Power Efficient Virtual Machine Packing for Green Datacenter

Satoshi Takahashi[1], Atsuko Takefusa[3], Maiko Shigeno[2], Hidemoto Nakada[3], Tomohiro Kudoh[3] and Akiko Yoshise[2]

[1] University of Electro-Communications

[2] University of Tsukuba

[3] National Institute of Advanced Industrial Science and Technology (AIST)

Cloud computing is now considered to be a new computing paradigm to provide scalable Infrastructure, Platform and Software as a Service via the Internet. While, the diffusion of Cloud computing is expected to cause an explosive increase in power consumption for IT resources in data centers. Virtual Machine(VM)-based flexible capacity management is an effective scheme to reduce total power consumption in the data centers. However, there remain the following issues, trade-off between power-saving and user experience, decision on VM packing plans within a feasible calculation time, and collision avoidance for multiple VM live migration processes. In order to resolve these issues, we propose two VM packing algorithms, a matching-based (MBA) and a greedy-type heuristic (GREEDY). MBA enables to decide an optimal plan in polynomial time, while GREEDY is an aggressive packing approach faster than MBA. We investigate the basic performance and the feasibility of proposed algorithms under both artificial and realistic simulation scenarios, respectively. The basic performance experiments show that the algorithms reduce total power consumption by between 18% and 50%, and MBA makes suitable VM packing plans within a feasible calculation time. The feasibility experiments employ two power consumption models, one is the linear model and the other is piecewise linear model. In the linear model, the feasibility experiments show that the reduction ratio of total power consumption observed with MBA is smaller than that of GREEDY, but the performance degradation of MBA is less than that of GREEDY. In the piecewise-linear model, the feasibility experiments show that MBA investigates more reducing power consumption than GREEDY. The performance degradation of MBA is also less than GREEDY.

Keywords: Keywords related to the papers, comma separated.

1. INTRODUCTION

Cloud computing is now considered to be a new computing paradigm to provide scalable Infrastructure, Platform and Software as a Service (IaaS / PaaS / SaaS) via the Internet. Cloud providers manage large numbers of IT resources in data centers and provide services running on those resources in a pay-as-you-go paradigm. However, the diffusion of Cloud computing is expected to cause an explosive increase in power consumption for IT resources in data centers.

One of the reasons for this is the difficulty of proper planning of data center capacity planning. The frequency of service requests and their loads generally fluctuate, and thus cannot be estimated in advance. Cloud providers need to design their data center capacity to provide IT resources that can process assumed 'peak' service loads, because user experience will decrease if the capacity is insufficient. Therefore, there is a risk of a waste of IT resources when service loads are low. In addition, the accumulated power consumption of idle servers is wasted because their power consumption may equal about 50% of that of a saturated server, even if the server supports DVFS (Dynamic Voltage Frequency Scaling) technology. One of the key technologies used to reduce the total power consumption of a data center is server consolidation, which makes

This study is partially supported by JST/CREST ULP and Grant-in-Aid for Scientific Research (KAKENHI 22510135, 24·1920). Author's address: S. Takahashi, Faculty of Informatics and Engineering, University of Electro-Communications, 1-5-1 Chofugaoka, Chofu, Tokyo, 1828585; M. Shigeno and A. Yoshise, Faculty of Engineering, Information and Systems, University of Tsukuba, 1-1-1 Tennoudai, Tsukuba, Ibaraki, 3058573; A. Takefusa, H. Nakada and T. Kudoh, National Institute of Advanced Industrial Science and Technology (AIST), 1-1-1 Umezono, Tsukuba, Ibaraki, 3058568.

multiple low loaded-services run on a few servers, allowing idle servers to be switched off. An effective scheme to achieve server consolidation is Virtual Machine(VM)-based flexible capacity management implemented using VM live migration technologies for VM host relocation. VM-based capacity management can consolidate multiple VMs onto a few physical machines (PMs) and allow other PMs to be put in 'stand-by' mode. In order to avoid degradation of service performance, stand-by PMs will be resumed when service loads increase. Hirofuchi, et al., have proposed a VM-based server consolidation system based on their post-copy VM live migration technology, which enables live migration with less downtime [Hirofuchi et al. 2011b]. Such a system may achieve flexible server consolidation, and, hopefully, its users will not experience degradation in service performance.

In order for effective VM-based server consolidation, there are VM packing issues:

(1) Trade-off between power consumption and performance: An aggressive packing scheme may cause a remarkable reduction of power consumption while experiencing a considerable amount of performance degradation. We have to consider both the performance metrics in order to investigate the feasibility of packing algorithms.

(2) VM packing planning within a feasible calculation time: In order for efficient server consolidation, packing systems have to remap VMs periodically, because each VM load fluctuates constantly. In addition, a VM packing problem is an $\mathcal{NP}$-complete problem. It is important to make VM packing plans within a feasible calculation time.

(3) Collision of multiple live migrations: Post-copy migration technologies enable live migration with less downtime. However, it causes burst transfer of a VM image in the background after an ostensible migration even though the upper bound of migration is (# of PMs)/2. Remapping of selected VMs takes a long time if collision between multiple VM image transfers has occurred.

To address the above issues, we propose VM packing algorithms that aim to reduce power consumption, assure the user experience in terms of performance, and avoid collision of multiple live migrations. We define the VM packing problem as 0-1 integer programming (IP) and propose a matching-based algorithm (MBA) and a greedy heuristic algorithm (GREEDY). MBA is an approach used to find an optimal packing plan in polynomial time. On the other hand, GREEDY is an aggressive packing approach. The proposed algorithms set a limit to the number of sending or receiving VMs for each PM in order to avoid collision of migrations.

We investigate basic performance and the feasibility of the proposed algorithms under both artificial and realistic simulation scenarios, respectively. In two distinct simulation scenarios, we use randomly generated artificial data and trace data from the 648 node T2K-Tsukuba supercomputer, provided by the Center for Computational Sciences of the University of Tsukuba, respectively. We compare MBA, GREEDY, IP and their variations in terms of power consumption, performance degradation, and calculation time. The basic performance experiments show that the packing algorithms reduce total power consumption by between 18% and 50%, and MBA makes suitable VM packing plans within a feasible calculation time. In the feasibility experiments, we employ two power consumption models; linear and piecewise-linear model. In the linear model, the feasibility experiments show that the reduction ratio of total power consumption observed with MBA is smaller than that of GREEDY, but the performance degradation of MBA is less than that of GREEDY. In addition, the results show that MBA can make packing plans for an actual supercomputer within a feasible calculation time. In the piecewise-linear model, the feasibility experiments show that MBA investigates more reducing power consumption than GREEDY. The performance degradation of MBA is also less than GREEDY.

2. RELATED WORK

There have been a lot of studies on power-aware VM packing algorithms that aim to reduce total power consumption by using VM migration technologies. A VM packing problem can be modeled as a bin-packing problem with different sized bins. However, the classical bin-packing problem is known as an $\mathcal{NP}$-complete problem [Garey and Johnson 2000]. Most of those studies have defined heuristic algorithms based on First Fit Decreasing (FFD) and Best Fit Decreasing (BFD) [Vazirani 2001].

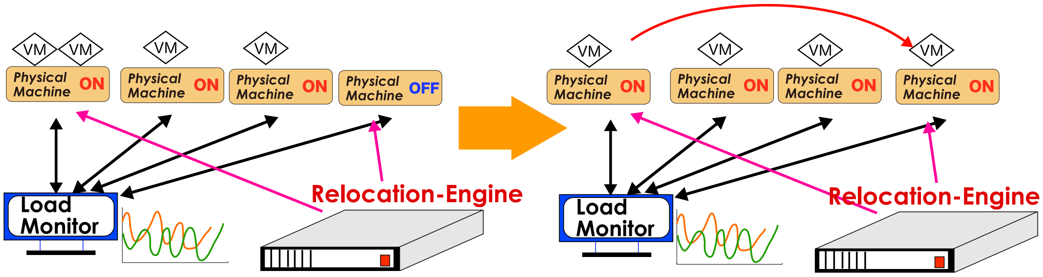
Chen, et al., have defined an Effective Sizing-based correlation-aware BFD heuristic algorithm [Chen et al. 2011]. 'Effective Sizing' is an estimation of a VM's resource demand through a stochastic approach. They assume a homogeneous environment and aim to reduce the number of active servers. Li, et al., have defined an energy-aware heuristic algorithm based on BFD [Li et al. 2009] that aims to narrow resource gaps between each PM's resource capacity and its VM loads. The algorithm showed lower power consumption than FFD and BFD. Performance degradation due to VM consolidation has not been well investigated in the above two studies. It is important to consider both power consumption and performance degradation, since there is a trade-off between them. Also, live migration overheads have not been considered.

Beloglazov, et al., have proposed a modified BFD algorithm with a minimizing of migrations policy [Beloglazov et al. 2012] that aims to minimize the number of VM migrations. They have investigated its performance in terms of total power consumption and SLA violation on heterogeneous environments over the CloudSim simulation platform [Calheiros et al. 2011]. Verma, et al., have considered an FFD-based algorithm with optimization functions [Verma et al. 2008], which consists of performance, power, and migration factors. The performance and power factors depend on SLA and power consumption, respectively. The migration factor is determined by live-migration cost, estimated by quantifying the decrease in throughput because of live migration and estimating the revenue loss because of the decreased performance. Sinha, et al., have proposed a BFD algorithm with dynamic higher and lower thresholds for CPU utilization for PMs at the data center [Sinha et al. 2011]. This idea is that the thresholds are defined to meet both power efficiency requirements and quality of service to the user by minimizing Service Level Agreement violations, since VM consolidation will work on dynamic and unpredictable workloads, avoiding unnecessary power consumption.

Y. Wang and X. Wang have defined a heuristics-based optimization algorithm to find a polynomial time approximate solution [Wang and Wang 2010]. The algorithm begins with the extended Minimum Bin Slack problem, which can be solved in pseudo-polynomial time [Fleszar and Hindi 2002]. Then, it allocates VMs to power-efficient servers, preferentially. In order to take migration overheads into account, they provide a migration cost function, which can be defined by a DC administrator. The algorithm showed better performance than the FFD-based algorithm. Nakada, et al., have investigated heuristic algorithms, based on FFD, the 0-1 Integer Programming (IP) model, and a Genetic Algorithm (GA) [Nakada et al. 2010]. This study showed that the 0-1 IP and GA-based algorithms are not feasible because of the long calculation time needed to solve the problem, while the FFD-based algorithm finds it difficult to minimize the total number of migrations.

The above five studies have considered three performance matrix; power consumption, performance degradation, and migration cost, related to the number of migrations. However, performance degradation due to collision of simultaneous live migrations has not been taken into account. In this work, we propose VM packing algorithms to avoid migration collision and investigate the performance based on power consumption and performance degradation metrics.

Takeda and Takemura have defined an extended FFD algorithm that selects a migration destination server on the basis of server rank [Takeda and Takemura 2010]. This rank represents a server selection priority. In order to improve search efficiency, only VMs hosted by a high-load or low-load PM become migration candidates. In addition, multiple VMs are not allowed to simultaneously migrate to a PM due to live migration overheads. This idea is similar to one in our algorithm, but we propose both heuristic and optimal algorithms, which aim to solve VM remapping in polynomial time.



Detail of VM packing system.

3. VIRTUAL MACHINE PACKING PROBLEM

As a method to reduce energy consumption in data centers, the virtual machine consolidation technique is widely used. The efficiency of the 'static' VM consolidation technique is largely limited by the maximum load-based capacity planning, where all the VMs are assured all the time to have sufficient resources to process any possible maximum load. In the 'dynamic' VM consolidation method, where VMs are dynamically assigned resources based on load, data center resources can be utilized more efficiently.

The 'VM Packing Problem' is designed to minimize the energy consumption of the whole data center while assuring all the VMs are assigned sufficient resources to handle the load at that time.

3.1 VM Packing Implementation

Hirofuchi *et. al.* proposed a virtual machine management system with a rather naive virtual machine packing implementation [Hirofuchi et al. 2010]. Figure 1 illustrates the virtual machine packing system that is proposed by [Hirofuchi et al. 2010]. There are three active PMs before migration. After migration, the system allocates a VM to a standby mode PM. The virtual machine packing system consists of a load monitor and a relocation engine. The load monitor observes virtual machines' resource usage amounts and stores observed data into a database. The relocation engine reallocates virtual machines to physical machines based on the observed data of resource usage amounts. When all virtual machines are lightly loaded, the system consolidates them onto few physical machines in the data center, keeping the rest of the servers in the ACPI S3 standby mode, in which each node consume only 5-7 watts. When the system detects that a virtual machine has became heavily loaded, it wakes up one of the standby physical machines and migrates the virtual machine onto the physical machine to provide the virtual machine sufficient resources.

With high-speed post-migration techniques[Hirofuchi et al. 2010], virtual machine migration can be performed within a second. It takes 1-2 seconds to wake up the standby physical machine in ACPI S3 mode. This means that we can reallocate resources to virtual machines in 2-3 seconds in total.

3.2 Problem Formulation

Given many VMs are located in a number of PMs, the *virtual machine packing problem* (VMPP) is to reallocate VMs to PMs such that minimizing total power consumption is achieved. Let P be a set of PMs and V a set of VMs. For representing a reallocation, we denote by V_i a set of VMs assigned to PM i. Each PM $i(\in P)$ has capacities of CPU and memory, denoted by c_i and m_i, respectively. Each VM $k \in V$ has CPU and memory usage, denoted by vc_k and vm_k, respectively. To avoid performance degradation, a set of VMs assigned to PM i has to satisfy resource constraints:

$$\sum_{k\in V_i} vc_k \leqslant c_i \tag{1}$$

$$\sum_{k\in V_i} vm_k \leqslant m_i \tag{2}$$

Moreover, each PM can send or receive at most one VM per one time in order to prevent performance loss on PMs due to collisions. Thus, each V_i needs to satisfy

$$|V_i^0 \setminus V_i| + |V_i \setminus V_i^0| \leqslant 1, \tag{3}$$

where V_i^0 stands for the set of VMs initially located in PM i.

Let $cost_i(z)$ be the electricity usage of the PM $i(\in P)$ with respect to its CPU usage z, and $base_i$ be the basic electricity usage. The total power consumption for reallocation $\{V_i\}_{i\in P}$ is given by

$$\sum_{i\in P}\{(cost_i(\sum_{k\in V_i} vc_k) + base_i) \mid V_i \neq \emptyset\}, \tag{4}$$

since the variation of electricity consumption of memory and network communication resource is low with respect to total power consumption due to observations made in [Hirofuchi et al. 2011a]. Then, VMPP is formulated as follows:

$$\text{(VMPP) minimize}\{(4) \mid (1)(2) \text{ and } (3)\ \forall i \in P\} \tag{5}$$

We now formulate this VMPP as a 0-1 integer programming problem by using 0-1 decision valuables,

$$x_{ik} = \begin{cases} 1 & (k \in V_i) \\ 0 & (\text{otherwise}), \end{cases} \tag{6}$$

and

$$y_i = \begin{cases} 1 & (V_i \neq \emptyset) \\ 0 & (\text{otherwise}). \end{cases} \tag{7}$$

For representing the initial location V_i^0, we introduce

$$x_{ik}^0 = \begin{cases} 1 & (k \in V_i^0) \\ 0 & (\text{otherwise}). \end{cases} \tag{8}$$

Then, VMPP can be rewritten as follows:

$$
\begin{aligned}
&\text{(VMPP-IP)} && \\
&\text{minimize} && \sum_{i\in P}(cost_i(\sum_{k\in V} vc_k x_{ik}) + base_i y_i) && (9)\\
&\text{subject to} && \sum_{k\in V} vc_k x_{ik} \le c_i,\ \forall i \in P && (10)\\
& && \sum_{k\in V} vm_k x_{ik} \le m_i,\ \forall i \in P && (11)\\
& && \sum_{k\in V} |x_{ik} - x^0_{ik}| \le 1,\ \forall i \in P && (12)\\
& && \sum_{i\in P} x_{ik} = 1,\ \forall k \in V && (13)\\
& && \sum_{k\in V} x_{ik} \le |V|\cdot y_i,\ \forall i \in P && (14)\\
& && x_{ik} \in \{0,1\},\ \forall i \in P, \forall k \in V && (15)\\
& && y_i \in \{0,1\},\ \forall i \in P. && (16)
\end{aligned}
$$

Obviously, the objective function (9) of VMPP-IP coincides with the total power consumption (4). Constraints (10) and (11) correspond to resource constraints (1) and (2), respectively. Constraint (12) is due to the restriction of the number of migrations by (3). The equation (12) is represented by the following equivalent linear equation using the property such that if the product of x_{ik} and x^0_{ik} is 1 then the VM k is reallocated to the PM i.

$$\sum_{k\in V}(x_{ik} + x^0_{ik} - 2x_{ik}x^0_{ik}) \le 1,\ \forall i \in P. \tag{17}$$

We need two additional constraints. Constraint (13) represents the fact that each VM has to be allocated to one of the available PMs. Constraint (14) implies that if each PM assigned more than one VM then the PM has to operate. For an optimal solution $\hat{\boldsymbol{x}}$ of VMPP-IP, define $V_i(\hat{\boldsymbol{x}}) = \{k \in V \mid \hat{x}_{ik} = 1\}$ for any $i \in P$. Obviously, the reallocation $\{V_i(\hat{\boldsymbol{x}})\}_{i\in P}$ is optimal for VMPP.

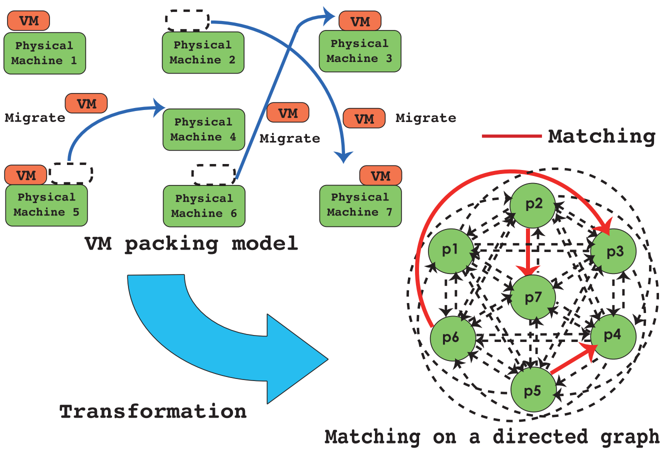
4. ALGORITHMS FOR VMPP

This section proposes two efficient algorithms for VMPP. One is an exact algorithm based on maximum weight matchings. The other is a greedy-type heuristic algorithm.

4.1 Matching-based Algorithm

We configure a matching-based algorithm for VMPP. Let (P, A) be a complete digraph, where the vertex set P is given by the PM set and the arc set $A = \{(i,j) \mid i,j \in P, i \neq j\}$ consists of all the pairs of both directed arcs. A subset M of arcs such that no two arcs in M are incident on the same vertex is referred to as *matching*. Note that a matching is one to one correspondence to a collection of a pair of PMs between which a migration of some VM occurs per one time without considering resource constraints, since the number of migrations from or to each PM is at most one per one time. For example, Fig. 2 shows the relationship between migrations of VMs and matching on (P, A). This example has seven PMs, and has three VMs that migrate from PMs 5, 2, and 6 to PMs 4, 7, and 3, respectively.

Recall that V^0_i is the set of VMs initially located in $i \in P$. By $\tilde{c}(i,j,k)$, the reduced power consumption is denoted when VM $k \in V^0_i$ is reallocated from $i \in P$ to $j \in P$. Then, $\tilde{c}(i,j,k)$ is given by



Example of translation.

$$cost_i(\sum_{k'\in V_i^0} vc_{k'}) - cost_i(\sum_{k'\in V_i^0\setminus\{k\}} vc_{k'}) + base_i \cdot \max\{0, 2-|V_i^0|\}$$

$$+ cost_j(\sum_{k'\in V_j^0} vc_{k'}) - cost_j(\sum_{k'\in V_j^0\cup\{k\}} vc_{k'}) + base_j \cdot \min\{0, |V_j^0|-1\}. \tag{18}$$

The weight $c(i,j)$ of each arc $(i,j) \in A$ is defined by

$$\max\{\tilde{c}(i,j,k) \mid k \in V^{i\to j}\}, \tag{19}$$

where $V^{i\to j}$ stands for a set of VMs each of which can be reallocated from $i \in P$ to $j \in P$ without violating resource constraints (1) and (2), i.e.,

$$V^{i\to j} = \left\{ k \in V_i^0 \;\middle|\; \begin{array}{l} \sum_{\ell\in V_i^0\setminus\{k\}} vm_\ell \le m_i, \\ \sum_{\ell\in V_i^0\setminus\{k\}} vc_\ell \le c_i, \\ \sum_{\ell\in V_j^0\cup\{k\}} vm_\ell \le m_j, \\ \sum_{\ell\in V_j^0\cup\{k\}} vc_\ell \le c_j \end{array} \right\}. \tag{20}$$

If a matching M corresponds to a set of migrations satisfying the resource constraints, the sum of weight $\sum_{(i,j)\in M} c(i,j)$ exhibits the reduced power consumption when the most efficient VMs migrate between PMs indicated by M. Representing a maximizer of (19), we use a function $\sigma:\ A \to V$ which returns a maximizer $k\ (\in V)$ for each $(i,j) \in A$, i.e., $c(i,j) = \tilde{c}(i,j,\sigma(i,j))$.

In order to treat only matchings that correspond to a set of migrations satisfying the resource constraints, we introduce an arc subset $\tilde{A} = \{(i,j) \in A \mid V^{i\to j} \neq \emptyset\}$. In addition, because the initial allocation might violate resource constraint (1) or (2) for some PMs, we define $U = \{i \in P \mid \sum_{k\in V_i^0} vc_k > c_i\} \cup \{i \in P \mid \sum_{k\in V_i^0} vm_k > m_i\}$ as a set of such machines. Since, for each PM i in U, some VM in V_i^0 has to migrate to another PM, a reallocation that satisfies the resource constraints can be represented by a matching in which each vertex in U is matched. Such a matching is referred to as *U-matching*.

THEOREM 4.1. *Let M^* be a maximum weight U-matching on $(P,\tilde{A})$ and*

$$V_i^* = \begin{cases} V_i^0 \setminus \{\sigma(i,j)\} & ((i,j) \in M^*), \\ V_i^0 \cup \{\sigma(j,i)\} & ((j,i) \in M^*), \\ V_i^0 & (otherwise), \end{cases} \tag{21}$$

for each $i \in P$. Then, $\{V_i^*\}_{i\in P}$ is an optimal reallocation for VMPP.

Proof. As in the above discussion, U-matchings on $(P, \tilde{A})$ are a one-to-one, correspondence to sets of migrations satisfying all constraint (1), (2) and (3). Thus, $\{V_i^*\}_{i \in P}$ is a feasible reallocation for VMPP. From decision of the arc set $\tilde{A}$ and U, it is clear that an obtained reallocation satisfies the resource constraints. Also each PM's number of migration is at most one, since the reallocation obtains from the matching. We now consider an optimal reallocation $\{\hat{V}_i\}_{i \in P}$ of VMPP. Let $\hat{Y}(\subseteq V)$ be a set of VMs reallocated to another PM from the initial location to the optimal reallocation, i.e., $\hat{Y} = \bigcup_{i \in P}(V_i^0 \setminus \hat{V}_i)$. For each VM $\tilde{k} \in \hat{Y}$, the pair of PMs that $\tilde{k}$ migrates from and to are denoted by $s(\tilde{k})$ and $r(\tilde{k})$, respectively. Then, $\hat{V}_{s(\tilde{k})} = V_{s(\tilde{k})}^0 \setminus \{\tilde{k}\}$ and $\hat{V}_{r(\tilde{k})} = V_{r(\tilde{k})}^0 \cup \{\tilde{k}\}$ hold, since the number of migrations from or to a PM is limited to at most one. The reduced power consumption for the optimal reallocation $\{\hat{V}_i\}_{i \in P}$ is given by

$$\sum_{i \in P}(cost_i(\sum_{k \in V_i^0} vc_k) + base_i \cdot \min\{1, |V_i^0|\}) - \sum_{i \in P}(cost_i(\sum_{k \in \hat{V}_i} vc_k) + base_i \cdot \min\{1, |\hat{V}_i|\}) \qquad (22)$$

Since the first term in (22) is constant, a reallocation minimizing (4) maximizes the reduced power consumption (22). By using $\hat{Y}$, we can rewrite the reduced power consumption (22) as

$$\begin{aligned}
&\sum_{\tilde{k} \in \hat{Y}} \left(cost_{s(\tilde{k})}(\sum_{k \in V_{s(\tilde{k})}^0} vc_k) - cost_{s(\tilde{k})}(\sum_{k \in \hat{V}_{s(\tilde{k})}} vc_k) + cost_{r(\tilde{k})}(\sum_{k \in V_{r(\tilde{k})}^0} vc_k) - cost_{r(\tilde{k})}(\sum_{k \in \hat{V}_{r(\tilde{k})}} vc_k) \right) \\
&\quad + \sum_{\tilde{k} \in \hat{Y}} \Big(base_{s(\tilde{k})}(\min\{1, |V_{s(\tilde{k})}^0|\} - \min\{1, |V_{s(\tilde{k})}^0| - 1\}) + base_{r(\tilde{k})}(\min\{1, |V_{r(\tilde{k})}^0|\} - \min\{1, |V_{r(\tilde{k})}^0| + 1\}) \Big) \\
&= \sum_{\tilde{k} \in \hat{Y}} \left(cost_{s(\tilde{k})}(\sum_{k \in V_{s(\tilde{k})}^0} vc_k) - cost_{s(\tilde{k})}(\sum_{k \in V_{s(\tilde{k})}^0 \setminus \{\tilde{k}\}} vc_k) + cost_{r(\tilde{k})}(\sum_{k \in V_{r(\tilde{k})}^0} vc_k) - cost_{r(\tilde{k})}(\sum_{k \in V_{r(\tilde{k})}^0 \cup \{\tilde{k}\}} vc_k) \right) \\
&\quad + \sum_{\tilde{k} \in \hat{Y}} \Big(base_{s(\tilde{k})} \cdot \max\{0, 2 - |V_{s(\tilde{k})}^0|\} + base_{r(\tilde{k})} \cdot \min\{0, |V_{r(\tilde{k})}^0| - 1\} \Big) \\
&= \sum_{\tilde{k} \in \hat{Y}} \tilde{c}(s(\tilde{k}), r(\tilde{k}), \tilde{k}). \qquad (23)
\end{aligned}$$

Clearly, since VM $\tilde{k}$ belongs to $V^{s(\tilde{k}) \to r(\tilde{k})}$, $\tilde{c}(s(\tilde{k}), r(\tilde{k}), \tilde{k}) \leqslant c(s(\tilde{k}), r(\tilde{k}))$ holds. By defining $\hat{M} = \{(o(k), i(k)) \in A \mid k \in \hat{Y}\}$, we have $\sum_{\tilde{k} \in Y} \tilde{c}(s(\tilde{k}), r(\tilde{k}), \tilde{k}) \leqslant \sum_{(i,j) \in \hat{M}} c(i,j)$. Note that $\hat{M}$ is U-matching on $(P, \tilde{A})$. Thus, for the maximum weight U-matching M^*, $\sum_{(i,j) \in \hat{M}} c(i,j) \leqslant \sum_{(i,j) \in M^*} c(i,j)$ holds. On the other hand, let $Y^* = \bigcup_{i \in P}(V_i^0 \setminus V_i^*)(= \{\sigma(i,j) \mid (i,j) \in M^*\})$. By replacing $\hat{Y}$ by Y^* in (23), we also show that $\sum_{(i,j) \in M^*} c(i,j)$ is equivalent to the reduced power consumption by $\{V_i^*\}_{i \in P}$. Therefore, the reduced power consumption of the reallocation V_i^* $(i \in P)$ following from M^* is not smaller than the one of the optimal reallocation $\hat{V}_i$ $(i \in P)$. Therefore, the reallocation V_i^* $(i \in P)$ is also optimal for VMPP. □

Thus, we can construct an algorithm for VMPP, which uses a maximum U-matching. We call the algorithm described below a matching-based algorithm (MBA).

Matching Based Algorithm (MBA)

Step 1.. Construct a directed graph $(P, \tilde{A})$ and $U = \{i \in P \mid \sum_{k \in V_i^0} vc_k > c_i\} \cup \{i \in P \mid \sum_{k \in V_i^0} vm_k > m_i\}$, and calculate a weight $c(i,j)$ for each $(i,j) \in \tilde{A}$.

Step 2.. Find a maximum weighted U-matching M^* on $(P, \tilde{A})$.

Step 3.. For each $(i,j) \in M^*$, reallocate $\sigma(i,j)$ from i to j. That is to say, make the reallocation $\{V_i^*\}_{i \in P}$ by (21).

In Step 2 of MBA, we can compute a maximum weighted U-matching in $\mathrm{O}(|P|^3)$ time or in $\mathrm{O}(|P||\tilde{A}| + |P|^2 \log |P|)$ time by using a polynomial time algorithm due to Edmonds [Edmonds 1965] (see also [Korte and Vygen 2004] [Galil 1986].)

THEOREM 4.2. *Assume that $cost_i(z)$ can be computed in $\mathrm{O}(1)$ time for any $i \in P$ and z. Then, MBA find an optimal reallocation for VMPP in $\mathrm{O}(|P|^3 + |V||P|)$ time. Hence, VMPP belongs to class $\mathcal{P}$.*

PROOF. In MBA, Step 1 spends $\mathrm{O}(\sum_{i \in V} |V_i^0||P|) = \mathrm{O}(|V||P|)$ time, because it needs to search all elements in V_i^0 to compute weight $c(i,j)$ for each $(i,j) \in \tilde{A}$, which implies that it computes all weights $c(i,j)$ for each arc (i,j) leaving from vertex i in $\mathrm{O}(|V_i^0||P|)$ time. Since $|\tilde{A}| = \mathrm{O}(|P|^2)$, Step 2 spends $\mathrm{O}(|P|^3)$ time to compute a maximum weighted U-matching. Step 3 spends $\mathrm{O}(|P|)$ time. Summing these complexities, we obtain the desired complexity. □

4.2 Greedy Algorithm

Next we propose a greedy-type heuristic algorithm (GREEDY) for VMPP. To represent the resource usage ratios of each PM i, define

$$util_i = \frac{\sum_{k \in V_i^0} vc_k}{c_i} + \frac{\sum_{k \in V_i^0} vm_k}{m_i}. \tag{24}$$

GREEDY first tries to migrate one VM from a PM for which resource constraint (1) or (2) does not hold at the initial allocation. For such PMs to need migration away, a priority is given by the value of $util_i$. The algorithm subsequently tries to perform migrations also by using a priority of $util_i$, in order to reduce the number of active PMs. We describe the details of GREEDY as follows.

Greedy Algorithm (GREEDY)

Step 1.. Partition P into two sets $Low = \{i \in P \mid (1) \text{ and } (2) \text{ hold}\}$ and $High = P \setminus Low$. (Store PMs in Low and $High$, respectively, by descending order of $util_i$.)

Step 2.. Migrate one VM from each PM in $High$ by the following process, until $High = \emptyset$.

2-1.. Select PM h attaining $\max\{util_i \mid i \in High\}$. Delete h from $High$.

2-2.. Let $V_h^c = \{k \in V_h^0 \mid vc_k \geqslant oc_h\}$, where $oc_h = \max\{0, \sum_{k \in V_h} vc_k - c_h\}$, and $V_h^m = \{k \in V_h^0 \mid vm_k \geqslant om_h\}$, where $om_h = \max\{0, \sum_{k \in V_h} vm_k - m_h\}$. For each $k \in V_h^0$, calculate

$$diff_k = \frac{|vc_k - oc_h|}{c_h} + \frac{|vm_k - om_h|}{m_h}. \tag{25}$$

(a). If $V_h^c \cap V_h^m \neq \emptyset$, then choose VM k' attaining $\min\{diff_k \mid k \in V_h^c \cap V_h^m\}$.
(b). Else, if $V_h^m \neq \emptyset$, then choose VM k' attaining $\min\{diff_k \mid k \in V_h^m\}$.
(c). Else, if $V_h^c \neq \emptyset$, then choose VM k' attaining $\min\{diff_k \mid k \in V_h^c\}$.
(d). Else, choose VM k' attaining $\min\{diff_k \mid k \in V_h^0\}$.

2-3.. Find PM l attaining $\max\{util_i \mid i \in Low,\ \sum_{k \in V_i^0} vc_k + vc_{k'} \leqslant c_i,\ \sum_{k \in V_i^0} vm_k + vm_{k'} \leqslant m_i\}$. If l exists, delete l from Low, and allocate $V_h = V_h^0 \setminus \{k'\}$ and $V_l = V_l^0 \cup \{k'\}$. If l does not exist, keep $V_h = V_h^0$.

Step 3.. Migrate one VM from each PM in Low by the following process, until $Low = \emptyset$.

3-1.. Select PM h attaining $\min\{util_i \mid i \in Low\}$. Delete h from Low.

3-2.. For each $i \in Low$, let $add_i = \{k' \in V_h^0 \mid vc_k + vc_{k'} \leqslant c_i,\ \sum_{k \in V_i^0} vm_k + vm_{k'} \leqslant m_i\}$. Find a pair of a PM l and a VM k' attaining $\max\{\tilde{c}(h,i,k) \mid i \in Low, k \in add_i\}$. If l exists, delete l from Low, and allocate $V_h = V_h^0 \setminus \{k'\}$ and $V_l = V_l^0 \cup \{k'\}$. If l does not exist, keep $V_h = V_h^0$.

Finally, we discuss the time complexity of GREEDY. Step 1 runs in $\mathrm{O}(|P| \log |P|)$ time, since it sorts P by $util_i$. Thus, Steps 2-1 and 3-1 can be performed in $\mathrm{O}(1)$ time. Step 2-2 and Step 2-3 need $\mathrm{O}(|V_h^0|)$ time and $\mathrm{O}(|Low|)$ time, respectively, if $cost_i(z)$ can be computed in constant time for any $i \in P$ and z. Thus, the complexity of Step 2 is $\mathrm{O}(|High|(|V| + |Low|))$. Since Step 3-2 searches for a pair of elements in $Low \times V_h^0$, Step 3 needs $\mathrm{O}(\sum_{h \in Low}(|Low||V_h^0|)) = \mathrm{O}(|Low| \cdot |V|)$. Thus, the total time complexity of GREEDY is $\mathrm{O}(|P|^2 + |P||V|)$.

5. EVALUATION EXPERIMENTS

We investigate the performance and the feasibility of virtual machine packing algorithms. First, we compare energy consumption, degradation of user experiences and calculation times required for each packing decision under artificial simulation environment. Then, we investigate how much electricity power can be reduced by using the packing algorithms under a realistic simulation environment, based on trace data from the T2K-Tsukuba supercomputer system [?] and a sophisticated energy consumption model.

5.1 Virtual machine packing algorithms and their implementations

The experiments evaluate the performance of MBA, GREEDY and IP. The proposed algorithms, MBA and GREEDY, were coded in Python. In Step 2 of MBA, we use the $\mathrm{O}(|P|^3)$ algorithm in [Galil 1986], which is coded by J. van Rantwijk[1], by modifying to return a U-matching. IP solves VMPP-IP by commercial solver CPLEX (IBM ILOG CPLEX Optimization Studio_Academic V12.3 64bit). In our implementation, we have been using CPLEX from Python library pulp [2]. If the solver does not return an optimal solution within 600 seconds, then we use the best solution among the feasible solutions found by it.

While GREEDY reallocates virtual machines to minimize the total power consumption although VMPP is infeasible, MBA and IP remains the initial allocation when the problem is infeasible. We replace constraints (10) and (11) in VMPP-IP with

$$\sum_{k \in V_i} vm_k x_{ik} - m_i \le m_i p_i,\ \forall i \in P, \tag{26}$$

$$\sum_{k \in V_i} vc_k x_{ik} - c_i \le c_i q_i,\ \forall i \in P, \tag{27}$$

by using two nonnegative decision variables p_i, q_i representing the excess amount of memory and CPU usage, respectively, in order to reallocate even if a problem is infeasible. Also we substitute

$$\sum_{i \in P}(cost_i(\sum_{k \in V_i} vc_k x_{ik}) + base_i y_i) + \sum_{i \in P} p_i + \sum_{i \in P} q_i \tag{28}$$

for the objective function (9) of VMPP-IP. It is clear that the modified integer programming problem is relaxation of VMPP-IP. To solve this new integer programming problem is denoted by IPr, which also use CPLEX with the same settings as IP.

Also MBA is modified to reallocate virtual machines in order to reduce the excess amount of memory and CPU usage even if VMPP is infeasible. This modified MBA, denoted by MBA-mod, treats the performance degradation of the user experience as the weight of each arc of the graph. The degradation of the user experience of physical machine i is formulated by

$$deg(V_i) = \frac{\max\{0, \sum_{k \in V_i} vc_k - c_i\}}{c_i} + \frac{\max\{0, \sum_{k \in V_i} vm_k - m_i\}}{m_i}, \tag{29}$$

and the total performance degradation is given by $\sum_{i \in P} deg(V_i)$. By $\hat{c}(i,j,k)$, a reduced performance degradation is denoted when virtual machine k is reallocated from $i \in P$ to $j \in P$. Then $\hat{c}(i,j,k)$ is given by

$$deg(V_i^0) - deg(V_i^0 \setminus \{k\}) + deg(V_j^0) - deg(V_j^0 \cup \{k\}). \tag{30}$$

[1] http://jorisvr.nl/maximummatching.html, Jan. 6, 2013 access.

[2] http://code.google.com/p/pulp-or/, Jan. 6, 2013 access.

Table I: Condition of parameters

Parameter	Condition
Algorithm	MBA, GREEDY, IP(CPLEX)
# of physical machines and virtual machine	50, 100, 150 200, 250, 300
# of rounds	200
CPU capacity of the physical machine	1.0
Memory capacity of the physical machine	4096 [MB]
Avg. / max of CPU usage of virtual machine	0.5 / 1.0
Avg. / max of memory usage of virtual machine	2048 / 4096 [MB]
Fluctuation of virtual machine's resource usage	Sine wave (Phase: uniform random, Avg. amptitude: $1/12\pi$, uniform random)
$base_i$ / $cost_i(z)$	53.0 / $47.0z$

A new weight $c'(i,j)$ of each arc $(i,j) \in A$ is defined by $\max\{\hat{c}(i,j,k) \mid k \in V_i^0\}$. MBA-mod computes a maximum weight matching on (P, A) with a new weight c' to obtains a new allocation $\{V^i\}_{i \in P}$, when VMPP is infeasible.

5.2 Basic performance of the packing algorithms

We evaluate the basic performance of each proposed algorithm under the following numerical simulation experiment. A virtual machine packing system monitors virtual machine resource utilization on physical machines, makes a virtual machine rearrangement plan by using a virtual machine packing algorithm, and then migration of selected virtual machines, periodically. The number of virtual machines is fixed during a simulation and each virtual machine's memory and CPU usage will vary every time step. Each experiment runs for two hundred time steps. Table I shows conditions of the experiment. Suppose that the number of physical machines and virtual machines are equal and the size is set to 50, 100, 150, 200, 250 and 300. The CPU capacity (the number of cores × upper bound of the usage ratio) is set to 1.0 and the memory capacity is set to 4096 [MB]. The average and maximum usage amounts of CPU and memory of virtual machine is 0.5 / 1.0 and 2048 / 4096 [MB], respectively. We employ the Sine curve to represent the fluctuation of each virtual machine's resource usage. This fluctuation model is a more strict condition compared with real operation, since resource usage generated by the Sine curve fluctuates every time step. The parameters of the power consumption were decided to be $base_i = 53.0$, $cost_i(z) = 47.0z$ for each $i \in P$ based on [Hirofuchi et al. 2011a]. Every physical machine has initially one virtual machine individually at random. We conduct the simulations on a MacOSX 10.7 64bit with an Intel Core i7 3.4GHz, 8 Cores and 16GB of RAM.

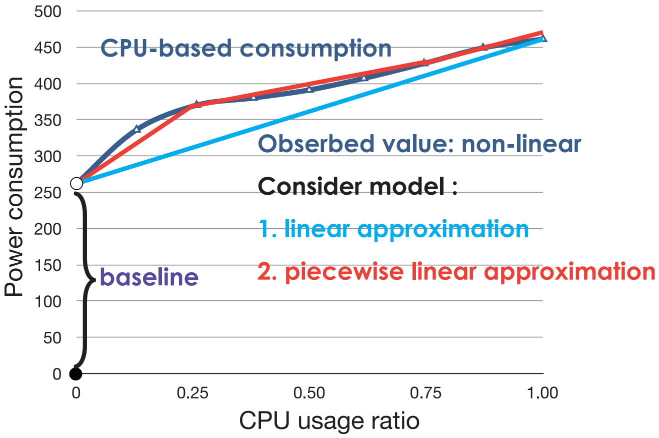
5.3 Feasibility of the packing algorithms

We compare the algorithms by using real trace data from the T2K-Tsukuba supercomputer in order to show the feasibility. In these experiments, idle servers are switched off in all cases.

Our simulation assumes that the packing system operates the monitoring of the virtual machines, as well as the planning and reallocation on a routine schedule, as same as in subsection 5.2. Although the simulation in subsection 5.2 supposes that every virtual machine is operating until end-of rounds, this simulation supposes that each virtual machine's operating time is decided by the trace data from the T2K-Tsukuba, i.e. each virtual machine is not required to operate continuously among considering rounds. The T2K-Tsukuba consists of 648 nodes with total theoretical peak capability of 95Tflops. Each node consists of AMD quad-core Opteron 2.3GHz × 4 CPUs and 32GB memory. The T2K-Tsukuba has been utilized by a lot of users for scientific computation since 2008. Because the power consumption model of the T2K-Tsukuba is not open, we assume each node of the T2K-Tsukuba is configured using Intel quad-core Xeon W5590 3.33GHz × 2 CPUs, 48GB of memory in the simulation. Table II shows extensive details

Table II: Details of our model's physical machine.

CPU	Intel(R) Xeon(R) W5590 3.33GHz
# CPU per node	2
# Cores per CPU	4
Memory per node	48GB
Operating System	CentOS 5.5 x86_64
Storage (ioDrive)	Fusion-io ioDrive Duo 320GB
Storage (SAS HDD)	Fujitsu MBA3147RC 147GB/15000rpm
Network Interface	Mellanox ConnextX-II 10G Adapter
Network Switch	Cisco Nexus 5010



Power consumption of observed value, linear approximation and piecewise linear approximation.

of the physical machine used in the simulation model. Figure 3 shows the power consumption measured by a physical machine of Table II. A nonlinear power consumption function is observed, hence we assume a linear function model and a piecewise linear function model of the power consumption function and evaluate our algorithms by using these models.

In this evaluation, the number of physical machines and the maximum number of virtual machines in a round is 648. We set the experiment rounds are 10080 rounds in the linear model and 1000 rounds in the piecewise linear model. For each physical machine $i(\in P)$, we set $base_i = 263$ as a baseline power consumption and $cost_i(z) = 175z$ as a power consumption function of the linear model. Also for each physical machine $i(\in P)$, we set a power consumption function of the piecewise linear model by

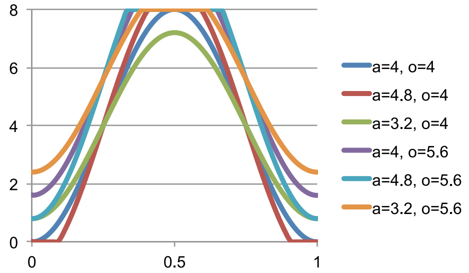
$$cost_i(z) = \begin{cases} 432z & 0 < z \leq 0.25, \\ 115z + 108 & 0.25 < z \leq 0.75, \\ 134z + (108 + 86.25) & 0.75 < z, \\ 0 & z \leq 0. \end{cases}$$

The trace data of the T2K-Tsukuba consists of submit times of jobs, the number of cores, elapsed times and maximum memory usage. We assume every job included parallel jobs as virtual machines and assume every virtual machine can be migrated individually. First of all, we use two weeks of job data in the trace data to make an allocation plan using a Gantt chart. The elapsed time is considered as the life-span of each virtual machine based on the job set-up time decided using the Gantt chart. We regard the second week job data induced by the Gantt chart as an input of the experiment and apply the proposed algorithms to the input with respect to each minute.

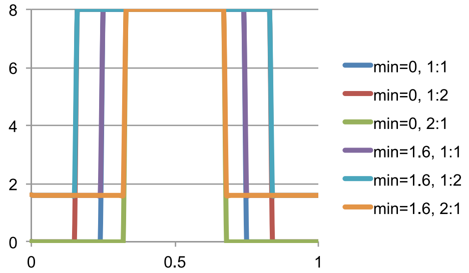
The fluctuation of each job's memory usage amount is fixed by its maximum memory usage amount. We vary only CPU usage amount. The fluctuation of the CPU usage amount is represented by the six patterns of the Sine curve and square wave showed in Figures 4 and 5. Each pattern's frequency is 25 or 50, i.e., each virtual machine's cycle is either elapsed time / 25 or

Table III: Parameters of fluctuation models.

Model	Parameters	Values
Sine curve	Offset	4, 5.6
	Frequency	25, 50
	Amplitude	3.2, 4, 4.8
Square wave	Minimum value	0,1.6
	Frequency	25, 50
	Duty cycle (off : on)	1:1, 1:2, 2:1



Sine wave variation model



Square wave variation model

Table IV: Comparison of average power consumption and computation time per one round.

# of PM (= # of VM)	Avg. of power consumption (watt) FIX	 MBA	 IP	 GREEDY	 IPr	 MBA-mod
50	3814.716	1951.501	2817.311	3076.691	2854.433	2527.346
100	7623.835	3959.415	5925.185	6009.985	5601.287	5044.060
150	11469.435	5841.630	9395.015	8947.165	8229.118	7614.215
200	15259.594	7752.674	11755.764	11814.064	10842.380	10124.424
250	19055.052	9579.447	14552.893	14657.112	13023.482	12636.487
300	22840.991	11423.201	17479.776	17541.521	15976.583	15128.166

# of PM (= # of VM)	Ave. of comp. time (sec.) MBA	 IP	 GREEDY	 IPr	 MBA-mod
50	0.018	0.659	0.008	0.479	0.051
100	0.076	13.243	0.026	5.492	0.208
150	0.157	12.360	0.052	10.575	0.472
200	0.280	22.335	0.089	72.597	0.834
250	0.433	21.358	0.133	285.430	1.346
300	0.625	48.815	0.187	530.083	1.943

elapsed time / 50. We set the minimum cycle of the virtual machines as 10 minutes. Table III shows some parameters of twenty four fluctuation models of CPU usage amount.

6. RESULTS OF THE EVALUATION EXPERIMENTS

6.1 Result of basic performance

We compare the average power consumption, computation time and performance degradation of each problem using the proposed algorithms. Table IV shows the experimental results MBA, IP, GREEDY, IPr and MBA-mod. FIX shows the resulting values when the packing system does not reallocate any VM at each round. MBA and IP employ the previous allocation when the problem of the current round is infeasible.

Table V: Results of the average power consumption and computation time in large size problems.

# of PM	Average power consumption (watt)			Average computation time (sec.)	
(= # of VM)	FIX	MBA	GREEDY	MBA	GREEDY
400	30507.199	15273.144	23375.519	1.145	0.314
500	38130.399	19116.119	29155.379	1.890	0.480
600	45808.771	22914.891	34862.151	2.771	0.651
700	53408.342	26723.107	40640.907	3.892	0.870
800	60994.286	30621.311	46263.201	5.090	1.132
900	68603.518	34603.753	52057.448	6.719	1.431
1000	76260.032	38524.827	57715.597	8.499	1.718

Table VI: Average performance degradation of user experience.

# of PM	Average performance degradation				
(= # of VM)	MBA	IP	GREEDY	IPr	MBA-mod
50	1.34	8.05	0.17	0.56	0.91
100	1.21	18.39	0.40	1.11	0.93
150	1.16	23.47	0.70	1.59	0.93
200	0.65	34.66	1.07	2.51	0.57
250	0.41	36.68	1.37	3.04	0.36
300	0.61	48.11	1.56	7.67	0.49

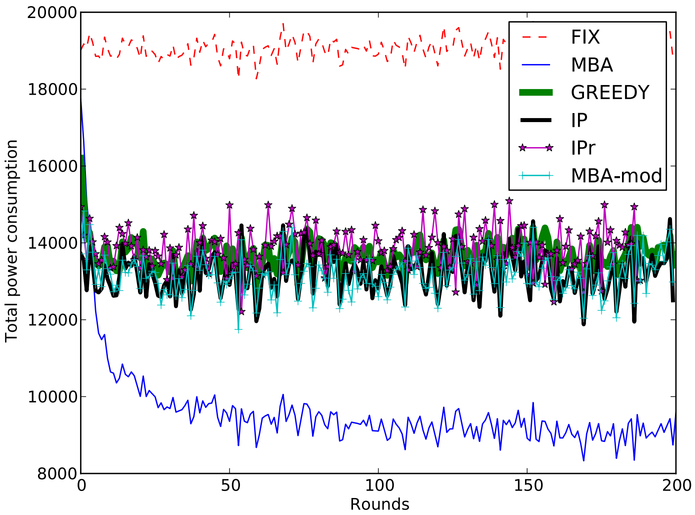
In Table IV, MBA establishes the lowest average power consumption in almost all instances. For some instances, GREEDY obtains lower average power consumption than IP. The results shows each algorithm reduces power consumption between 18% to 50%, compared with FIX.

While both of MBA and IP solve VMPP exactly, their average power consumptions are different. One of the reasons is that MBA and IP obtain a different optimal solution at some rounds in the simulation. The other reason is that IP employs a temporary solution if CPLEX does not obtain the optimal solution of the problem within 600 seconds. Since IPr tries to reallocate VMs to decrease total power consumption when an instance is infeasible, IPr obtains better power consumption compared with IP. MBA-mod reduces the power consumption more than IPr, however, MBA has larger decreasing ratio than MBA-mod. GREEDY, MBA and MBA-mod, which are polynomial time algorithms, run in shorter computation time than IP dramatically. Especialy, GREEDY has the shortest average computation time. The computational time of MBA-mod is longer than MBA, since MBA-mod computes two distinct matchings in infeasible case.

Fig. 6 shows the comparison of power consumption fluctuation in each round at a problem size of 250. The horizontal axis shows the round and the vertical axis shows the power consumption. Each algorithm shows the same trends in changing the power consumption at almost all rounds.

Table V shows the results of average power consumption and computation times obtained by MBA and GREEDY in large size problems. MBA's average computation time is about 8.5 seconds at one round when the problem size is 1000. The results show MBA is strong enough to use in practices although GREEDY has a shorter average computation time.

Finally we compere a performance degradation of the user experience. Although there are many factors involved in the loss of user experience, we employ the ratio of overbalance of CPU or memory usage, defined by $\sum_{i \in P} deg(V_i)$. Table VI evaluates the average performance degradation by summing each PM's degradation per one round. The performance degradation of each of IP, GREEDY, and IPr is proportional to the increase in the size of the problem. On the other hand, MBA-mod shows the smallest average performance degradation at the problem size 250. IPr shows smaller performance degradation than IP, which suggests the importance of the reallocation for the infeasible instance.



The power consumption fluctuation of FIX, MBA, MBA-mod, GREEDY, IP and IPr in each round at a problem size of 250.

6.2 Results of feasibility testing of the proposed algorithms

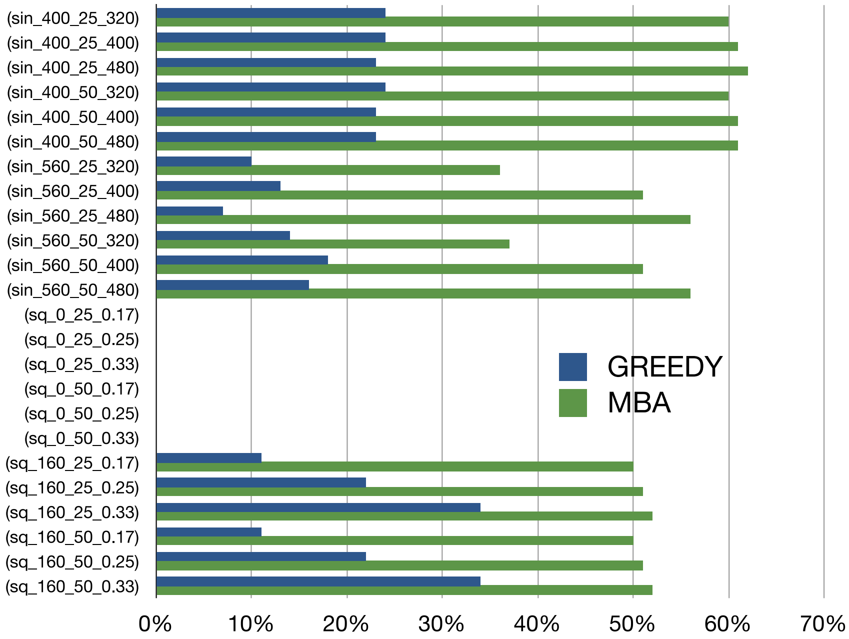
We compare the average power consumption, computation time and performance degeneration of each problem using the proposed algorithms. Table VII shows the average value of power consumption, computation time, and performance degradation of sine curve and square wave model in the T2K-Tsukuba trace data experiment.

First of all, we show the results of linear power consumption model. In these results, MBA and GREEDY reduce the power consumption compared with FIX. MBA does not establish smaller power consumption than GREEDY, but the performance degradation does not occur at every problem level, even if GREEDY shows greater performance degradation. Since MBA sometimes utilizes new physical machines for a new input virtual machines, it is hard for MBA to reduce total power consumption in the T2K-Tsukuba trace data, in which the number of virtual machines changes dynamically, although the performance degradation does not occur. Considering the differences of load models, with the square wave model it is difficult to reduce total power consumption compared with the sin curve. There was no influence from difference between the average value of load models, however, a large amplitude showed better performance than a small one.

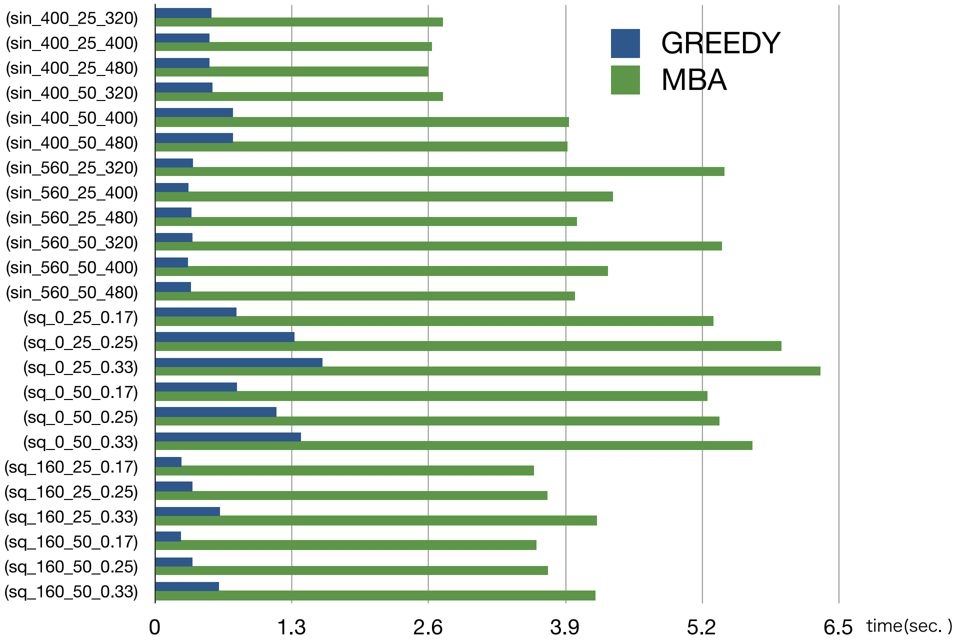
Next, we show results of the piecewise linear model. In the result of average reducing ratios of power consumption (Figure 7), MBA and GREEDY reduce the power consumption compared with FIX. MBA reduces more power consumption than GREEDY. Since a reduced cost $c(i,j)$ is positive when $|V_i^0| = 1$ in linear model, MBA does not reduce more power consumption than GREEDY. In piecewise linear model, the reduced cost $c(i,j)$ is almost positive, since a power consumption function is piecewise linear. Hence the performance degradation is occurred by both MBA and GREEDY (Figure 9). The result of computing times shows in Figure 8. In this result, GREEDY is faster than MBA in both load models of VM resources.

We summarize our proposed algorithms in Table VIII. MBA reduces power consumption by 0%-62% in piecewise linear consumption model, and GREEDY reduces power consumption by 0%-35% in piecewise linear consumption model.

To minimize the system down time of each virtual machine, the proposed algorithms are applied to the packing system as premises for a fast post-copy live migration technique, however, it is also possible to apply these algorithms to a pre-copy live migration packing system. Pre-copy live migration can operate at high-speeds transporting a memory image and reduction down time using the proposed algorithms, since the transport time of the memory image is influenced by



Average reducing ratio of power consumption of piecewise linear model.



Average computing times of piecewise linear model.

Table VII: Average power consumption, computation time and performance degradation per one round.

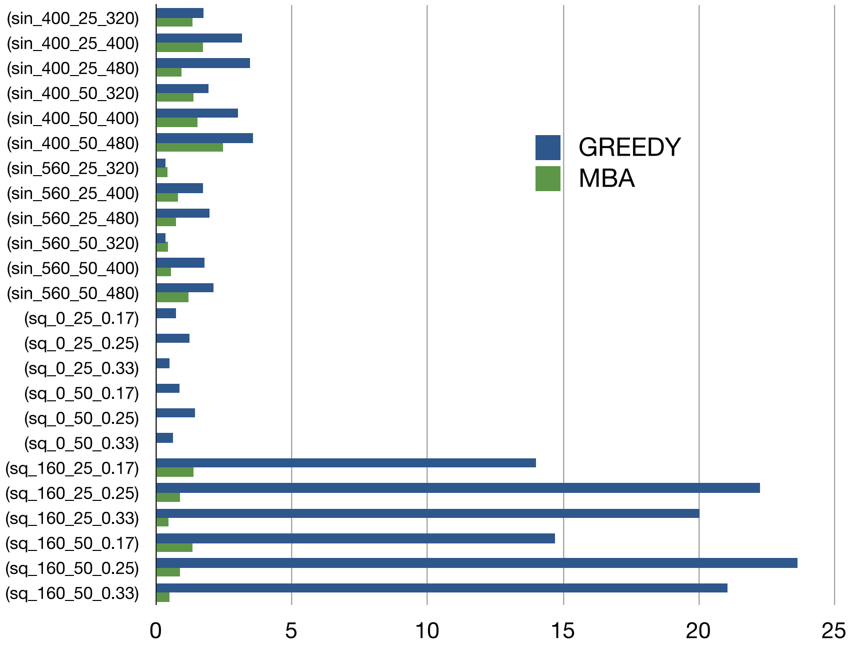
Problem	Avg. of power consumption (watt)			Avg. of comp. times (sec.)		Ave. of perf. deg.	
	FIX	MBA	GREEDY	MBA	GREEDY	MBA	GREEDY
(sin, 400, 25, 320)	181679.055	170924.824	156143.078	3.431	0.031	0	54.67
(sin, 400, 25, 400)	181811.637	172489.179	153769.755	3.530	0.029	0	61.89
(sin, 400, 25, 480)	181832.904	172358.907	150575.507	3.539	0.024	0	64.94
(sin, 400, 50, 320)	181688.048	171644.548	154889.590	3.441	0.028	0	60.08
(sin, 400, 50, 400)	181822.945	171594.104	153043.758	3.487	0.026	0	67.11
(sin, 400, 50, 480)	181842.927	171606.529	150728.567	3.505	0.023	0	72.65
(sin, 560, 25, 320)	198388.229	190166.168	177904.317	3.262	0.045	0	52.26
(sin, 560, 25, 400)	196253.132	188687.078	172384.297	3.321	0.037	0	56.42
(sin, 560, 25, 480)	193972.238	185755.649	167734.479	3.373	0.032	0	60.09
(sin, 560, 50, 320)	198397.941	191952.787	177266.118	3.305	0.042	0	55.93
(sin, 560, 50, 400)	196254.524	188367.950	171915.746	3.311	0.035	0	61.96
(sin, 560, 50, 480)	193979.134	187563.264	167533.284	3.424	0.031	0	67.28
(sq, 0, 25, 0.17)	198491.906	193072.944	171236.815	3.515	0.011	0	71.25
(sq, 0, 25, 0.25)	181827.965	171453.007	147876.866	3.622	0.013	0	76.05
(sq, 0, 25, 0.33)	165182.704	151929.562	123296.289	3.819	0.015	0	72.69
(sq, 0, 50, 0.17)	198660.604	190419.429	171166.550	3.426	0.011	0	79.01
(sq, 0, 50, 0.25)	181825.812	171855.850	147909.956	3.738	0.013	0	82.20
(sq, 0, 50, 0.33)	165046.420	153977.773	123724.216	3.900	0.015	0	77.54
(sq, 160, 25, 0.17)	204063.156	197787.754	179979.072	3.178	0.024	0	70.23
(sq, 160, 25, 0.25)	190732.003	181948.732	157342.427	3.278	0.021	0	76.53
(sq, 160, 25, 0.33)	177415.795	166977.972	138447.890	3.237	0.019	0	71.52
(sq, 160, 50, 0.17)	204198.114	198608.988	180396.688	3.007	0.024	0	78.16
(sq, 160, 50, 0.25)	190730.281	182874.520	157954.966	3.083	0.021	0	87.33
(sq, 160, 50, 0.33)	177306.767	165398.584	138592.703	3.169	0.019	0	79.38

Table VIII: Summary of proposed algorithms.

	MBA		GREEDY	
	Linear	Piecewise linear	Linear	Piecewise linear
Power Reduction	3%-7%	0%-62%	10%-15%	0%-35%
Comp. time	$O(\lvert P\rvert^3 + \lvert V\rvert\lvert P\rvert)$		$O(\lvert P\rvert^2 + \lvert V\rvert\lvert P\rvert)$	
Perform. deg.	None	Small	Large	Middle

the down times of virtual machines.

Finally, we consider a situation where we have relaxed the restriction on the number of migrations of each PM per each one time step. GREEDY can be revised for this situation flexibly. It can be modified so as to delete PMs where reach the upper limit of the number of migrations (from or to) from the set of candidates for reallocation. Unfortunately, MBA can not support this situation, because Theorem 4.1 is based on the assumption that the number of migrations of each PM is at most one. Indeed, if VMPP permits two or more migrations of each PM, we can show this problem is $\mathcal{NP}$-hard by reducing it to a 3-partition problem. Let S be a set with $|S| = 3q$, where q is a positive integer. Each element $e \in S$ has a size $s(e)$ such that $B/4 < s(e) < B/2$ and $\sum_{e \in S} s(e) = 3B$ for a given positive integer B. The 3-partition problem finds a partition of S into m subsets $S_1, S_2, \ldots, S_m$ where $\sum_{e \in S_i} s(e) = B$ for $i = 1, \ldots, m$. This problem is known as $\mathcal{NP}$-hard [Garey and Johnson 2000]. From an instance of the 3-partition problem, we construct an instance of VMPP permitting at most two migrations, as follows. Each element $e \in S$ corresponds to a VM. Thus, a set of VM, V is given by S. Each VM e has $vm_e = vc_e = s(e)$. In addition, there are $3m$ PMs, where each PM i has $m_i = c_i = B$, $base_i = 1$ and $cost_i(z) = 0$ for any z. Initially, every VM is allocated to a distinct machine. We can see that the 3-partition



Average performance degradation of piecewise linear model.

problem has a solution $S_1, S_2, \ldots, S_m$, where $S_i = \{e_{i_1}, e_{i_2}, e_{i_3}\}$, if and only if VMs corresponding to e_{i_2} and e_{i_3} migrate into the machine assigned e_{i_1}. In this solution, the objective function of the VM packing problem is exactly m.

THEOREM 6.1. *The VM packing problem permitting more than one migration to each machine, is $\mathcal{NP}$-hard. It remains $\mathcal{NP}$-hard, even if the number of migrations for each machine per one round is bounded by at most two.*

From Theorem 6.1, The VM packing problem permitting more than one migrations has no efficient algorithm such as MBA, unless $\mathcal{P} = \mathcal{NP}$. Therefore, for this situation, we should use an integer programming solver, or heuristic approaches like GREEDY.

7. CONCLUDING REMARKS

This paper proposed two VM packing algorithms, MBA and GREEDY. Evaluation experiments show that these algorithms are valuable for VM packing systems. We consider the fast post-copy migration system. However, our algorithm can apply to the pre-copy migration system.

We consider a situation where we have relaxed the restriction on the number of migrations of each PM per each one time step. GREEDY can be revised for this situation flexibly. It can be modified so as to delete PMs where reach the upper bound of the number of migrations from the set of candidates for reallocation. Unfortunately, MBA can not support this situation. Indeed, if VMPP permits two or more migrations of each PM, we can show this problem is $\mathcal{NP}$-hard by reducing it to a 3-partition problem.

In future work, we interest to extend the virtual machine packing problem to a multiperiod model in order to investigate the efficiency of global power consumption. We also interest to investigate the performance degradation of real applications together with CPU and memory performance in order to verify the practicality of the proposed algorithms by applying them to a real VM packing system. We did not treat an influence of recovering from the standby mode in our experiments. Thus, to consider the performance degradation of recovering from the standby mode might give more realistic evaluation. Moreover to evaluate relation between network overhead and performance degradation, we improve our algorithm such as dealing with network topology and characteristic of applications.

ACKNOWLEDGMENT

We thank Prof. Taisuke Boku, Prof. Mitsuhisa Sato and Center for Computational Science of University of Tsukuba for willingly and agreeably preparing the T2K Tsukuba trace data. We thank Dr. Kiyo Ishii for valuable discussion.

REFERENCES

Beloglazov, A., Abawajy, J., and Buyya, R. 2012. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer System 28*, 755–768.

Calheiros, R. N., Ranjan, R., Beloglazov, A., Rose, C. A. F. D., and Buyya, R. 2011. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice Experiment 41*, 23–50.

Chen, M., Zhang, H., Su, Y., Wang, X., Jiang, G., and Yoshihira, K. 2011. Effective vm sizing in virtualized data centers. In *Proceedings of IFIP/IEEE International Symposium on Integrated Network Management*. 594–601.

Edmonds, J. 1965. Paths, trees, and flowers. *Canadian Journal of Mathematics 17*, 449–467.

Fleszar, K. and Hindi, K. S. 2002. New heuristics for one-dimensional bin-packing. *Computers & Operations Research 29,* 7, 821–839.

Galil, Z. 1986. Efficient algorithms for finding maximum matching in graphs. *ACM Computing Surveys 18,* 1, 23–38.

Garey, M. R. and Johnson, D. S. 2000. *Computers and Intractability - A Guide to The Theory of NP-Completeness, Books in the Mathematical Sciences, 22nd printing.* W. H. Freeman and Company.

Hirofuchi, T., Nakada, H., Itoh, S., and Sekiguchi, S. 2010. Enabling instantaneous relocation of virtual machines with a lightweight vmm extension. In *Proceedings of IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing.* 73–83.

Hirofuchi, T., Nakada, H., Itoh, S., and Sekiguchi, S. 2011a. Making vm consolidation more energy-efficient by postcopy live migration. In *Proceedings of the 2nd International Conference on Cloud Computing, GRIDs, and Virtualization.* 195–204.

Hirofuchi, T., Nakada, H., Itoh, S., and Sekiguchi, S. 2011b. Reactive consolidation of virtual machines enabled by postcopy live migration. In *Proceedings of the 5th International Workshop on Virtualization Technologies in Distributed Computing.* 11–18.

Korte, B. H. and Vygen, J. 2004. *Combinatorial optimization: theory and algorithms.* Springer-Verlag, Berlin Heidelberg.

Li, B., Li, J., Huai, J., Wo, T., Li, Q., and Zhong, L. 2009. Enacloud: An energy-saving application live placement approach for cloud computing environments. In *Proceedings of IEEE International Conference on Cloud Computing.* 17–24.

Nakada, H., Hirofuchi, T., Ogawa, H., and Itoh, S. 2010. Toward virtual machine packing optimization based on genetic algorithm. In *LNCS.* Vol. 5518. Springer, 651–654.

Sinha, R., Purohit, N., and Diwanji, H. 2011. Energy efficient dynamic integration of thresholds for migration at cloud data centers. *International Journal of Computer Applications Special Issue on CN*, 44–49.

Takeda, S. and Takemura, T. 2010. A rank-based vm consolidation method for power saving in datacenters. *Information and Media Technologies 5,* 3, 994–1002.

Vazirani, V. V. 2001. *Approximation Algorithms.* Springer-Verlag, Berlin Heidelberg.

Verma, A., Ahuja, P., and Neogi, A. 2008. pmapper: power and migration cost aware application placement in virtualized systems. In *Proceedings of the 9th Middleware.* 243–264.

Wang, Y. and Wang, X. 2010. Power optimization with performance assurance for multi-tier applications in virtualized data centers. In *Proceedings of International Conference on Parallel Processing Workshops.* 512–519.

Satoshi Takahashi Dr. Satoshi Takahashi is an assistant professor of University of Electro-Communications. He got Ph. D in Engineering from University of Tsukuba in 2012. He worked as research fellow of JSPS in 2011. Special area of research is electronic commerce and mathematical optimization.

Atsuko Takefusa Dr. Atsuko Takefusa is a researcher of National Institute of Advanced Industrial Science and Technology. She got Ph.D in Science from Ochanomizu University. She worked as research fellow of JSPS in 2000 to 2002 and assistant professor of Ochanomizu University in 2002 to 2005. Special area of research is distributed computing, grid and cloud computing, and scheduling.

Maiko Shigeno Dr. Maiko Shigeno is an associate professor of University of Tsukuba. She got Ph.D in Science from Tokyo Institute of Technology in 1996. She worked as assistant professor of Tokyo Institute of Technology in 1995 to 1997. Special area of research is combinatorial optimization.

Hidemoto Nakada Dr. Hidemoto Nakada is a chief researcher of Grid Infraware Research Group of National Institute of Advanced Industrial Science and Technology. He got Ph.D in Engineering from University of Tokyo in 1995. He worded as visiting associate professor of Tokyo Institute of Technology in 2001 to 2005. Special area of research is Global computing and distributed computing.

Tomohiro Kudoh Dr. Tomohiro Kudoh is a deputy director of Information Technoloty Research Institute of National Institute of Advanced Industrial Science and Technology. He graduated Graduate School of Science and Engineering of Keio University in 1991. He worked as assistant professor, Lecturer and associate professor of Tokyo University of Technology until 1997. Special area of research is parallel processing and communication architecture.

Akiko Yoshise Dr. Akiko Yoshise is a Professor of University of Tsukuba. He got doctor of Engineering from Tokyo Institute of Technology in 1990. She worked as researcher of University of Tsukuba in 1990. Special area of research is algorithms for continuous optimization and mathematical modeling for service industry.