between hosts.

3 Challenges

Realization of Practical Response Time. It is not practical to use a brute-force method to find optimum VM placement because of the large numbers of hosts and VMs in a cloud computing data center. To realize practical computation time, the number of possible combinations must be limited.

Arbitration. As mentioned in the previous section, the various policies established by the cloud computing data center to determine VM placement may conflict with each other; for example, reducing power consumption conflicts with high availability, and conversely, ensuring high availability may increase power consumption. One of the challenges for a cloud administrator is determining how to arbitrate such conflicting constraints. Prioritization could be a solution, but arbitrating the conflict between availability and power consumption is not axiomatic.

Flexibility. Since the requirements and prioritization of policies can differ among data centers, a multi-objective optimization system should allow a data center operator to configure (add and remove) policies to satisfy particular requirements. Moreover, any system for determining VM placement must be sufficiently flexible to respond to changes in economic, political, and social conditions, such as the rising cost of energy, preferential taxation systems for ecological initiatives, regulations requiring the restriction of CO_2 emissions, and other responses to global warming and climate change. For example, data centers in Japan had to respond to restricted electricity supply after the earthquake and tsunami, which occurred in March, 2011.

4 Solutions in the Design

Avoiding Excessive Numbers of Possible Placement Combinations. This is essential for the realization of a practical multi-objective optimization mechanism. We have solved this difficulty by defining and introducing equivalent sets of hosts.

Although a large number of hosts exist in a cloud computing data center, they can be classified into a relatively small number of equivalent sets on the basis of the status of each host. One host from a set can be used to evaluate the cost, thereby significantly reducing the required computations. For example, all hosts can be classified into two equivalent sets (powered on and off) to determine how to apply the electricity-saving policy mentioned in section 2.

Here, we assume that n is the number of hosts and m is the number of VMs to be deployed. When a brute-force method is used, there are n choices for placement of each VM. Therefore, the order of required computation is $O(n^m)$.

We are proposing an algorithm, which will be described in detail in a following section, which utilizes a representative VM from an equivalent set to minimize the number of computations. First, because we need to check the status of each host, for each constraint, the order of computations to classify n hosts into equivalent sets is $O(n)$. It is also $O(n)$ for k constraints. Note that it is reasonable to assume $k << n$. Second, $O(n)$ computations are required to calculate a comprehensive cost for n hosts. Additionally, if all the hosts are neighboring, a maximum of $O(n)$ computations are needed to conduct a neighborhood search. All these computations of $O(n)$ are necessary for m VMs. Therefore, the order of computation is $O(n*m)$.

For example, for 120 hosts and 8 VMs:

$$\begin{cases} O(n^m) = 120\text{^}8\ (\sim 10\text{^}16)\text{: brute-force method} \\ O(n*m) = 120*8\ (\sim 10\text{^}3)\text{: proposed mechanism.} \end{cases}$$

Flexibility and Arbitration. Multi-objective optimization must be flexible and capable of arbitrating conflicting constraints. For example, electricity saving and availability of a virtual system are compatible policies in a multi-tenant environment. Although the latter policy acts to distribute one tenant's VMs on different hosts, the former policy works to cluster several tenants' VMs on one of host. As a result, the number of powered-on hosts can be decreased by sharing hosts among tenants. Considering other optimization objectives, the proposed mechanism programmatically finds an optimum VM placement from a large number of possible combinations.

The structure of the proposed mechanism is illustrated in Figure 1. It consists of following three modules:

1. Cost Evaluation Plug-in Module
   The proposed mechanism has been designed with a plug-in structure to enable data center operators to implement various operational policies. The cost evaluation plug-in module evaluates the cost on the basis of a specific optimization objective function. The exceptional value of this design feature is that is allows a comparison of the impact of various constrains on the basis of the cost.

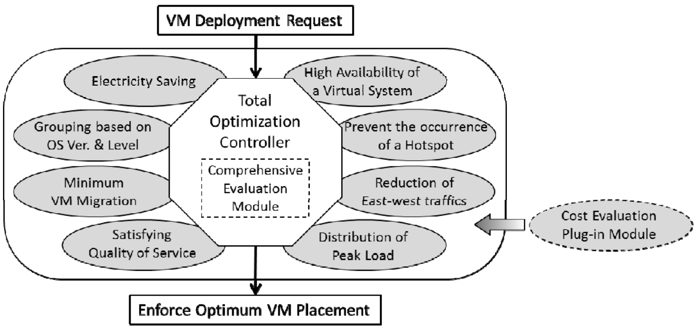

VM Deployment Request
Electricity Saving
High Availability of a Virtual System
Total Optimization Controller
Comprehensive Evaluation Module
Grouping based on OS Ver. & Level
Prevent the occurrence of a Hotspot
Minimum VM Migration
Reduction of East-west traffics
Satisfying Quality of Service
Distribution of Peak Load
Cost Evaluation Plug-in Module
Enforce Optimum VM Placement

Fig. 1. Structure of the proposed mechanism

2. Comprehensive-Evaluation Module
   This module gathers the evaluated cost from all plug-in modules. Subsequently, the comprehensive-evaluation module calculates a comprehensive cost considering the weight rating of each policy. The formula is as follows:

$$C_i = \sum_{j=1}^{k} w_j \cdot e_{i,j}$$

   where C_i is the comprehensive cost for host i, w_j is the weight of the policy j, and $e_{i,j}$ is an evaluated cost for host i by applying policy j.
3. Total Optimization Controller
   The total optimization controller allows the API to accept optimization requests. Interacting with other modules, this controller determines and enforces an optimum VM placement.

An overview of the algorithm is shown in pseudo code in Figure 2. The algorithm works in two phases. In the first phase, a temporal VM placement is determined as an initial state. Subsequently, in the second phase, a neighborhood search is performed.

```
Phase 1: Determine a temporal VM placement
for each VM to be deployed do
  for each Cost Evaluation Plug-in do
    Classify all hosts into equivalent sets based on
    status of each host;
    for each equivalent set of hosts do
      Calculate required additional cost if the VM is
      placed on a host;
    end
  end
  for each host do
    Calculate a comprehensive cost;
  end
  Sort the list of hosts based on comprehensive cost;
  for the head of list to the end of list do
    next unless the host has capacity to host the VM;
    Select the host as the temporal place for the VM;
    break;
  end
end.
```

```
Phase 2: Neighborhood search
for each VM do
  Find neighborhood hosts of the temporal host;
  for each neighborhood host do
    Calculate a comprehensive cost if the VM is placed
    on the host alternative on the temporal host;
  end
  Select the lowest cost host among the temporal and
  neighborhood hosts as an optimum place for the VM;
end.
```

Fig. 2. Overview of the algorithm

5 A Prototype Implementation

Total Optimization Controller / Comprehensive-Evaluation Module. A prototype of the Total Optimization Controller has been developed as a web application. We used Ruby on Rails [6], which is a framework for developing web applications, to realize fast implementation. The controller provides a REST API to accept requests from the Resource Orchestrator, which will be described in greater detail in Section 6. The comprehensive-evaluation module has been implemented as a component module of the total optimization controller.

Cost Evaluation Plug-in Modules. We have implemented plug-in modules for four typical VM placement policies. The plug-in modules are written in Ruby 1.8.

1. Electricity saving
   This plug-in module evaluates the cost of electricity. For simplicity, we focus only on whether a physical server is powered on or off. The actual electricity consumption depends on the load, but there is a significant difference between powered on and off.
   Evaluated cost will be:
   $$\begin{cases} E\text{, if a VM is placed on a host which needs to be powered on} \\ 0\text{, otherwise} \end{cases}$$

2. Availability of a virtual system
   This plug-in module takes particular note of redundancy of VMs in the same tier. An example of a commonly used three-tiered web system is shown in Figure 3. In the example, each tier (web, application, and database) has redundant VMs. However, the whole tier will be downed by the failure of a single host if redundant VMs are deployed on the same host. To prevent such a situation, this module considers the loss of redundancy as a cost.
   Evaluated cost will be:
   $$\begin{cases} F\text{, if a VM is placed with another VM that belongs to the same tier} \\ 0\text{, otherwise} \end{cases}$$

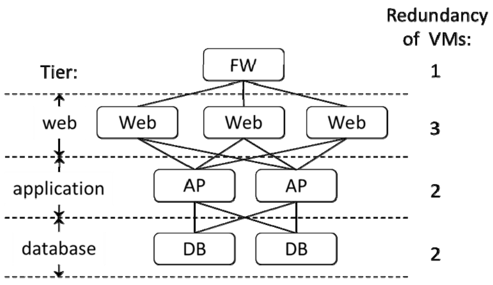


Fig. 3. Example of a 3-tiered web system

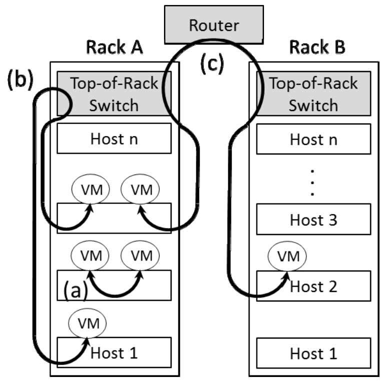


Fig. 4. Example of 3 types of communication situations

3. Affinity (reduction of network traffic beyond a top-of-rack switch)
   This plug-in module evaluates the impact of network traffic. As shown in Figure 4, communication between VMs is classified into three types:

   a. *S* (small): No network traffic is required outside of a host. VMs are placed on the same host.

   b. *M* (medium): Network traffic between hosts via a top-of-rack switch is required when VMs are placed on different hosts exiting in the same rack.

   c. *L* (large): Network traffics with inter-rack routing are required when VMs are placed on the hosts in different racks.

   Additionally, we introduced a *filling rate* to represent rack occupancy. For each rack, the *filling rate* is given as the number of deployed VMs divided by the total capacity (number of possible VMs) of the hosts. Note that the number of VMs is determined by the smallest VM equivalent. When the *filling rate* exceeds the predefined threshold, this plug-in module charges additional cost *A*. Therefore, placement of VMs on racks that are at or close to capacity is discouraged.

4. Repulsion
   We assume that a virtual system contains at least one VM that acts as a firewall. Typically, a firewall requires high bandwidth because all traffic to and from associated networks pass through it. Therefore, placing multiple VM firewalls on the same host is generally undesirable. This plug-in module considers competition for network bandwidth as a cost.
   Evaluated cost will be:
   $$\begin{cases} B^2\text{, if a firewall VM is placed with other firewall VMs} \\ 0\text{, otherwise} \end{cases}$$
   Note: The value of B is proportional to the number of VM firewalls on the host. We use B^2 as an analogy of the charge repulsion.

6 Preliminary Evaluation

We conducted simulations to evaluate the proposed mechanism. Fujitsu ServerView Resource Orchestrator [7] and a hardware simulator were used to construct a mock cloud computing data center environment. The Resource Orchestrator manages all pseudo physical resources (servers, network switches and storage) and VMs. In addition, it manages and allocates addressing resources (MAC addresses, IP addresses, and VLAN IDs). We have made a small modification to the Resource Orchestrator to invoke the proposed mechanism when it receives a request to deploy a virtual system from a user.

Conditions of Simulations

- Physical Servers: 120 homogeneous physical servers; each capable of hosting 20 "economy" VMs (see Table 1).
- Virtual Machines: As shown in Table 1, VMs has been classified into economy, standard, advanced, and high performance. These types were determined by reference to the Amazon EC2 and the Fujitsu Global Cloud Platform.
- Virtual Systems: A virtual system consists of 2-12 VMs that cab be comprised of various types. All VMs in a virtual system will be deployed or deleted synchronously on the basis of a request from a user.
- Duration of a simulation: 1 year.
- Overall CPU utilization: The average overall CPU utilization in a data center starts from 0% (no VM deployed) and grows up to 60% by the end of one year. Note that some deployed virtual systems are deleted during the simulation. In this preliminary evaluation, VM placement that resulted in an over-committed state was not allowed.
- Electricity costs: We assume that a physical server will consume 300 W of electricity (e.g., Fujitsu PRIMERGY RX200S5 equipped with two Intel Xeon 2.53GHz processors and 24GB memory). Based on the cost of the special high-tension voltage power in Tokyo, Japan, the cost is approximately 3.4 yen per hour.

Table 1. Types of VMs

Type	CPU	Memory (GB)
Economy	1	1.7
Standard	2	3.4
Advanced	4	7.5
High Performance	8	15.0

Note: "CPU=1" is equivalent CPU performance of Intel Xeon 1.0 GHz.

Simulation Results. Configurations of the weighted values for the applied plug-in modules are summarized in Table 2. "A" represents the lowest boundary of electricity cost. "F" distributes VMs considering both availability and repulsion (i.e., no electricity saving). The weight of availability and repulsion are gradually increased from "C" to "E."

Three patterns of request sequences, p1, p2, and p3, were assessed. Under conditions described in above, each virtual system was given randomly generated parameters: type and number of VMs, date and time of deployment, and deletion.

Figure 5 shows the average calculation time to find an optimum VM placement by the proposed mechanism and a brute-force method (simulated on a PC equipped with Intel Core i5-2520M 2.50GHz CPU and 4GB memory). The x-axis represents the number of VMs included in a virtual system. We can see that the proposed mechanism realized a practical calculation time even the number of VMs was increased. For example, for 8 VMs, the average calculation time by the proposed mechanism and a brute-force method were 0.097sec and 160sec, respectively.

Figure 6 shows the accumulated costs for one year of simulated operation. Figure 6(a) indicates the real cost of electricity. Figures 6(b) and (c) represent the assigned costs of availability and repulsion, respectively. As mentioned in Section 5, assigned costs are charged when constraints are not satisfied; i.e., smaller value is preferable.

Table 2. Configurations of weight for the applied plug-in modules

config.	electricity saving	availability	repulsion	affinity
A	1.0	-	-	-
B	1.0	2.0	-	-
C	1.0	0.5	0.5	1.0
D	1.0	1.0	1.0	1.0
E	1.0	2.0	2.0	1.0
F	-	1.0	1.0	-

(- : not applied)

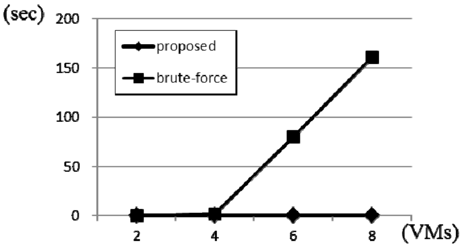
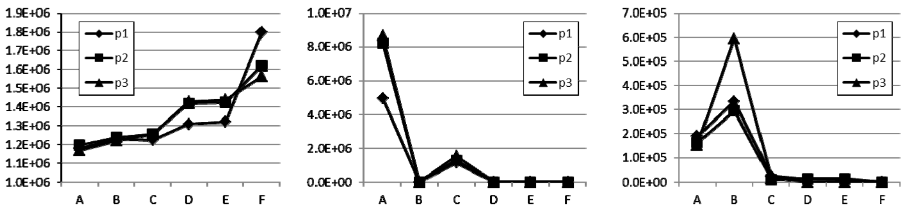


Fig. 5. Average calculation time to find an optimum VM placement



(a) electricity (b) availability (c) repulsion

Fig. 6. Accumulated costs (duration: 1 year)

The simulation results show that the electricity cost increases as other constraints are satisfied. In this simulation, however, both availability and repulsion were well satisfied for configurations “D” and “E.” Compared with “F,” 8.4% to 27.3% of electricity cost was saved.

7 Related Work

Ni et al. [8] implemented a probabilistic scheme to determine VM placement. A roulette wheel is used in their scheme. A sector on the roulette wheel corresponds to a host. To create larger selection probability, a larger central angle was assigned to a sector associated with a host that has larger amount of available resources. In the proposed VM mapping policy, multi-dimensional resource usage (e.g., CPU and memory) was considered. However, other constraints, such as electricity saving and high availability of a virtual system, were not considered.

Xu et al. [9] and Garces et al. [10] implemented multi-objective optimization mechanisms for VM placement and migration. Their common approach applies a genetic algorithm to solve a multi-objective optimization problem. Our approach does not use a genetic algorithm. As mentioned in Section 4, we introduced equivalent sets of hosts to avoid extremely large numbers of possible placement combinations.

Tsakalozos et al. [11] proposed an approach similar to ours; i.e., identifying potentially compatible groups of physical servers to reduce the search space. Moreover, a few constraints such as power saving and minimizing network traffic by co-deploying a set of VMs on the same physical server were considered. In this research, however, physical servers were classified into groups based solely on VM migration ability because the goal was load balancing through migration. In contrast, our mechanism generates equivalent sets of physical servers (hosts) for each constraint or policy. Note that the VM migration ability of a host can be added as a constraint by implementing a plug-in module. Subsequently, our mechanism places a VM on a host that was evaluated as having the lowest comprehensive cost because our goal is the reduction of operational cost and not load balancing.

8 Conclusion

We have designed and implemented a multi-objective optimization mechanism for VM placement. The proposed mechanism is flexible and allows data center operators to add their own desired optimization objectives or evaluate specific policies by implementing plug-in modules.

The unique value of our system is that a constraint is translated into an estimated cost (real or assigned). Each plug-in module evaluates the additional cost that would accrue if a VM is placed on a host. Subsequently, the Comprehensive-Evaluation Module gathers the results and calculates the total cost considering the weight of each policy. The Total optimization controller conducts a neighborhood search to find the lowest cost VM placement, and ultimately, it enforces the optimum VM placement.

A practical calculation time to find an optimum VM placement was realized by introducing the equivalent sets of hosts. The simulation results showed that both availability and repulsion could be satisfied with a cost saving of 8.4%-27.3% for electricity.

Acknowledgments. We would especially like to express our heart-felt gratitude to Messrs. Toru Sakiyama, Noriyuki Murata, and Makoto Nasuno who are systems software engineers at ASCADE Inc.

References

1. Amazon Elastic Compute Cloud (EC2), `http://aws.amazon.com/ec2/`
2. Amazon Simple Storage Service (S3), `http://aws.amazon.com/s3/`
3. Google App Engine, `http://code.google.com/appengine/`
4. Saleseforce CRM, `http://www.salesforce.com/crm/`
5. Fujitsu Global Cloud Platform, `http://www.fujitsu.com/global/solutions/cloud/solutions/global-cloud-platform/`
6. Ruby on Rails, `http://rubyonrails.org/`
7. Fujitsu ServerView Resource Orchestrator Cloud Edition, `http://www.fujitsu.com/fts/products/computing/servers/primergy/management/dynamize/ror-ce/ror-ce.html`
8. Ni, J., Huang, Y., Luan, Z., Zhang, J., Qian, D.: Virtual Machine Mapping Policy Based on Load Balancing in Private Cloud Environment. In: Proc. of 2011 Int'l Conf. on Cloud and Service Computing, Hong Kong, China (2011)
9. Xu, J., Fortes, J.A.B.: Multi-objective Virtual Machine Placement in Virtualized Data Center Environments. In: Proc. of 2010 IEEE/ACM Int'l Conf. on Green Computing and Communications, Hangzhou, China (2010)
10. Garces, N., Ortiz, N., Mendez, D., Donoso, Y.: Multi-Objective Optimization for Virtual Machine Migration on LANs for Opportunistic Grid Infrastructure. In: Proc. of the 3rd Int'l Conf. on Emerging Network Intelligence, Lisbon, Portugal (2011)
11. Tsakalozos, K., Roussopoulos, M., Delis, A.: VM Placement in non-Homogeneous IaaS-Clouds. In: Proc. of the 9th Int'l Conf. on Service Oriented Computing, Paphos, Cyprus (2012)