A Multi-objective Virtual Machine Migration Policy in Cloud Systems

Ahmed Sallam1,2 and Kenli Li1,*

1The College of Information Science and Engineering, Hunan University & the National Supercomputing Center in Changsha, Hunan, P.R. China
2The Faculty of Computers and Informatics, Suez Canal University, Ismailia, Egypt
*Corresponding author: lkl@hnu.edu.cn

In a Cloud computing environment, the workload fluctuates dynamically producing undesirable situations such as load imbalance, lower utilization or workload hotspots. In such cases, virtual machine migration is a potential solution. However, an algorithm based on a single objective (e.g. service-level agreement) is usually used to direct the migration process. On the contrary, there exist unconsidered conflicting factors impacting the migration process such as load volume, power consumption and resource wastage. In this paper, we consider the migration process as a multi-objective problem where the objectives are typically non-commensurable. Therefore, we propose a novel migration policy consolidated by a new elastic multi-objective optimization strategy to evaluate different objectives (including migration cost) simultaneously, and to provide the flexibility for manipulating different cases. We have tested the proposed policy through an extensive set of simulation experiments using CloudSim, and the results ensure the efficiency of our policy to control the system performance by adjusting the migration objectives to suit various workload situations.

Keywords: cloud computing; virtualization; multi-objective optimization; CloudSim

Received 28 July 2012; revised 25 January 2013
Handling editor: Jongsung Kim

1. INTRODUCTION

Recently, customers and providers have intensively directed their business to cloud systems because of the availability and the scalability, in addition to the promising cost savings which is the most attractive advantage, since you only pay for what you actually use. This owes everything to Virtualization, not to the big data centers, the use of web services nor the automated ability to balance workloads.

Virtualization in its simplest form is the process of taking a physical machine and subdividing it through software into the equivalent of several discrete machines [1]. Although they share the resources of a single server, they work independently from each other without conflicts, and as a result, reduce the amount of hardware in use, improve the utilization of resources, improve the fault and performance isolation between applications sharing the same resources, ease moving virtual machines (VMs) from one physical host to another using live or off-line migration, and support hardware and software heterogeneity [2].

Primal cloud providers such as Amazon, Sun Microsystems, Google, Salesforce, Microsoft and IBM have established powerful cloud datacenters for hosting real-world cloud computing application services such as social networks, gaming portals, business applications, media content delivery and scientific workflows. Although the high technology used with datacenter equipment, such applications serve millions of users all over the world and are periodically updated with new features and plug-ins. Consequently, the workload fluctuates rapidly and produces undesirable situations such as load imbalance when some of the cloud servers are overloaded while others are underloaded because the tasks being processed are completed, or when the resources remaining on each server prevent any further VM placement. Another case occurs when hotspots take place in several situations because the aggregate usage of any resource (processor, network bandwidth or memory) exceeds the predefined thresholds. Moreover, with time customers may request to scale up the assigned service, but it is not possible as the hosting server has reached the maximum load.

Whenever this happens, migration is used to relax the workload by moving the virtual machine from one server to another to operate the migrated VM. Furthermore, it can be
used to free up unloaded servers to save the consumed power as proposed in many researches [3–7].

Practically, the migration process usually considers predefined thresholds in datacenters or a set of service-level agreement (SLA) terms for the running applications and tries to handle the problem with a greedy algorithm to determine the sequence of moves or swaps to migrate VMs from overloaded to underloaded datacenters until some predefined thresholds are achieved. In detail, the VMs are ordered based on their workload volume, and also the servers are ordered based on their load status. The algorithm then proceeds by considering the highest virtual machine’s workload from the highest loaded server and determines if it can be housed on the least loaded server [8].

Indeed, migration is a direct solution for the workload issues; however, there are some other important factors to be considered in the migration process such as utilization of resources which can be controlled by the resource wastage (RW) to prevent imbalanced VM placement, and migration cost (MC) which basically depends on the size of the migrated VMs and the transfer rate [9, 10]. Another important factor is the power consumption (PC), because there are other crucial problems that arise from high PC [4, 6, 9]. For example, insufficient or malfunctioning cooling system can lead to overheating of resources, reducing system reliability and reducing the lifetime of devices [4].

The rest of this paper is organized as follows: In Section 2, we review the related works and illustrate our contribution. In Section 3, we describe the multi-objective genetic algorithm used in this work. In Section 4, we introduce the overall idea of the migration policy, and in Section 5 we explain in detail the multi-objective evaluation model used. Section 6 describes the implementation and the applicability of the proposed policy followed by the experiments and the results analysis in Section 7. Finally, we conclude this work and mention the future work plan in Section 8.

2. CONTRIBUTION AND RELATED WORKS

Many researches exploit the virtual machine migration to improve the efficiency for only a single objective such as scaling resources, saving PC or increasing the utilization of resources [11, 12]. On the contrary, a few research studies introduce direct approaches to manipulate more than one objective together; however, there still exist some drawbacks in their works.

In [8], the authors adopted a black-box technique to monitor the system resource usage, detect hotspots and initiate the necessary migrations, and gray-box technique to access a small amount of OS level statistics to better inform the migration algorithm. The drawback of this work comes from the migration phase where the VM with the highest volume/memory size can be housed on the least loaded volume server with sufficient resources, ignoring other objectives such as thermal status, PC and RW.

Another system was proposed in [4], where the authors introduced an energy-efficient resource management model followed by several stages of VM placement optimization to reduce operational costs and to provide the required quality of service. The system depends on three components: dispatcher to respond to new requests and to dispatch VMs to a global manager which controls a set of nodes. Then the global manager starts to distribute and allocate VMs using a semi-online multidimensional bin-packing algorithm based on information received from a local manager component attached to each node. The system then follows several reallocation policies to optimize power efficiency and to guarantee the SLA; these policies are triggered by upper and lower utilization thresholds for CPU utilization, network communication and temperature.

The main drawback of this system is that it performs the reallocation policies as separated phases where the improvement gained from the first phase could be wasted by VMs migration in the following phases. Moreover, the system does not consider the MC. With these drawbacks, the overhead that results from the migrated VMs degrades the proposed system efficiency and makes it inapplicable, especially on a large-scale cloud datacenters.

On the other hand, the authors of [12] created a model for VMs allocation using the min-max and shares features to evaluate resource usage, PC and application utilization together. Although this technique is simple, it does not provide the flexibility to suit different workload situations.

In this paper, we introduce a novel multi-objective VM migration policy to optimize possibly conflicting objectives including LV, PC, TS, RW and MC simultaneously. The contribution of this work is summarized as follows:

(i) To the best of our knowledge, this is the first work to consider both the dynamic fluctuation of the workload in a cloud environment, especially on large-scale cloud datacenters, and the migration cost; thus, the model can minimize the migration overhead which makes it applicable in practice.

(ii) The new multi-objective optimization strategy is relied on our efficient Static Bayesian Game based Multi-Objective Genetic Algorithm (SBG-MOGA), and completed by the Augmented Weighted Tchebycheff Program (AWTP) to provide a flexible way to adjust the migration processes selection to different objectives. This helps to find a proper migration decision to different workload situations.

(iii) We have investigated the efficiency of VMs migration based on different single objective, and then verified the impact of using multi-objective evaluation to find an optimal solution and to control this solution to suit different situations.
A Multi-objective Virtual Machine Migration Policy in Cloud Systems

3. STATIC BAYESIAN GAME BASED MULTI-OBJECTIVE GENETIC ALGORITHM

Most problems in the real world have more than one conflicting objective that need to be optimized simultaneously. During the past two decades a large number of multi-objective optimization algorithms have been proposed to solve multi-objective problems. However, through our continuous research in this field we detected several challenges with such algorithms stated in detail in a previous work [13], where we proposed a novel multi-objective genetic algorithm to overcome these challenges based on SBG.

In brief, most of the previous conventional MOGAs use non-dominated sorting methods to push the population to move toward the real Pareto front. This approach had a good performance at earlier stages of the evolution but it has become hypodynamic at the later stages. On the other hand, Static Bayesian Game first introduced by Harsanyi [14] to transform the game of incomplete information (for example, an employer ignores how productive the worker is, a second-hand car is good or not, etc.) as a game of augmented information with the introduction of a new player.

Thus, we proposed SBG-MOGA, which adds static Bayesian game strategy into MOGA to decide how a new population is created. In other words, besides the force generated by the non-dominated sorting, there will be a tensile force generated by the running game between objectives. These two kinds of forces together act on the population, which will obtain a better performance in terms of finding a diverse set of solutions and drawing near the true Pareto-optimal set.

Typically, in SBG-MOGA each generation of evolution is considered as a game and each objective to be optimized is considered as a player. A player is clever enough to know how to select an appropriate strategy to get the biggest income in the game. The algorithm contains a sequence of rounds; in each round players play with each other, which will provide the population with tensile force to pull them to move toward the true Pareto front. In addition, the algorithm adopts the elitism mechanism (see Algorithm 1).

**Algorithm 1** SBG-MOGA.

1: Initialize \( B(0) \) randomly
2: \( A(0) = M_f(B(0), \prec) \)
3: \( t = 0 \)
4: repeat
5: \( \text{generate } B(t + 1) \text{ from } B(t) \)
6: \( A(t + 1) = M_f(B(t + 1) \cup A(t), \prec) \)
7: \( t = t + 1 \)
8: until stop condition satisfied

(iv) We have evaluated the proposed model through extensive set of experiments using CloudSim kit.

Here \( B(0) \) is the initial population set, \( M_f \) is the fitness matrix, \( t \) is the generation and \( A \) is the elitism archive to record all the non-dominated\(^1\) solutions.

Let \( F \) be a set of objective space; the non-dominated set in \( F \) is denoted as \( M(F, \prec) \). Also Let \( X \) be the finite search space and \( f : X \rightarrow F \) be a mapping from \( X \) to \( F \) for some \( A \subseteq X \); then

\[
M_f(A, \prec) = \{a \in A : f(a) \in M(f(a)), \prec\}
\]

represents the non-dominated individuals in \( A \).

It is worth mentioning that, in our situation, all the running VMs at the problem side will be considered to initialize the population set to represent the decision space of the migration problem.

The reason behind choosing SBG-MOGA is that the algorithm is exactly fitting the dynamic nature of the defined problem in this work. In addition, we have verified through six benchmark functions in [13] that the algorithm outperforms other popular algorithms to solve multi-objective optimization problems.

4. MODEL DESCRIPTION

We consider all VMs which are hosted by the source node and suffer from one of the workload situations (e.g. workload hotspots, or overheat) as part of the set of possible migration processes. In our model, the decision to select one or a subset of these VMs to be migrated to other destination nodes depends on two phases of evaluations.

(i) Phase I: To find an optimal set of non-dominated alternatives among the available VMs based on SBG-MOGA described in Section 3, and then rank VMs in the optimal set using the AWTP formula to select VMs with best ranks to satisfy the following objectives simultaneously: (1) LV, (2) PC, (3) TS, (4) RW and (5) MC.

(ii) Phase II: To evaluate the main targets behind the migration. In connection with this, we consider two targets, the CPU and the memory usage, which reflect resource usage efficiency, power efficiency and temperature efficiency at the source node.

If the migration of the selected VM has met the targets, the procedure terminates; otherwise, the procedure includes the selected VM in the migration list and updates the VMs’ ranks to select another one repeatedly until the targets are met as depicted in Fig. 1.

---

\(^1\)Suppose we have \( K \) objectives, if all objective functions are for minimization, a feasible solution \( x \) is said to dominate another feasible solution \( y \) \( (x \succ y) \), if and only if \( z_i(x) \leq z_i(y) \) for \( i = 1, \ldots, k \) and \( z_j(x) < z_j(y) \) for at least one objective function \( j \).
In the following subsections, we will describe the proposed model mathematically. For convenience, we have summarized all the important abbreviations in Table 1.

### 4.1. Objectives formulation

In this model, we consider five objectives to evaluate the migration of virtual machine $v_i$ to destination $d_j$.

**Load volume (LV):** Since a VM or a server can be overloaded along one or more dimensions such as CPU, network and memory, we will reuse the metric described in [8] that captures the combined CPU–network–memory load of a virtual and physical server.

Let $U_{CPU}$ be the CPU utilization, $U_{mem}$ be the memory utilization and $U_{net}$ be the network utilization. The LV of a physical or virtual server is defined as the product of its CPU, network and memory loads such that

$$LV = \frac{1}{1 - U_{CPU}} \cdot \frac{1}{1 - U_{mem}} \cdot \frac{1}{1 - U_{net}}.$$  \hspace{1cm} (1)

**Power consumption (PC):** The operational PC for a server node comprises constant energy consumption $E_c$, which represents the constant power consumption $P_c$ of all components except the CPU in proportion to the operational time, and dynamic energy consumption $E_d$ of the executed applications, which depends mainly on the dynamic power $P_d$ of the CPU [15] given by

$$PC = E_c + E_d.$$  \hspace{1cm} (2)

The dynamic power $P_d$ of the processor is usually in proportion to processor frequency $f$ given by

$$P_d = C \cdot f^3,$$  \hspace{1cm} (3)

where $C$ is a coefficient. Accordingly, if we consider to execute an application with execution time $t$, and if the processor runs at frequency $f$ such that $0 < f < f_{max}$, the execution time is defined by $t/(f/f_{max})$. Thus the dynamic energy consumption $E_d$ for this application is given by

$$E_d = \alpha \cdot t \cdot S^2,$$  \hspace{1cm} (4)

where $\alpha$ is the proportional coefficient and $S$ is the associated processor speed related to the frequency $f (S = f/f_{max})$. From Equation (3), the power consumption for the server node is affected directly by the LV described above. However, even the LV of two nodes are equal, the power consumption of these nodes can vary with respect to the physical device capabilities for each node as shown in Equation (2).

**Thermal state (TS):** The TS for a server is proportional to the operational power of the server and the ambient temperature $T_{amb}$ given by

$$T = P \cdot R + T_{amb},$$  \hspace{1cm} (5)

where $R$ is the thermal resistance. We already considered the power consumption as an objective; however, we cannot ignore the TS as it comprises $T_{amb}$ as another factor.

**Resource wastage (RE):** This can be estimated based on the waste residual resource notation RW explained in [16]. This notation depends on balancing the resource usage with...
respect to smallest normalized residual resource which is formulated as

\[ \text{RW} = \sum_{i,j,k} (\text{NR}_i - \text{NR}_k), \]  

(6)

where NR denotes the normalized residual (the ratio of the residual resource to the total resource), \( K \) to identify the dimension that has the smallest normalized residual capacity and \( i \) for other dimensions. For example, if we consider three dimensions (i.e. CPU, memory and network), RW can be calculated by \( \text{RW} = (U_{\text{mem}} - U_{\text{CPU}}) + (U_{\text{mem}} - U_{\text{net}}) \) where \( U_{\text{mem}} \) represent \( \text{NR}_k \) in this situation.

**Migration cost (MC):** Migration cost (overhead) may vary significantly for different workloads due to the variety of VM configurations and workload characteristics. In brief, the performance of VM migration is affected by many factors, mainly the size of the VM memory, the network transmission rate and the energy consumption due to migration, especially in a large-scale system. In addition, live migration has more additional factors such as the memory dirtying rate, which reflects the memory access patterns of different applications and impacts the number of pre-copying rounds to complete the VM migration [9, 17]. All the MC metrics for a VM migration are integrated in the following formula:

\[ \text{MC} = aV_{\text{mig}} + bT_{\text{mig}} + cE_{\text{mig}} + [dT_{\text{down}}], \]  

(7)

where \( a + b + c + d = 1 \); \( a, b, c \) and \( d \) are the weights of cost metrics, \( T_{\text{mig}} \) is the duration of migration, \( V_{\text{mig}} \) is the total network traffic of the migration process, \( E_{\text{mig}} \) is total energy consumed by the migration process and \( T_{\text{down}} \) is the downtime caused in the migration process. Considering the offline migration only, the total network traffic \( V_{\text{mig}} \) equals the memory size of the migrated virtual machine \( V_{\text{mem}} \).

The migration duration \( T_{\text{mig}} \) is calculated by

\[ T_{\text{mig}} = \frac{V_{\text{mem}}}{M_{\text{TR}}}, \]  

(8)

where \( M_{\text{TR}} \) is the memory transmission rate.

The total energy \( E_{\text{mig}} \) can be calculated by

\[ E_{\text{mig}} = E_{\text{sour}} + E_{\text{dest}} + [E_{\text{net}}], \]  

(9)

where \( E_{\text{sour}} \), \( E_{\text{dest}} \) and \( E_{\text{net}} \) represent the additional energy consumed by the source, destination and network switches, respectively. The energy consumed by network switches \( E_{\text{net}} \) is complex to identify; thus, here we consider only the energy consumed by the source \( E_{\text{sour}} \) and the destination \( E_{\text{dest}} \), which can be calculated by identifying the frequency consumed for copying/moving 1 MB in each device which varies with respect to the device configuration. Next, by using Equation (3) we can estimate the additional power. Suppose, for example, the power consumed to copy 1 MB in a source node equals \( P_{\text{sour}} \), then

\[ E_{\text{sour}} = V_{\text{mem}} \cdot P_{\text{sour}}. \]

### 4.2. Targets estimation

The estimation of targeted CPU usage \( T_{\text{CPU}} \), generally depends on the workload situation of the migration’s source node (e.g. load imbalance, lower utilization, workload hotspot or scaling necessity). To be more specific, \( T_{\text{CPU}} \) can be estimated in terms of the processor frequency following two situations:

(i) Situation I: When the CPU usage exceeds the predefined thresholds; in this case \( f \) can be specified directly to revert to the CPU usage thresholds.

(ii) Situation II: To reduce the power consumption or to lower the thermal state (TS) of the running server, since the dynamic power \( P_d \) is proportional to the frequency \( f \) as shown in Equation (3). Thus, the desired power consumption limit can be determined in terms of \( f \) to express \( T_{\text{CPU}} \). This assumption is also valid in order to reach the desired TS as the temperature \( T \) is directly affected by the circuit power [18, 19] according to Equation (5).

On the other hand, the estimation of targeted memory usage \( T_{\text{mem}} \) also depends on two situations:

(i) Situation I: When the memory usage reaches the maximum thresholds of the available memory, in the same way as \( T_{\text{CPU}} \), the \( T_{\text{mem}} \) can be estimated directly to return to the predefined memory usage thresholds.

(ii) Situation II: To reduce the RW. In this situation \( T_{\text{mem}} \) can be estimated by adjusting the amount of memory to minimize the RW following Equation (6).

### 5. MULTI-OBJECTIVE EVALUATION

Let set \( V \) contain all virtual machines \( v_i \) hosted by the migration’s source node and let set \( D \) contain all the destinations \( d_j \) available to host the migrated virtual machines, and fulfilling the following constraints \( C \):

\[ d^\text{CPU}_j < c^\text{CPU}_j, d^\text{mem}_j < c^\text{mem}_j. \]

These constraints are used to filter the destinations such that the memory and CPU capacity of the selected destination should not exceed the maximum thresholds.

Moreover, let set \( X \) include possible migration processes of a virtual machine \( v_i \in V \) to a destination \( d_j \in D \). For simplicity we express \( X \) as a matrix where \( v_i \) represent the rows and \( d_j \) represent the columns as follows:

\[
\begin{bmatrix}
  x_{11} & \cdots & x_{1n} \\
  \vdots & \ddots & \vdots \\
  x_{m1} & \cdots & x_{mn}
\end{bmatrix}
\]
The goal is to find the most desirable migration process \( x_{ij} \in X \), in other words, the migration process with the highest membership toward all the objective functions \( f_k = 1, 5 \in \{LV, PC, TS, RW, MC\} \). This can be translated to: find \( x_{ij} \) with minimum \( (f_{LV}, f_{PC}, f_{TS}, f_{RW}, f_{MC}) \) simultaneously.

To evaluate \( x_{ij} \), first we use the SBG-MOGA algorithm shown in Section 3, where the result of this step is an optimal set, denoted here as \( X \); then we adopt the AWTP method proposed by Steuer [20] to find only a single migration process from \( X \), given by

\[
\min \max_k \left[ w_k (f_k(x_{ij}) - u_k) \right] + \rho \sum_{k=1}^{m} (f_k(x_{ij}) - u_k)
\]

s.t. \( x \in \tilde{X} \),

(10)

where \( w_k \) is a predefined weight for objective \( k \), \( u_k \) denotes the utopia point (optimum value for objective function \( f_k \)), \( m \) is the number of objectives and \( \rho \) is a sufficiently small positive scalar. According to Steuer, \( \rho \) can affect the outcomes of the equation such that a small \( \rho \) can provide weakly non-dominated solution and a big \( \rho \) can lead to missing non-dominated outcomes. Thus, he suggested \( \rho \) to be in the range \( [10^{-2}, 10^{-4}] \). Many research studies later on introduced practical solution to choose the value of \( \rho \); however, in our model AWTP is used to rank the proposed migration of VMs, so a larger value for \( \rho \) will be effective to choose the best ranked migration process. Therefore, we choose \( \rho = 10^{-2} \) as Steuer suggested.

Reasons behind choosing AWTP are: AWTP formula includes utopia point \( u_k \), which gives us the ability to adjust the solution to desirable objectives for each destination. This transforms the problem from one of choosing the best solution blindly that of to selecting the nearest option to the desired solution. Moreover, the formula also facilitates the process of assigning different weights for objects with respect to their importance. In other words, if the energy efficiency is more essential than the MC, the user can drive the AWTP to take side with the energy efficiency by assigning a higher weight.

In more details, to select a single migration process with AWTP from the optimal set, we calculate the functions \( f_{LV}, f_{PC}, f_{TS}, f_{RW}, f_{MC} \) with the assumption that virtual machine \( v_i \) has been migrated to destination \( d_j \). Also, we assume that there is a set of desired thresholds \( (u_{LV}, u_{PC}, u_{TS}, u_{RW}, u_{MC}) \) defined for the destination \( d_j \). Based on the first part of the AWTP equation

\[
\max_{k=1,...,m} \left[ w_k (f_k(x_{ij}) - u_k) \right],
\]

we can determine the most effective objective which is as close as possible to the desired threshold.

We can stop at this level by applying this part on every \( x_{ij} \in X \) and selecting \( x_{ij} \) with the minimum value as a solution. Formally, this is called the Weighted Tchebycheff method; however, the solution based on this method is weakly Pareto optimal [21].

To avoid the weakly Pareto-optimal solution, we proceed with the calculation by adding the second part

\[
\rho \sum_{k=1}^{m} (f_k(x_{ij}) - u_k)
\]

to ensure that the result is Pareto Optimal and to consider all the objectives simultaneously. The whole procedure is encoded in Algorithm 2.

**Algorithm 2** Multi-objective Virtual Machine migration policy.

1:Main
2: migration list \( L = \{} \)
3: migration process \( x_{ij} = \) null
4: Calculate targets \( T_{CPU}, T_{mem} \) \( //\) see Section 4.2
5: Construct matrix \( X \) \( //\) see Section 5
6: \( X = \) SBG-MOGA(\( X \))
7: While (True)
8: \( x_{ij} = \) AWTP(\( X \))
9: Add(\( L, x_{ij} \))
10: If (\( T_{CPU} \) and \( T_{mem} \)) met break
11: else
12: \( X, d_{i,j} \)
13: EndWhile
14: return \( L \)
15:EndMain
16:Update( \( X, d_{i,j} \)
17: remove row(\( i \))
18: update column(\( j \)) \( //\) update destinations \( j \) with
19: EndUpdate
20:EndUpdate

\( //\) respect to the migrated VM \( i \)

### 6. Model Implementation

In this work, our concern focuses on developing a migration policy rather than developing a system to manage virtual machines placement.

Generally, the physical cloud system is composed of \( n \) heterogeneous physical nodes. Each node is characterized by CPU (which can be multi-core), amount of RAM and network bandwidth. In addition, another component called Virtual Machine monitor is responsible for logging the node’s resources and TS. Here, we introduce a new component called the migration manager (Mman). Whenever there is a need for a migration process, the Mman will perform Algorithm 2.

It is worth mentioning that, in some situations, the customer may ask to scale up the assigned service; however, it is not possible to resize the VM that operates his applications because the node hosting this service reaches the maximum thresholds. Logically, we can suggest to migrate the defined VM to another free node which makes the Mman’s first step unnecessary. However, this is not an adequate solution because we can choose...
to migrate another VM that runs on the same node to decrease the migration overhead.

6.1. Algorithm analysis and applicability

The influencing factor in the running time of Algorithm 2 is ‘SBG-MOGA’ (line 6). This function represents a MOGA, and the running time at the worst case is \( O(mN^2) \) [22, 23], where \( N \) is the population size (number of VMs) and \( m \) is the number of objectives.

Practically, this running time is insignificant if one considers that the current technology available supports at most 512 virtual machines per host (see Table 2). Thus, if we consider the worst case of Algorithm 2 to migrate 512 VMs, the running time is still insignificant. Even the increase of these numbers in the future will stay limited with the host configuration. In other words, the fundamental factor \( N \) is a predefined limited constant; consequently, the running time is insignificant.

7. EXPERIMENTS AND RESULTS EVALUATION

We have performed extensive simulation experiments using the CloudSim toolkit [24]. The reasons behind choosing to work with this simulator are: CloudSim supports most of cloud architecture components such as data centers, virtual machines; moreover, it supports custom Java interfaces that can be simply extended. In addition, and most importantly, CloudSim provides the possibility of monitoring and providing information (e.g. utilization, power consumption) for different components such as datacenters.

The experiments were conducted on a Pentium Dual-Core CPU E6700 machine with 3.19 GHz, 2 MB of L2 cache and 2 GB of memory running Windows XP SP3 and JDK 1.6.

The test simulation environment setup includes ‘PowerDataCenter’ which is a CloudSim datacenter with additional methods to monitor the power consumption, and ‘DataCenterBroker’ to dispatch cloudlets over the datacenter hosts/servers. The number of hosts in the simulation \( \sim 60000 \) servers; every server has different configuration for memory size and CPU speed. Moreover, every server operates a different number of Xen VMs up to 32 with respect to the installation guide in [25].

The goals of the experiments are to test the impact of selecting the migration processes based on different individual objectives and compare the results with the selection based on multi-objective evaluation, and to exhibit the efficiency of our migration policy for adjusting the system performance to desired levels.

To complete the simulation, we have generated a set of migrated VMs up to \( \sim 600000 \) GB. The migration processes were evaluated on the bases of individual different single objective such as power consumption, LV, TS, RW and MC. Next, we have evaluated the migration processes based on a multi-objective policy (MOP), and then recorded the statistics for each case in Table 3. For example, the illustrated statistics in the first row reflects the after migration status of the destination hosts which have been selected as the hosts with the minimum expected LV among other hosts. Furthermore, to clarify the impact of migration with respect to every objective, we generate Fig. 2 to represent the ratio between each row and the totals.

At first glance, Fig. 2 shows that the most effective evaluation objectives are RW, LV and MOP, respectively; however, this observation is tricky because criteria such as the LV depends on Memory, Network and CPU utilization (see Equation (1)).

\[
\text{LV} = \frac{\text{Memory} + \text{Network} + \text{CPU}}{3}
\]

<table>
<thead>
<tr>
<th>Selection criteria</th>
<th>Number of hosts</th>
<th>Total load volume</th>
<th>Total resource wastage</th>
<th>Total power consumption (Watts)</th>
<th>Total increase in temperatures (C)</th>
<th>Total migration cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>LV</td>
<td>5000.00</td>
<td>17 989.14</td>
<td>1858.06</td>
<td>5 211 073.17</td>
<td>15 635.99</td>
<td>624 295.99</td>
</tr>
<tr>
<td>RW</td>
<td>3198.00</td>
<td>40 207.05</td>
<td>767.25</td>
<td>3 629 017.03</td>
<td>9989.49</td>
<td>632 450.89</td>
</tr>
<tr>
<td>PC</td>
<td>15 287.00</td>
<td>256 876.82</td>
<td>8960.09</td>
<td>10 000 658.07</td>
<td>45 975.13</td>
<td>700 556.92</td>
</tr>
<tr>
<td>TS</td>
<td>5180.00</td>
<td>101 120.53</td>
<td>2867.74</td>
<td>5 080 392.77</td>
<td>15 992.63</td>
<td>633 496.67</td>
</tr>
<tr>
<td>MC</td>
<td>18 237.00</td>
<td>300 986.31</td>
<td>11 245.50</td>
<td>14 116 493.66</td>
<td>55 304.90</td>
<td>672 637.74</td>
</tr>
<tr>
<td>AWTB</td>
<td>5704.00</td>
<td>36 259.65</td>
<td>2908.56</td>
<td>5 471 827.03</td>
<td>17 665.23</td>
<td>626 997.50</td>
</tr>
<tr>
<td>Totals</td>
<td>52 606.00</td>
<td>753 439.51</td>
<td>28 607.22</td>
<td>43 509 461.73</td>
<td>160 563.37</td>
<td>3 890 407.70</td>
</tr>
</tbody>
</table>

The results from Table 3 show that there is a significant difference in the total load volume, total resource wastage, and total power consumption. The LV, RW, and MOP objectives have the highest values, followed by the PC, TS, and MC objectives. The AWTB objective has the lowest values.

The Computer Journal, 2013
while the RW depends on the differences between the usage of Memory, Network and CPU (see Equation (6)).

Thus, minimum differences between RW dimensions may lead to an undesirable increase in the utilization of one of these dimensions and increase the LV as well.

On the other hand, the statistics resulting from the migration based on our model do not seem to have the most effective impact on the statistics of the selected destination servers. This can be clarified if we reveal that we set the utopia/desired utilization for memory usage to 75% during the calculation of MOP values to decrease the RW of the whole system instead of reducing the RW beyond the selected destination, as explained before in Section 1, that is to avoid selecting hosts with low memory utilization which can alternatively be freed up to terminate the running host and save more power.

To show the significance of our model, we subtract the rows in Table 3 from the utopia values expected from the selected destination servers for each criterion as shown in Table 4. Meanwhile, we generate Fig. 3 to simplify this illustration as well, by measuring the ratio of each row with respect to the totals. The figure shows that the model based on our algorithm generates the minimum distance from predefined desired values for the selected destination with respect to each objective. In other words, our proposed model gives remarkable control for selecting the most desired migration processes considering all the objective criteria simultaneously.

In addition, to inspect the impact of the migrated amount on the results, we apply the experiments to different amounts of migrated VMs, as shown in Fig. 4. Regarding the post-migration statistics, the selection based on some criteria such as LV, RW and MOP records a minor dominate increase in the statistics with respect to the migration amount. On the contrary, the PC and MC record a minor decrease. This illustration is depicted in the left column of Fig. 4.

8. CONCLUSION

Virtual machine migration is a potential solution for many critical situations in a Cloud environment such as load imbalance, lower utilization and workload hotspots resulting from the dynamic fluctuation of the system workload. However,
the selection of migrated VMs and destinations is usually based on a single objective such as power consumption.

Practically, considering only single objective can contradict with other objectives and may lead to the loss of possible optimal solutions to handle the existing situation. Thus, in this work we introduce a multi-objective evaluation policy to evaluate the migration process based on a set of objectives simultaneously.

Hence, we investigate the impact of single and multi-objective evaluation on the migration process through an intensive set of simulation experiments using CloudSim. The results confirm the efficiency of our proposed policy to control the system performance by bounding the objectives to desired values for each criterion.

Our future plan is to extend the experiments in two directions: First, to involve Live Migration parameters such as the memory dirtying rate. Secondly, to increase the policy running time efficiency by applying different methodologies to evaluate a bulk of migrated VMs instead of individual evaluation.

FUNDING

This work was supported by the Key Program of National Natural Science Foundation of China (Grant No. 61133005); the Key Projects in the National Science & Technology Pillar Program (2012BAH09B02); the National Natural Science Foundation of China (Grant Nos. 61070057, 61103047); the Cultivation Fund of the Key Scientific and Technical Innovation
Project, Ministry of Education of China (Grant No. 708066); and the National Science Foundation for Distinguished Young Scholars of Hunan (12JJ1011).

REFERENCES


