Strict and Heuristic Optimization of Virtual Machine Placement and Migration

SATORU OHTA
Department of Information Systems Engineering, Faculty of Engineering
Toyama Prefectural University
5180 Kurokawa, Imizu-shi, Toyama, 939-0398
JAPAN
ohta@pu-toyama.ac.jp https://www.researchgate.net/profile/Satoru_Ohta

Abstract: - In virtualization, power consumption is minimized and good performance is achieved by optimally performing virtual machine placement and migration. This paper investigates two approaches to optimal virtual machine placement and migration. First, a mixed integer programming (MIP) approach is presented. For this approach, the problem is formulated by mathematically representing the number of migrations, heterogeneous power consumption, and multiple performance-related resources. Although this approach is time-consuming and possibly impractical, it is indispensable in evaluating the goodness of other methods. In addition to the MIP, a heuristic approach is also examined, which relies on a metric that estimates the efficiency of virtual machine placement. The paper proposes a new metric to estimate efficiency more accurately. The effectiveness of the proposed metric is confirmed via computer simulation. Simulation also evaluates how parameters used in the algorithm affect solutions. Moreover, the solution obtained by the heuristic algorithm is compared with that obtained by the MIP approach.

Key-Words: - virtualization, live migration, optimization, heuristic, mixed integer programming

1 Introduction

Virtualization [1], [2] is a key technology for cloud computing, which is the indispensable basis of information technology. Virtualization provides various advantages, including flexibility, availability, scalability, etc.

Virtualization enables multiple virtual machines (VMs) to run on a single physical machine (PM). When VMs are hosted on multiple PMs, VMs should be optimally placed on PMs to minimize electrical power consumption without overutilizing computational resources. An important feature of virtualization is live migration [3], which enables a VM to move from the current host to another PM without stopping computation. When using live migration, VM placement can always be optimized for changing load [4].

To minimize power consumption under the constraint on resource capacity, we need to develop an algorithm that optimizes VM placement for dynamic load. Meanwhile, it is known that migration offers considerable load on the network [5]. Thus, the algorithm should avoid too frequent migrations. Moreover, the algorithm must quickly find a solution for real time control of VMs. Therefore, a fast heuristic approach is required.

This paper first formulates the VM placement into a linear mixed integer programming (MIP) problem [6]. Through this formulation, it becomes possible to strictly minimize power consumption under the constraints on resource capacity and migration count. This approach is time consuming and may be impractical. However, the approach is necessary to evaluate the goodness of a heuristic algorithm.

This paper also examines a heuristic algorithm, which is basically proposed in the author's previous study [7]. This paper introduces a new metric that estimates the efficiency of the VM placement on a PM. The algorithm is tested by computer simulation to evaluate the effectiveness of the proposed metric. The simulation also clarifies the optimal values for the parameters used in the heuristic. In addition, the solution obtained by the heuristic is compared with that obtained by the MIP approach.

This paper is organized as follows. First, related studies are reviewed in Section 2. Then, the problem tackled in this study is explained in Section 3. Section 4 presents how the VM placement problem is formulated as an MIP problem. A heuristic algorithm is explored in Section 5. Moreover, a new efficiency metric is also presented. The MIP and heuristic

ISBN: 978-1-61804-276-7

approaches are evaluated by computer simulation in Section 6. Finally, Section 7 concludes the paper.

2 Related Work

The VM placement and migration problem have been studied from various perspectives [4], [7-10].

Reference [4] presents a method to classify the load characteristics for obtaining gain by migration, a load forecasting technique, and a VM placement algorithm. In [9], a migration scheme is examined by considering various factors, including temperature, CPU, disk I/O, and network I/O in the physical and virtual machine layers. The algorithm presented in [8] minimizes the number of PMs by considering multiple performance-related resources and keeping their consumptions less than a given bound. Multiple resources are also assumed in the method of [7]. This method also assumes heterogeneous power consumption and minimizes the total power consumption. Meanwhile, the method of [10] considers only one resource, CPU. However, the method assumes sophisticated power management, which is used in computers today.

As seen above, the VM placement problem has been studied for various models. However, the assessment on solution goodness of algorithms is insufficient is a common problem found in previous studies. To rectify this problem, it is required to develop a method that yields a strict optimum.

3 Problem Description

Suppose that many VMs are supported by multiple PMs. In this situation, if a few PMs support as many VMs as possible, the required number of PMs will be small, and unused PMs can be turned off. Unfortunately, it is impossible for a PM to operate for unlimited number of VMs because the performance will degrade due to resource shortage. Thus, we must appropriately place each VM on a PM to avoid resource shortage as well as save electrical power.

When different loads are offered to VMs, the optimal placement of VMs is a combinatorial optimization problem. By solving this problem, it becomes possible to achieve a lower electrical power cost as well as avoid performance degradation caused by resource shortage.

If the load on VMs dynamically changes with time, the optimal VM placement will also change. Thus, some VMs must migrate to achieve the optimal placement. Essentially, the determination of optimal VM placement for new load is equivalent to performing the best migration.

It is known that migration offers considerable load on the network [5]. Thus, too frequent migrations should not be allowed. Therefore, power consumption should be minimized through as few migrations as possible.

Migration is categorized into two different types. The first type is executed to avoid performance degradation caused by overload, which is triggered by the load increase on VMs. For this type of migration, if an appropriate destination PM does not exist, some sleeping PM should be turned on. The second type of migration is performed to decrease the number of operating PMs by integrating lightly loaded VMs into as few PMs as possible. If a PM is vacated by this type of migration, it can be turned off. Hereafter, let us refer to these types as type 1 and type 2, respectively. We need to develop an algorithm for both of these migration types.

3.1 Stability Problem

To perform migration, we must determine the source (sender) PM, destination (receiver) PM, and target VM to migrate. An important problem for this decision is to avoid an infinitely repeated migration. To understand the possibility of infinite migrations, consider the following case of type 1 migration. Let PM_1, PM_2, and VM_1 denote the source PM, destination PM, and target VM, respectively. Infinitely repeated executions occur when the remaining capacity of PM_2 is not sufficient to accept VM_1. If this happens, the performance of PM_2 will degrade after migration. Thus, the VM must be moved again and possibly sent back to PM_1. Obviously, this causes performance degradation on PM_1. If the same algorithm is applied, VM_1 will be returned to PM_2. Thus, VM_1 will infinitely migrate between PM_1 and PM_2.

Infinite migrations can also occur for type 2 migration when the algorithm selects a VM that has already moved as the target and sends it back to its original host. Thus, a migration algorithm must be carefully designed to avoid infinite migrations.

3.2 Assumptions

Suppose that m VMs should be optimally placed among n PMs at discrete time $t = 0, 1, 2, \ldots, T$ with a constant interval. Thus, migration is performed at $t = 1, 2, \ldots, T$ to modify the placement determined at $t - 1$. VMs are denoted by $VM_1, VM_2, \ldots, VM_m$ and the PMs are denoted by $PM_1, PM_2, \ldots, PM_n$. The

performance of a VM relies on K computational resources R_1, R_2, …, R_K. Each VM consumes the resouce of its host PM. Let $u_{i,k}^{(t)}$ denote the percentile value that represents how much VM_i $(1 \le i \le m)$ consumes resource R_k $(1 \le k \le K)$ of its host at time t. It is assumed that $u_{i,k}^{(t)}$ is given for every i and k at t. The resource consumption of PM_j, denoted by $U_{j,k}^{(t)}$, is the sum of the resource consumption by each VM hosted by PM_j. It is also assumed that the performance of VMs hosted by PM_j does not degrade during the period from t to $t + 1$, if $U_{j,k}^{(t)}$ does not exceed a given percentile constant $U_{\max}$ for every k on PM_j.

This paper considers the situation where the electric power consumed by PMs is heterogeneous. Thus, power consumption must be minimized by considering the difference among the specifications of PMs. It is assumed that power consumption depends on resource consumption $U_{j,k}^{(t)}$.

4 MIP Approach

Under the assumptions described in 3.2, the VM placement problem can be formulated as a linear MIP problem [6]. This means that the strictly optimal solution is obtainable by using optimization software listed in [11]. Hereafter, this scheme is referred to as the MIP approach.

A solution for the VM placement problem should achieve less power consumption and fewer migrations by satisfying the constraints on resource consumption on PMs. Thus, the problem is considered as the optimization for two distinct objectives. To tackle this problem, we examine the following two scenarios.

1) Energy consumption is minimized under the constraint that the number of migrations does not exceed a specified bound.

2) The weighted sum of energy consumption and the migration count is minimized.

For VM_i $(1 \le i \le m)$ and PM_j $(1 \le j \le n)$, the placement at time t is represented by the following the binary (0-1) variable.

$$x_{i,j}^{(t)} = \begin{cases} 1, & \text{if } VM_i \text{ is placed on } PM_j \text{ at } t \\ 0, & \text{otherwise} \end{cases}. \quad (1)$$

To estimate the energy consumption, we need to know which PMs are turned on among n PMs. This is expressed by the next binary variable.

$$y_j^{(t)} = \begin{cases} 1, & \text{if } PM_j \text{ is turned at } t \\ 0, & \text{otherwise} \end{cases}. \quad (2)$$

The number of migrations is counted by introducing the following variable.

$$z_i^{(t)} = \begin{cases} 1, & \text{if } VM_i \text{ migrates at } t \\ 0, & \text{otherwise} \end{cases}. \quad (3)$$

The purpose of the problem is to decide these variables for a given condition of resource and power consumption. As described in 3.2, $U_{j,k}^{(t)}$ is defined as

$$U_{j,k}^{(t)} = \sum_{i \in \{i | VM_i \text{ is placed at } PM_j\}} u_{i,k}^{(t)}. \quad (4)$$

Then, the placement should be determined such that $U_{j,k}^{(t)}$ will not exceed $U_{\max}$ to avoid resource shortage on each PM.

Let $P_j^{(t)}$ denote the power consumed by PM_j at time t. Thus, the energy consumption is obtained by multiplying $P_j^{(t)}$ by the time interval. It is assumed that $P_j^{(t)}$ is defined as follows.

$$P_j^{(t)} = \begin{cases} P_{j,0} + \sum_{k=1}^{K} P_{j,k} U_{j,k}^{(t)}, & \text{if } PM_j \text{ is turned on} \\ 0, & \text{if } PM_j \text{ is turned off} \end{cases}. \quad (5)$$

Thus, the power consumption consists of a constant portion and portions proportional to resource consumption. As seen above, $P_j^{(t)}$ is a variable, which depends on the on/off state of PM_j as well as VMs assigned to PM_j.

With the decision variables $x_{i,j}^{(t)}$, $y_j^{(t)}$, $z_i^{(t)}$, $P_j^{(t)}$ and given constants $u_{i,k}^{(t)}$, $P_{j,0}$, $P_{j,k}$, the problem is formulated as follows:

$$\text{minimize} \quad \sum_{t=0}^{T} \sum_{j=1}^{m} P_j^{(t)} \quad (6)$$

subject to

$$P_j^{(t)} - P_{j,0} y_j^{(t)} + \sum_{k=1}^{K} \sum_{i=1}^{m} P_{j,k} u_{i,k}^{(t)} x_{i,j}^{(t)} = 0, \text{ for all } j, t, \quad (7)$$

$$\sum_{i=1}^{m} u_{i,k}^{(t)} x_{i,j}^{(t)} \le U_{\max}, \text{ for all } j, k, t, \quad (8)$$

$$\sum_{t=1}^{T} \sum_{i=1}^{m} z_i^{(t)} \le C, \text{ for all } t, \quad (9)$$

$$z_i^{(t)} - x_{i,j}^{(t-1)} + x_{i,j}^{(t)} \ge 0, \text{ for all } i, j, \text{ and } t > 0, \quad (10)$$

$$\sum_{j=1}^{n} x_{i,j}^{(t)} = 1, \text{ for all } i, t, \quad (11)$$

$$y_j^{(t)} - x_{i,j}^{(t)} \ge 0, \text{ for all } i, j, t, \quad (12)$$

$$x_{i,j}^{(t)}, y_j^{(t)}, z_i^{(t)} \in \{0,1\}, \text{ for all } i, j, t. \quad (13)$$

Here, parameter C is the upper bound for the number of migrations. Among the constraints, (7) determines the power consumed by PM_j. This equation is immediately obtained from (4) and (5).

ISBN: 978-1-61804-276-7

Resource consumption of a PM is set smaller than $U_{\max}$ by (8). The number of migrations does not exceed a given constant C by (9). The value of $z_i^{(t)}$ is determined by (10). This equation sets $z_i^{(t)}$ to 1 if VM_i migrates at time t. Strictly, $z_i^{(t)}$ may take 0 or 1 in (9) when VM_i does not migrate. However, we can precisely set the upper bound for the number of migrations with (9) and (10). By (11), a VM is certainly placed to one of the PMs. Moreover, a VM is assigned on a PM that is turned on with (12).

The weighted sum of energy consumption and migration count is minimized by a slightly modified formulation. Let w denote the weight parameter for the number of migrations. Migration is emphasized more for a larger value of w. Thus, by setting w at a large value, fewer migrations and greater power consumption will be obtained. The formulation for this model is as follows.

$$\text{minimize} \sum_{t=0}^{T}\sum_{j=1}^{m} P_j^{(t)} + w\sum_{t=1}^{T}\sum_{i=1}^{m} z_i^{(t)} \tag{14}$$

subject to

$$P_j^{(t)} - P_{j,0} y_j^{(t)} + \sum_{k=1}^{K}\sum_{i=1}^{m} P_{j,k} u_{i,k}^{(t)} x_{i,j}^{(t)} = 0, \text{ for all } j, t, \tag{15}$$

$$\sum_{i=1}^{m} u_{i,k}^{(t)} x_{i,j}^{(t)} \le U_{\max}, \text{ for all } j, t, \tag{16}$$

$$z_i^{(t)} - x_{i,j}^{(t-1)} + x_{i,j}^{(t)} \ge 0, \text{ for all } i, j, t, \tag{17}$$

$$\sum_{j=1}^{n} x_{i,j}^{(t)} = 1, \text{ for all } i, t, \tag{18}$$

$$y_j^{(t)} - x_{i,j}^{(t)} \ge 0, \text{ for all } i, j, t, \tag{19}$$

$$x_{i,j}^{(t)}, y_j^{(t)}, z_i^{(t)} \in \{0,1\}, \text{ for all } i, j, t. \tag{20}$$

The difference from the previous formulation is that the constraint on the migration count is omitted. Instead, the number of migrations is added to the objective function.

The above formulations assume that resource consumption for every t in $0 \le t \le T$ is known and VM placement over every t is simultaneously determined. However, this scenario is impractical. In the real world, VM placement must be determined online. This scenario is specified as follows.

Assume that current time is t. At this time, the given parameter is the resource consumption for t, $u_{i,k}^{(t)}$. Future consumptions $u_{i,k}^{(t+1)}$, $u_{i,k}^{(t+2)}$,…, are unknown. In addition, the VM placement determined before t, $x_{i,j}^{(0)}$,…, $x_{i,j}^{(t-2)}$, $x_{i,j}^{(t-1)}$, is fixed and cannot be changed. Moreover, it is unnecessary and impossible to determine future placement $x_{i,j}^{(t+1)}$, $x_{i,j}^{(t+1)}$,…, $x_{i,j}^{(T)}$. For this setting, the problem is to optimally place VMs to minimize the power consumption for the next time interval and the number of migrations executed at t.

The above online decision problem can also be formulated into an MIP problem. Let us $\hat{x}_{i,j}^{(t-1)}$ denote the placement determined at previous time period $t-1$. Since this value is a fixed constant, let us use a different symbol than $x_{i,j}^{(t)}$. Then, the weighted sum of the power consumption and migration count is minimized by the following formulation:

$$\text{minimize} \sum_{j=1}^{m} P_j^{(t)} + w\sum_{i=1}^{m} z_i^{(t)} \tag{21}$$

subject to

$$P_j^{(t)} - P_{j,0} y_j^{(t)} + \sum_{k=1}^{K}\sum_{i=1}^{m} P_{j,k} u_{i,k}^{(t)} x_{i,j}^{(t)} = 0, \text{ for all } j, \tag{22}$$

$$\sum_{i=1}^{m} u_{i,k}^{(t)} x_{i,j}^{(t)} \le U_{\max}, \text{ for all } j, \tag{23}$$

$$z_i^{(t)} + x_{i,j}^{(t)} = \hat{x}_{i,j}^{(t-1)}, \text{ for all } i \text{ and } j \in \{j \mid \hat{x}_{i,j}^{(t-1)} = 1\}, \tag{24}$$

$$\sum_{j=1}^{n} x_{i,j}^{(t)} = 1, \text{ for all } i, \tag{25}$$

$$y_j^{(t)} - x_{i,j}^{(t)} \ge 0, \text{ for all } i, j, \tag{26}$$

$$x_{i,j}^{(t)}, y_j^{(t)}, z_i^{(t)} \in \{0,1\}, \text{ for all } i, j. \tag{27}$$

The above formulation is valid for $t > 0$. For $t = 0$, since no migration is performed, the problem can be formulated by omitting (24) and the migration count term from the objective function.

By starting at $t = 0$ and solving the above problem for $t = 0, 1, 2, \ldots, T$ repeatedly, the VM placement for every time period is determined stepwise.

As seen above, the formulation is obtained by slightly modifying (14)-(20). A notable difference is seen in (24), the constraint that indicates the migration of VM_i. The equation forces $z_i^{(t)}$ to be 1 only when PM_i migrates at t.

5 Heuristic Approach

5.1 Algorithm Description

Since the VM placement problem is formulated as an MIP problem, the strictly optimal solution can be obtainable. However, this approach is highly time consuming and not practical. For real time control of VM migration, it is essential to develop a fast heuristic algorithm.

Basically, this study examines a modified version of the method presented in [7]. To describe this method concisely, let us define the following vectors.

$$\mathbf{u}_i = (u_{i,1}^{(t)}, u_{i,2}^{(t)}, ..., u_{i,K}^{(t)}), \quad 1 \le i \le m, \tag{28}$$

$$\mathbf{U}_j = (U_{j,1}^{(t)}, U_{j,2}^{(t)}, ..., U_{j,K}^{(t)}), \quad 1 \le j \le n. \tag{29}$$

Let u and U denote the vector sets $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m\}$ and $\{\mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_n\}$, respectively. Moreover, set S is defined as follows.

$$S = \{j \mid \text{PM}_j \text{ is hosting one or more VMs}\}. \tag{30}$$

Here, S is a set of PMs that should be turned on. PMs that are not in S can be turned off to save energy.

Using the above notation, the following basic algorithm determines VM placement at $t = 1, 2, \ldots, T$ by appropriately moving VMs.

Algorithm *decide_type*(S, U, u)

if $U_{j,k}^{(t)} > U_{\max}$ for some j and k **then**
  migrate_type1(S, U, u);
migrate_type2(S, U, u);

In the above algorithm, procedures *migrate_type1* and *migrate_type2* execute the two migration types identified in 3.1. For fast computation, these procedures employ the greedy method [12], which relies on a locally optimal decision to find a solution. In the algorithm, local optimality is estimated by a metric denoted by $F_j(\mathbf{U}_j)$. The metric $F_j(\mathbf{U}_j)$ represents the goodness of VM placement on PM_j and is detailed in 5.2. The procedures are described as follows.

Procedure *migrate_type1*(S, U, u)

L:= $\{j \mid U_{j,k}^{(t)} > U_{\max}$ for some $k\}$;
while L != {} **do**
  s:= null, v:= null, $x_{\max}$:= $-\infty$;
  for each $j \in L$ **do**
    for each $i \in \{i \mid \text{VM}_i \text{ is hosted on PM}_j\}$ **do**
      for each $l \in S - L$ **do**
        if $U_{l,k}^{(t)} + u_{i,k}^{(t)} \le U_{\max}$ for every k **then**
          x:= $F_j(\mathbf{U}_j - \mathbf{u}_i) + F_l(\mathbf{U}_l + \mathbf{u}_i) - F_j(\mathbf{U}_j) - F_l(\mathbf{U}_l)$;
          if $x > x_{\max}$ **then** s:= j, v:= i, d:= l, $x_{\max}$:= x;
        end if
  if v = null **then**
    turn on a new PM;
    d:= index of a newly turned-on PM;
    S:= S + $\{d\}$;
    for each $j \in L$ **do**
      for each $i \in \{i \mid \text{VM}_i \text{ is hosted on PM}_j\}$ **do**
        x:=$F_j(\mathbf{U}_j - \mathbf{u}_i) + F_l(\mathbf{U}_l + \mathbf{u}_i) - F_j(\mathbf{U}_j) - F_l(\mathbf{U}_l)$;
        if $x > x_{\max}$ **then** s:= j, v:= i, d:= l, $x_{\max}$:= x;
      end for
  end if
  send VM_v from PM_s to PM_d, and update L and U;
}

Procedure *migrate_type2*(S, U, u)

do
  $x_{\max}$:= ε ($\varepsilon > 0$);
  v:= null;
  for each $j \in S$ **do**
    for each $i \in \{i \mid \text{VM}_i \text{ is hosted by PM}_j\}$ **do**
      for each $l \in \{l \mid U_{l,k}^{(t)} + u_{i,k}^{(t)} \le U_{\max}$ for all $k\}$ **do**
        x:= $F_j(\mathbf{U}_j - \mathbf{u}_i) + F_l(\mathbf{U}_l + \mathbf{u}_i) - F_j(\mathbf{U}_j) - F_l(\mathbf{U}_l)$;
        if $x > x_{max}$ **then** s:= j, t:= i, d:= l, x_{max}:= x;
      end for
  if v = null **then break**;
  send VM_t from PM_s to PM_d and update U;
  if no VMs are hosted on PM_s **then**
    turn off PM_s
    S:= S–$\{s\}$;
  end if
end do

In the above procedures, variables s, d, and v are indices of the source, destination, and target, respectively. Variable x represents how the sum of efficiency metrics changes for the source and destination candidates before and after migration. Essentially, the procedures determine the combination of the source, destination, and target in order to maximize the increase of this efficiency metric sum.

In *migrate_type2*, ε is a constant that determines the initial value of $x_{\max}$. This constant is indispensable to assure stability. If ε is set to a large value, it becomes difficult for x to exceed the initial value of $x_{\max}$. Thus, the probability of discovering the target VM will decrease, leading to fewer migrations, a greater number of operating PMs, and larger power consumption. Thus, we can control the tradeoff between power consumption and the number of migrations by the value of ε. This feature is confirmed by computer simulation.

The above procedures determine the VM placement by applying migration to the placement at the previous time period. Thus, at $t = 0$, the initial solution is necessary. The initial solution is obtained by the *best-fit* algorithm shown in [8]. This algorithm is described as follows.

Algorithm *best-fit*(U, u)

j:= 1;
V:= set of VMs;
while $V \neq \{\}$ **do**
  T:=$\{v \mid v \in V$ and v can be placed at PM$_j\}$;
  if $T \neq \{\}$ **then**
    find $v \in T$ such that $f(\mathbf{U}_j + \mathbf{u}_v) \geq f(\mathbf{U}_j + \mathbf{u}_w)$ for $w \in T$;
    Assign v to PM$_j$, $\mathbf{U}_j = \mathbf{U}_j + \mathbf{u}_v$;
    V:= V–$\{v\}$;
    else j:=j + 1;
  end if
end while

In the algorithm, function $f(\mathbf{U}_j)$ shows the efficiency of resource utilization. This function is detailed in the next section.

5.2 Efficiency Metric

The algorithm presented in 5.1 utilizes a metric $F_j(\mathbf{U}_j)$. This metric should reflect two factors—how efficiently multiple resources are utilized and the power consumed by PM$_j$. These factors are considered as follows:

If performance depends on the utilization of two or more resources, all resources should be equally utilized to avoid unused resource capacities. This is explained by Fig. 1, which compares two cases of resource utilization on a PM. The figure assumes that two resources are relevant to system performance. Fig. 1(a) shows the case where both resources are almost equally utilized and thus have small unused capacities. Meanwhile, only one resource is fully utilized in Fig. 1(b). Obviously, the placement of Fig. 1(b) is inefficient because of the presence of a large unutilized resource on the machine. This placement will require more PMs and thus higher operational cost.

From the above intuition, it is possible to use metrics that represent how far the current state of resource utilization on a PM is from the ideal state. Here, the ideal state means that every resource of the PM is utilized to $U_{\max}$ %. In other words, the state is best fit to the resource capacities. Let $f(\mathbf{u}_j)$ denote the value of this fitting metric for PM$_j$.

As the fitting metric, a simple product of the resource consumption was used in [7], [8], which is written as,

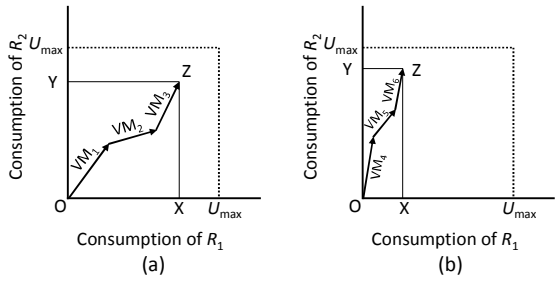

Fig. 1 Examples of resource consumption and the virtual machine placement: (a) balanced case and (b) unbalanced case.

$$f(\mathbf{U}_j) = \prod_{k=1}^{K} U_{j,k}^{(t)} . \quad (31)$$

The above function is maximized when the consumption of every resource reaches $U_{\max}$%. Thus, it can be used as the fitting metric. Actually, this function yielded relatively good solutions [7], [8]. However, it is uncertain whether the function is the best. There are other functions, which are maximized at the fully utilized state. As it turns out for the unbalanced utilization as shown in Fig. 1(b), the above function may not be appropriate. This is explained by the following numerical example.

Let us assume that two resources are concerned with performance. Then, consider the following two cases. For the first case, the consumption of each resource is 60%. The second case assumes that resource R_1 is consumed by 40% and resource R_2 is consumed by 90%. For both these cases, the output of $f(\mathbf{U}_j)$ is 3600. Thus, no difference is found by the function between the cases. However, in the second case, resource consumption is more unbalanced and likely to be more difficult to fully utilize every resource. Thus, a better result may be obtained by employing some other function, which outputs a smaller value for the second case than for the first case.

From the above consideration, this paper examines the following function:

$$f(\mathbf{U}_j) = \prod_{k=1}^{K} U_{j,k}^{(t)} \Bigg/ \left(\sum_{k=1}^{K} U_{j,k}^{(t)} \right)^{\alpha} . \quad (32)$$

In this function, α is a constant. Let us review the above numerical example assuming that α is 0.5 in (32). Then, the function outputs 328.6 for the first case and 315.7 for the second case. Thus, a larger value is obtained for balanced resource utilization. It is expected that this characteristic provides better solutions.

To reflect power consumption in the metric, the ratio of the fitting function $f(\mathbf{U}_j)$ and the power consumed by PM_j was used in [7]. However, it is uncertain whether the weight is appropriate between the fitting function and power consumption. Instead, this study examines the following metric:

$$F_j(\mathbf{U}_j) = f(\mathbf{U}_j) \Big/ \left(P_j^{(t)}\right)^{\beta}. \quad (33)$$

Here, β is the parameter that determines the weight between the fitting function and power consumption. If β is larger, power consumption is emphasized more. On the contrary, resource utilization is emphasized for a smaller value of β.

5.3 Stability of the Algorithm

The presented computational procedures finish with a finite number of migrations. For *migrate_type1*, migrations clearly stop after the overload of every PM is removed. For *migrate_type2*, it is necessary to show that migrations are not repeated permanently. For this problem, there is the following property.

Property Procedure *migrate_type2* stops after a finite number of migrations.

Proof) In the **do** loop of procedure *migrate_type2*, when VM_v migrates from PM_s to PM_d, the value x is defined as

$$x = F_s(\mathbf{U}_s - \mathbf{u}_v) + F_d(\mathbf{U}_d + \mathbf{u}_v) - F_s(\mathbf{U}_s) - F_d(\mathbf{U}_d).$$

The migration from PM_s to PM_d does not affect the value of $F_j(\mathbf{U}_j)$ for $j \neq s, d$. The above equation means that x is the difference of the sum $\Sigma_j F_j(\mathbf{U}_j)$ before and after the migration.

In *migrate_type2*, the sum increases at least by $x > \varepsilon$ if migration is performed. Meanwhile, the sum has an upper bound. Let $\hat{U}$ denote the maximum value among $U_{j,1}^{(t)}, U_{j,2}^{(t)}, \ldots, U_{j,K}^{(t)}$. Then, (32) implies

$$f(\mathbf{U}_j) \le \hat{U}^{K-\alpha} \le U_{\max}{}^{K-\alpha}, \quad (34)$$

for $\alpha < K$. Since $P_j^{(t)}$ is not smaller than $P_{j,0}$, $F_j(\mathbf{U}_j)$ is bounded by

$$F_j(\mathbf{U}_j) \le U_{\max}{}^{K-\alpha} \Big/ P_{j,0}. \quad (35)$$

The above equation confirms that the increase of $\Sigma_j F_j(\mathbf{U}_j)$ by migration cannot be repeated permanently. Thus, the loop is finished after a finite number of repetitions. (end of proof)

6 Evaluation

The heuristic algorithm presented in Section 5 was evaluated by computer simulation from two viewpoints. First, the effectiveness of employing the new efficiency metric defined by (33) is evaluated. The optimal values of parameters α and β are also determined. The second viewpoint compares the approximate solutions obtained by the heuristic algorithm with the near optimal solutions obtained by the MIP approach.

The simulation model involved 40 VMs and 20 PMs. Thus, m = 40 and n = 20. The number of performance-related resources, K, was 2. The maximum resource consumption, $U_{\max}$, was 90%.

The constants that determine the power consumption were given as follows:

$$P_{0,j} = 80, P_{1,j} = 0.4, P_{2,j} = 0, \text{ for } 1 < j \le 6, \text{ and}$$
$$P_{0,j} = 120, P_{1,j} = 0.6, P_{2,j} = 0, \text{ for } 6 < j \le 20.$$

The VM resource consumption $u_{i,k}^{(t)}$ was updated every hour. For each update of the consumption, the migration algorithm was executed. The simulation was executed for $0 \le t \le 72$. At t = 0, the initial solution is generated by algorithm *best-fit*.

VM resource consumption $u_{i,k}^{(t)}$ was given as follows. First, the VMs were divided into two groups, each of which consists of 20 VMs. For the first group, consumption gradually increases and decreases with a cycle of 24 hours. This is expressed as

$$u_{i,k}^{(t)} = \begin{cases} c_{i,k}(t - 24N)/12, & 24N < t \le 24N + 12 \\ c_{i,k}(24N + 25 - t)/12, & \text{otherwise} \end{cases}. \quad (36)$$

where N = 0, 1, 2 and $c_{i,k}$ is an integer constant, which was randomly selected with equal probability from 1 to 50. Since the actual server load for some applications alters periodically with a cycle of 24 h [4], the above model has some rationale. However, the above equation represents a gentle load change as shown in Fig. 2. Namely, the consumption increases and decreases with a small ratio of 1/12. Practically, the load on some service may more quickly increase or decrease and may have a different period length. To simulate such load, the following resource consumption was given to the second group of VMs.

$$u_{i,k}^{(t)} = \begin{cases} c_{i,k}, & \lfloor t/5 \rfloor \bmod 2 = 0 \\ 0.3c_{i,k}, & \lfloor t/5 \rfloor \bmod 2 = 1 \end{cases}. \quad (37)$$

By changing the random seed for $c_{i,k}$, 1000 problems were generated. For each problem, the power consumption and the number of migrations at each t were measured and then summed for t = 0, 1, …, 72. Since the update interval was 1 h, the sum of power consumption equals the energy measured in

Wh. The energy consumption and the total number of migrations were averaged over 1000 problems.

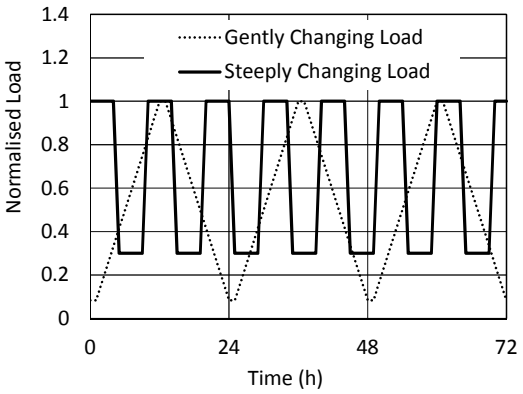

Fig. 2 Load offered to two groups of VMs.

It was difficult to obtain solutions by the MIP approach for 1000 problems because of excessive computational time. Thus, for the comparison of the heuristic algorithm and optimal solutions, 6 of the 1000 problems were used. The software `glpsol` distributed with `glpk` package [13] was used as the MIP solver.

6.1 Effectiveness of Efficiency Metric

Fig. 3 shows how the energy consumed for the heuristic algorithm depends on parameter α used in the fitting function. In the figure, parameter β was fixed to 0.0, 0.5, and 1.0. Parameter ε used in *migrate_type2* is 0.001. The figure shows that the energy consumption is minimized by selecting the value of α appropriately.

From (32) and (33), it is seen that the metric used in [8] is obtained by setting $\alpha = 0$ and $\beta = 0$ in $F_j(\mathbf{U}_j)$. Similarly, the metric employed in [7] is $F_j(\mathbf{U}_j)$ with setting $\alpha = 0$ and $\beta = 1$. Meanwhile, the figure clearly shows that energy consumption for appropriate values of α and β is smaller than that for $\alpha = 0$ and $\beta = 0$ as well as that for $\alpha = 0$ and $\beta = 1$. For example, if $\alpha = 0.4$ and $\beta = 1.0$, the energy is 0.6 kWh smaller than that obtained with using the metric of [7]. The smallest energy consumption was obtained for $\alpha = 0.48$ and $\beta = 0.8$. For this setting, the energy is saved by 0.61 kWh compared with the method of [7]. Thus, the metric of (33) becomes more effective than the metrics used in [7], [8] by selecting values of α and β appropriately. It is likely that this result comes from the characteristic of the fitting function $f(\mathbf{U}_j)$, which outputs a smaller value for unbalanced resource consumption.

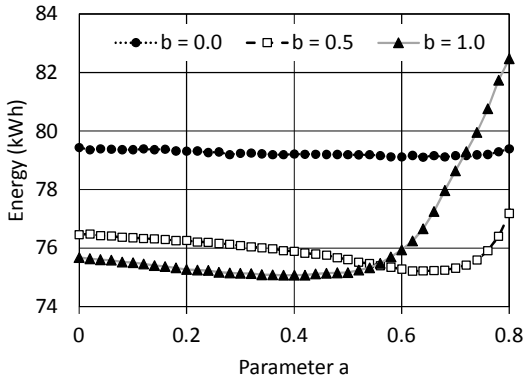

Fig. 3 Relationship between parameter α and energy consumption.

In Fig. 4, the energy consumption is plotted against parameter β, which determines the weight between power consumption and the fitting function. Parameter α is set to 0.0, 0.5, and 0.8, and ε is 0.001. Fig. 4 shows that the energy consumption is minimized by selecting an appropriate value of β. The figure shows that the characteristic considerably differs depending on the value of α. This is explained by the fact that the dimension of fitting function $f(\mathbf{U}_j)$ depends on α. That is, the dimension of $f(\mathbf{U}_j)$ is $(K-\alpha)$-th power of the percentile resource consumption. Thus, β should be larger for a smaller value of α.

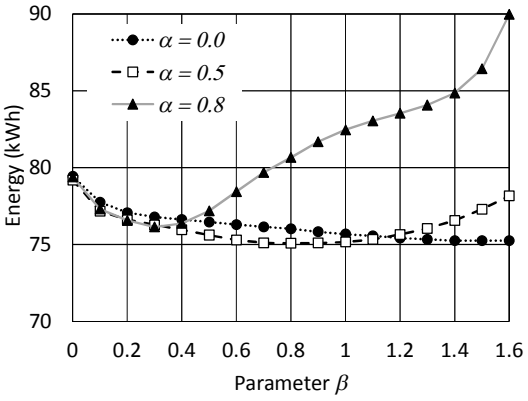

Fig. 4 Relationship between parameter β and energy consumption.

Fig. 5 displays how the number of migrations changes depending on α. The figure shows that the number of migrations does not significantly change for $\alpha < 0.5$. However, if $\beta = 1.0$, it decreases for $\alpha > 0.5$. Moreover, if $\beta = 0.5$, the migration count decreases for $\alpha > 0.7$. This result suggests that it becomes more difficult to discover the target of migration for these regions of α.

ISBN: 978-1-61804-276-7

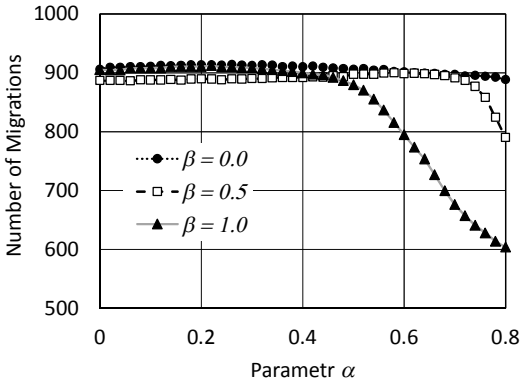

Fig. 5 Relationship between parameter α and the number of migrations.

The relationship between the number of migrations and energy consumption is plotted for different values of ε in Fig. 6. The figure shows the case of $\alpha = 0.5$ and $\beta = 0.8$. As expected, Fig. 6 shows that the value of ε successfully controls the tradeoff between the number of migrations and energy consumption. By setting $\varepsilon = 1$, the migration count reduces to 28% of that obtained for $\varepsilon = 0.001$. Meanwhile, energy consumption for $\varepsilon = 1$ becomes 1.28 times larger than that for $\varepsilon = 0.001$. Thus, by setting ε to a larger value, the number of migrations can be significantly reduced by sacrificing energy consumption. Therefore, if the network load caused by migration is critical, it is recommended to set ε to a large value.

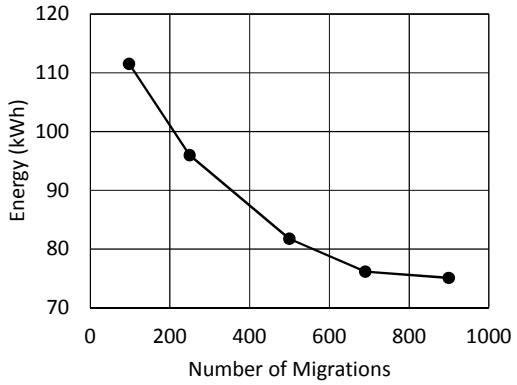

Fig. 6 Relationship between the number of migrations and energy consumption for different values of parameter ε.

6.2 Heuristic Algorithm vs. MIP Approach

Fig. 7 compares the energy consumption obtained by the MIP approach with that obtained by the heuristic algorithm. For the MIP approach, the online version defined by (21)-(27) was used. The weight parameter w was set to 0.001 and 1. For a larger value of w, the number of migrations is more emphasized. Thus, the number of migrations decreases while energy consumption may increase for $w = 1$. The MIP solver was executed with a time limit of 1800 s. Thus, the solutions obtained are not strictly optimum. For the heuristic algorithm, α and β are 0.48 and 0.8, respectively, while ε is 0.001.

As Fig. 7 shows, the solution obtained by the MIP approach yields lower energy consumption than that by the heuristic. The energy consumed by the heuristic algorithm is 1.04 times larger than that for the MIP approach with $w = 1$.

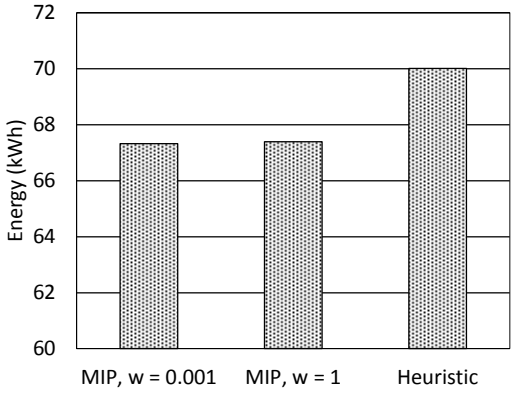

Fig. 7 Energy consumption for the MIP and heuristic approaches.

In Fig. 8, the number of migrations is compared between the MIP and heuristic approaches. The figure clearly shows that the heuristic approach yields considerably more migrations than the MIP approach. The number of migrations by the heuristic is 1.3 times larger than that by the MIP approach with $w = 0.001$ and 1.8 times larger than that of the MIP approach with $w = 1$. As seen before, the migrations of the heuristic algorithm can be decreased by a larger value of ε. However, this further degrades the power consumption.

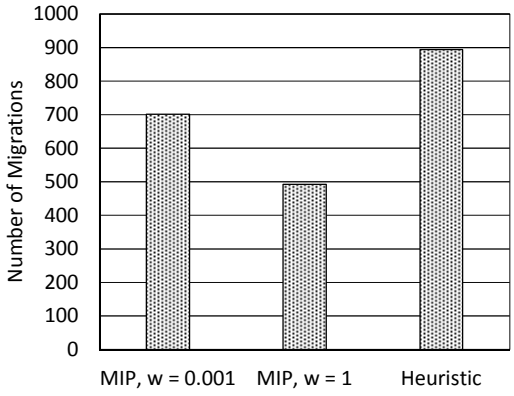

Fig. 8 The number of migrations for the MIP and heuristic approaches.

Fig. 9 plots the number of migrations against time for the MIP and heuristic approaches for one of problems. For the MIP approach, weight *w* was 0.001. Fig. 9 shows that the number of migrations is much greater for the heuristic algorithm particularly when load quickly increases. The probable cause of this result is that the method may perform migration twice for the same VM through *migrate_type1* and *migrate_type2* when load increases. Thus, the number of migrations may be reduced by eliminating this redundancy.

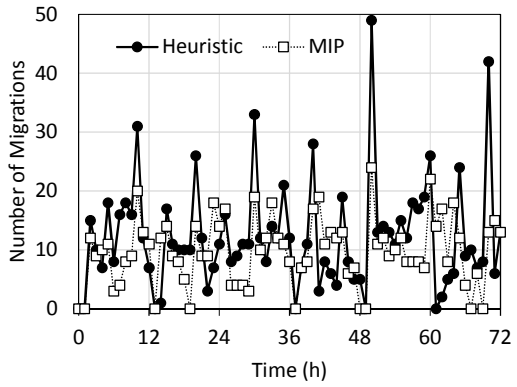

Fig. 9 The number of migrations vs. time for the MIP and heuristic approaches.

From the above result, it is concluded that the heuristic algorithm presented in Section 5.1 should be improved. Specifically, it is necessary to reduce migrations while maintaining low power consumption.

7 Conclusion

This paper explored two approaches to the optimal VM placement problem. First, the paper presented the formulation of the problem to obtain a strictly optimal solution through an MIP approach. It was shown how to represent VM migration by equations and achieve the least power consumption and migration count. The paper also examined a fast, but approximate heuristic approach. The algorithm presented is based on the method proposed in the author's previous work. However, a new efficiency metric is introduced to precisely estimate local optimality.

The approaches were evaluated by computer simulation. As a result, it was found that the new efficiency metric proposed for the heuristic algorithm is effective to reduce power consumption by setting parameter values appropriately. The simulation result also shows that the tradeoff between power consumption and migration count is successfully controlled by parameter setting. Moreover, the solutions were compared for the heuristic and MIP approaches. The result suggests that the solution optimality by the heuristic approach should be improved by reducing migrations keeping power consumption at a minimum.

Acknowledgement

The authors would like to thank Enago (www.enago.jp) for the English language review.

References:

[1] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauery, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in Proc. SOSP '03, Bolton Landing, New York, USA, 2003, pp. 164-177.

[2] J. Sahoo, S. Mohapatra, and R. Lath, "Virtualization: a survey on concepts, taxonomy and associated security issues," in Proc. Computer and Network Technology (ICCNT), 2010 Second International Conference on, 2010, pp. 222-226.

[3] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in Proc. USENIX NSDI '05, Boston, MA, USA, 2005, pp. 273-286.

[4] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing SLA violations," in Proc. IM '07, Munich, Germany, 2007, pp. 119-128.

[5] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya, "Cost of virtual machine live migration in clouds: a performance evaluation," Cloud Computing,, Lecture Notes in Computer Science, Vol. 5931, 2009, pp. 254-265.

[6] H. P. Williams, Model Building in Mathematical Programming. Wiley, 2013.

[7] S. Ohta and A. Sakai, "Virtual machine migration methods for heterogeneous power consumption," in Proc. the 11th IEEE International Conference on Autonomic and Trusted Computing (ATC-2014), Bali, Indonesia, 2014, pp. 577-582.

[8] S. Ohta, "Virtual machine placement algorithms to minimize physical machine count," in Proc. The 15th Asia-Pacific Network Operations and Management Symposium (APNOMS 2013), Hiroshima, Japan, 2013, pp. poster2-4.

[9] J. Xu and J. A. B. Fortes, "A Multi-objective approach to virtual machine management in datacenters," in Proc. ICAC '11, Karlsruhe, Germany, 2011, pp. 225-234.

[10] D. G. d. Lago, E. R. M. Madeira, and L. F. Bittencourt, "Power-aware virtual machine scheduling on clouds using active cooling control and DVFS," in Proc. MGC '11, Lisbon, Portugal, 2011.

[11] J. J. More and S. J. Wright, Optimization Software Guide. Philadelphia: SIAM, 1993.

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms. MIT Press, 2009.

[13] GLPK (GNU Linear Programming Kit), Available: http://www.gnu.org/software/glpk/