省電力化のためのマッチングに基づく仮想計算機パッキングアルゴリズム

高橋里司 † 竹房あつ子 †† 繁野麻衣子 †
中田秀基 †† 工藤知宏 †† 吉瀬章子 †

データセンターでの消費電力低減の手法として，低負荷の仮想計算機を高速にマイグレーションして集約し，使用していない物理計算機を省消費電力モードで運用する方法がある．これを実現するには，消費電力を抑えつつユーザ体験の劣化を防ぐ，負荷が変動する VM 群を現実的な時間で再配置する，再配置時のマイグレーションによる通信衝突を考慮する，という課題がある．本研究では，ユーザ体験を劣化させることなく省電力化を図る仮想計算機パッキング問題に対するマッチングベースアルゴリズムを提案，評価する．提案手法では，再配置時に 1 つの計算機が送受信する VM 数を制限し，VM と計算機の組み合わせをグラフのマッチングで表すことで，多項式時間で適切な再配置プランニングを可能にする．評価では，提案手法を 0-1 整数計画問題として定式化して最適化ソルバを用いる手法とグリーディな手法で，消費電力削減効果，計算時間，ユーザ体験の劣化について比較する．実験から，1) パッキングをしない場合より 18%から 50%の消費電力が削減できる，2) マッチングベースアルゴリズムにより，現実的な計算時間で適切な再配置を行うことができる，3) ユーザ体験は消費電力の削減効果に反比例する傾向があるが，再配置によりほぼ改善できることを示す．

Matching-based Virtual Machine Packing Algorithm for Lower Power Consumption

SATOSHI TAKAHASHI ,† ATSUKO TAKEFUSA ,†† MAIKO SHIGENO,†
HIDEMOTO NAKADA,†† TOMOHIRO KUDOH †† and AKIKO YOSHISE†

VM (Virtual Machine)-based flexible capacity management, which consolidates multiple VMs into few physical machines and other physical machines are allowed to be put in “stand-by” mode, is an effective scheme to reduce total power consumption in a datacenter. However, there have been the following issues, reconcilement of power-saving and user experience, decision of VM packing in feasible calculation time and collision avoidance of VM migration processes. In order to resolve these issues, we propose a matching-based VM packing algorithm (MBA), which enables to decide a suitable VM packing plan in polynomial time. In the experiments, we compare the proposed algorithm with 0-1 integer programming model, using optimization solvers, and a greedy algorithm. The results show that 1) the proposed algorithm MBA reduces the total power consumption by between 18% and 50% of when the packing plan does not work, 2) MBA enable to provide an optimal packing plan in feasible calculation time, and 3) there is a trade-off between the power-saving effect and the deterioration of the user experience.

1. はじめに

クラウドコンピューティングの普及により，データセンターにおける IT 機器の消費電力が今後急激に増大する事が予想されている．消費電力増大の原因の一つとして，資源割当て時に必要容量設定の困難さがあげられる．データセンター内で運用されるサービスの負荷やそのリクエストの頻度は，事前に予測が出来ないうえ，時間によって変動する．サービス要求に対する資源割当てが負荷に対して過小であると，サービス処理性能が低下してユーザ体験が劣化してしまう．よって，データセンターの設計者は資源利用のピーク時に合わせた資源割当てを行わなければならず，低負荷時には資源浪費が発生する．さらに，計算機は負荷が割当てられていない場合も稼働中の消費電力は飽和時の 5 割程度にもなることがあり，それによる電力の浪費も非常に大きい．

この問題に対して，仮想計算機（VM）の高速ライブマイグレーション技術を用いて，ユーザ体験を劣化させることなく VM を再配置して消費電力を削減する手法が提案されている[1]．低負荷時には，1 台の計算機に対して複数の VM を立ち上げ，例えば ACPI S3 のような省電力モードなどで不要な計算機の稼働を抑

† 筑波大学
University of Tsukuba
†† 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology (AIST)

えることでデータセンター内で消費される電力量を抑制する．負荷が増大した際には，計算機を復帰させてVM を再配置することでサービスの劣化を防止する．

一方で、VM の再配置手法に関して十分に議論されているとは言えない．具体的には，(1) 消費電力を抑えつつサービス劣化を防ぐこと，(2) 変動する VM の負荷に対する現実的な時間での VM の再配置，(3) 複数の VM の同時ライブマイグレーションにおける衝突回避が，技術的課題として挙げられる．(1) は，少数の計算機に VM を稼働させて消費電力量を削減したとしても，各 VM が必要とする資源が割当てられないとユーザ体験が劣化してしまうため，VM の負荷に応じて再配置をしなければならない．(2) は，各 VM の負荷は刻々と変化するため，VM と計算機の監視，再配置のプランニング，再配置の実行を繰り返し行うことが求められる．この一連の処理を現実的な時間内に実行しなければ負荷状況が変わってしまい，再配置の効果が得られない．(3) は，再配置の実行では高速ポストマイグレーション技術[1] により 1 秒程度でユーザ体験を劣化させることなく VM を移動させることができる．しかし，マイグレーション後にバックグラウンドでメモリイメージのコピーを続けるため，[メモリサイズ／帯域] 時間，送受信する計算機間で通信が行われることになる．この間に通信衝突が起こると，完全にマイグレーションが終了するまでの時間が長くなってしまう．

本研究では，ユーザ体験を劣化させることなく省電力化を行うことを目的として，VM を再配置する問題に対するマッチングベースアルゴリズム（MBA）を提案，評価する．提案手法では，計算機の CPU 使用率とメモリ量を VM への割当て対象の資源とし，1 つの計算機が送受信する VM の数を制限して再配置を試みる．この VM の再配置をグラフのマッチングで表すことで，計算機数と，VM 数の多項式時間で適切な再配置プランニングを可能にする．また，比較対象として 0-1 整数計画問題として定式化して最適化ソルバを用いる手法（IP），グリーディな手法 (GREEDY) を提案する[4]．評価では，各 VM の CPU 使用量とメモリ使用量をタイムステップごとに変動させ，各タイムステップにおける総消費電力量，プランニングに要する計算時間，ユーザ体験の劣化率を調査する．評価から，VM パッキング手法を用いることで，1) パッキングをしない場合より 17%から 31%の消費電力が削減できる，2)MBA により，現実的な計算時間で適切な再配置を行うことができる，3) ユーザ体験は消費電力の削減効果に反比例する傾向があるが，再配置によりほぼ改善できることを示す．

2. 関連研究

消費電力削減を目的とした VM の再配置手法は複数提案されている．

文献[2),3)] では，仮想計算機を割当てる計算機の台数とマイグレーション数の低減を目的として，仮想計算機割当て問題を 0-1 整数計画問題として定式化し，そのヒューリスティック解法として，遺伝的アルゴリズムおよびグリーディアルゴリズムを提案している．しかしながら，0-1 整数計画問題を分枝限定法で求解する手法や遺伝的アルゴリズムでは，求解までの計算時間が長く，実用化という面では課題がある．また，グリーディアルゴリズムでは，計算時間は短いが，マイグレーションの総数を考慮するのが困難であるという課題がある．

Entropy[15] は，本研究同様に制約によって仮想計算機パッキング問題を解いている．Entropy では，VM 群を資源要求によっていくつかのクラスに分類することで，大きく計算時間を短縮している．しかし，本研究が想定する環境では，VM の CPU 使用率は相互に独立に変動するものであり，VM 群が少数のクラスに分類可能であるという前提は受け入れがたい．

Stillwell ら[16] は，VM マイグレーションコストを考慮しつつ性能劣化を軽減するクラスタスケジューリングアルゴリズムを提案している．彼らの提案手法では，スタベーションを考慮した優先度付けを行い，従来のヒューリスティック手法に対してサスペンドやマイグレーションを適用し，高優先度ジョブを実行する．優先度を考慮する手法として興味深いが，消費電力を削減するものではない．

3. 仮想計算機パッキング

データセンターにおいて，複数の物理計算機上に多数の仮想計算機 (VM) を配置して消費電力量を低減する事を考える．個々の VM は物理計算機だけでなくデータセンター内の様々な資源を利用するが，ここでは，物理計算機のメモリ資源と計算資源 (CPU 量) を考える．VM が使用するメモリ量は比較的一定で安定しているが，CPU 使用量は時間に対して大きく変動する．特に，VM として，Web サーバを考えると，長時間の低負荷状態の合間に瞬間的に高負荷となることが一般的である．

3.1 VM パッキングシステム

このような VM を低消費電力で多数運用するための方法として，広渕らにより VM パッキングシステ

ムが提案されている[5]．パッキングシステムでは，各 VM の負荷状況を監視し，その結果を用いて再配置のプランニング，VM のマイグレーションを行う．低負荷時には，メモリ容量の許す限り多数の VM を物理計算機上に収納する．このとき使用していない物理計算機は ACPI S3 などの省消費電力モードで待機させる．ACPI S3 では，無負荷運用時 70w 程度の消費電力を 5-7w 程度の消費電力で待機させることができる．特定の VM の負荷が上昇した場合，省消費電力モードで待機させておいた物理計算機を復帰させ VM をライブマイグレーション技術によって移動させ CPU 使用量の上昇に対応する．

高速ポストライブマイグレーション技術[5),6)] を用いると，VM のマイグレーション所要時間は 1 秒程度となる．ACPI S3 からの復帰は 1-2 秒程度であり，総計で 2-3 秒程度で資源の再割当てを行うことができる．ただし，高速ポストライブマイグレーションでは VM をマイグレーションした後，バックグラウンドでメモリのコピーを続けるため，マイグレーションにおける通信衝突を考慮する必要がある。

3.2 仮想計算機パッキング問題

複数の物理計算機上に多数の VM が配置されている状態で，VM の負荷状況が変化したときに，消費電力が最小となるように，稼働する物理計算機を決定して VM の再配置を行う問題を，ここでは仮想計算機パッキング問題と呼ぶ．まず，記号の定義を与える．P を物理計算機の集合，V を仮想計算機 (VM) の集合とする．各物理計算機 $i \in P$ にはメモリと CPU のキャパシティ m_i, c_i が与えられている．また，各 VM $k \in V$ のメモリ使用量と CPU 使用量を vm_k, vc_k とする．さらに，CPU 使用量 z の時の物理計算機 $i(\in P)$ の電力使用量を $cost_i(z)$，物理計算機 i の基本電力使用量を $base_i$ とする．x^0_{ik} は VM $k(\in V)$ の初期配置を表し，k が物理計算機 $i(\in P)$ に配置されているときに 1，そうでないときに 0 の値をとる．決定変数は $x_{ik}(i \in P, k \in V)$ と $y_i(i \in P)$ の 2 種類の 0-1 変数で，それぞれ，

$$x_{ik} = \begin{cases} 1 & \text{(物理計算機 } i \text{ に VM } k \text{ を配置)} \\ 0 & \text{(そうでないとき)} \end{cases} \quad (1)$$

$$y_i = \begin{cases} 1 & \text{(物理計算機 } i \text{ を稼働)} \\ 0 & \text{(そうでないとき)} \end{cases} \quad (2)$$

を表す．このとき，仮想計算機パッキング問題 (VMPP) は以下のように定式化される．

$$\text{最小化} \sum_{i \in P}(cost_i(\sum_{k \in V} vc_k x_{ik}) + base_i y_i) \quad (3)$$

$$\text{条件} \sum_{i \in P} x_{ik} = 1, \forall k \in V \quad (4)$$

$$\sum_{k \in V} vm_k x_{ik} \leq m_i, \forall i \in P \quad (5)$$

$$\sum_{k \in V} vc_k x_{ik} \leq c_i, \forall i \in P \quad (6)$$

$$\sum_{k \in V} x_{ik} \leq |V| \cdot y_i, \forall i \in P \quad (7)$$

$$\sum_{k \in V} |x_{ik} - x^0_{ik}| \leq 1, \forall i \in P \quad (8)$$

$$x_{ik} \in \{0, 1\}, \forall i \in P, \forall k \in V \quad (9)$$

$$y_i \in \{0, 1\}, \forall i \in P \quad (10)$$

文献[7)] では，メモリの使用率の変動による使用電力量の変動と，物理計算機間の通信によるネットワーク機器の使用電力量の変動はほぼ見られないため目的関数 (3) は，CPU の使用率に応じて発生する電力量と，物理計算機を稼働しているときに発生する電力量の総和で表す．制約 (4) は，各 VM は必ずいずれかの物理計算機に割当てられなければならないことを表す．制約 (5) と (6) は，それぞれ割当てられた仮想計算機のメモリと CPU の上限を表す．制約 (7) は，1 台以上 VM を保持している物理計算機は必ず稼働していなければならないという条件である．制約 (8) は各物理計算機に対するマイグレーションの回数制約を表す．ここで，通信衝突による性能劣化を防ぐため，各物理計算機での VM の送受信は 1 回とする．

4. 仮想計算機パッキング問題の解法

本研究では，仮想計算機パッキング問題の最適解を現実的な計算時間で求める，マッチングに基づく仮想計算機パッキングアルゴリズムを提案する．また，比較として計算時間の短縮を目的としたヒューリスティック手法であるグリーディアルゴリズムを提案する．

4.1 マッチングベースアルゴリズム

本研究で扱う仮想計算機パッキング問題の VM の送受信数の上限は 1 回であるため，マッチングを用いたアルゴリズムが構成できる．物理計算機の集合を頂点集合とする有向完全グラフを (P, A) とする．つまり，$A = \{(i, j) \mid i, j \in P, i \neq j\}$ である．各物理計算機 i に対して，i に初期配置されている VM の集合を V^i，つまり $V^i = \{k \in V \mid x^0_{ik} = 1\}$ とする．物理計算機 i に初期配置されている VM k が物理計算機 j に再配

置されたときの消費電力量の削減量 $\tilde{c}(i,j,k)$ は

$$\begin{aligned} & cost_i(\sum_{k' \in V^i} vc_{k'}) - cost_i(\sum_{k' \in V^i \setminus \{k\}} vc_{k'}) \\ & + base_i \cdot \max\{0, 2 - |V^i|\} \\ & + cost_j(\sum_{k' \in V^j} vc_{k'}) - cost_j(\sum_{k' \in V^j \cup \{k\}} vc_{k'}) \\ & + base_j \cdot \min\{0, |V^j| - 1\} \end{aligned} \tag{11}$$

で与えられる．物理計算機 i と j に対して，資源制約 (5)(6) を満たすように i から j へ移動できる VM の集合 $V^{i \to j}$ を

$$\left\{ k \in V^i \;\middle|\; \begin{array}{l} \sum_{\ell \in V^i \setminus \{k\}} vm_\ell \le m_j, \\ \sum_{\ell \in V^i \setminus \{k\}} vc_\ell \le c_j, \\ \sum_{\ell \in V^j \cup \{k\}} vm_\ell \le m_j, \\ \sum_{\ell \in V^j \cup \{k\}} vc_\ell \le c_j \end{array} \right\} \tag{12}$$

としたとき，各枝 $(i,j) \in A$ の重み $c(i,j)$ を

$$\max\{\tilde{c}(i,j,k) \mid k \in V^{i \to j}\} \tag{13}$$

で与え，$c(i,j) = \tilde{c}(i,j,k)$ を達成する k を保持する関数を σ とする．つまり，$c(i,j) = \tilde{c}(i,j,\sigma(i,j))$ が成り立つ．また，$\tilde{A} = \{(i,j) \in A \mid V^{i \to j} \neq \emptyset\}$ とする．物理計算機間の VM の移動は，マイグレーションの回数が高々1 回に制限されているので，グラフ (P, A) 上の端点を共有しない枝集合，すなわち，マッチングで表される．図 1 は 7 台の物理計算機と VM が与えられている時のマイグレーションの様子を表している．物理計算機 5 から 4，2 から 7，および 6 から 3 へ VM を移動したとする．この再配置を物理計算機を頂点とする完全有向グラフで表すと，VM の移動方向を表す有向枝のマッチングに対応していることがわかる．

また，初期配置において，資源制約 (5)(6) を満たしていない物理計算機の集合 $\{i \in P \mid \sum_{k \in V^i} vc_k > c_i\} \cup \{i \in P \mid \sum_{k \in V^i} vm_k > m_i\}$ を U すると，U の物理計算機からは必ず VM を移動させるので，資源制約を満たすような再配置は $(P, \tilde{A})$ 上の U を端点として含むマッチング（以下，U-マッチング）で表せる．従って，U が空集合の場合も含めて次の定理が成り立つ．

定理 1. グラフ $(P, \tilde{A})$ で重み c の和を最大にする U-マッチングを M^* とする．このとき，各 $(i,j) \in M^*$ に対して，i から j へ $\sigma(i,j)$ を移動させる再配置は仮想計算機パッキング問題の最適解となる．

Proof. 枝集合 $\tilde{A}$ と U の決め方より，得られた再配置が資源制約を満たすことは明らか．また，マッチングから得られているので，各物理計算機に対するマイグレーションの回数も高々1 回である．

次に，仮想計算機パッキング問題の最適解から得られる各物理計算機 i への配置を $\hat{V}^i$ で表す．この

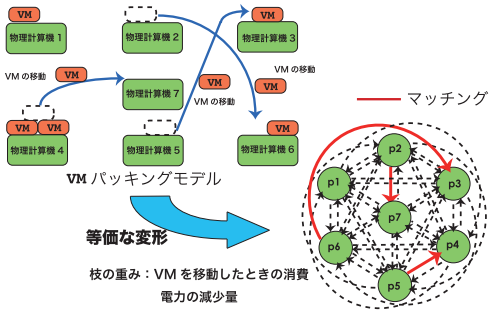

図 1　仮想計算機パッキング問題をマッチング問題に変換する例

とき，$\hat{M} = \{(i,j) \mid i,j \in P, V^i \cap \hat{V}^j \neq \emptyset\}$ とすると，各物理計算機間のマイグレーション回数が高々1 回なので，$\hat{M}$ は $(P, \tilde{A})$ の U-マッチングになる．ここで，$P^+ = \{i \in P \mid \hat{V}^i \setminus V^i \neq \emptyset\}$，$P^- = \{i \in P \mid V^i \setminus \hat{V}^i \neq \emptyset\}$ とし，各 $i \in P^+$ に対して，$\hat{V}^i \setminus V^i = \{k_i\}$，各 $i \in P^-$ に対して，$V^i \setminus \hat{V}^i = \{k_i\}$ とする．この性質を利用すると，初期配置と再配置との消費電力量の削減量は

$$\begin{aligned} & \sum_{i \in P}(cost_i(\sum_{k \in V^i} vc_k) + base_i \cdot \min\{1, |V^i|\}) \\ & - \sum_{i \in P}(cost_i(\sum_{k \in \hat{V}^i} vc_k) + base_i \cdot \min\{1, |\hat{V}^i|\}) \\ = & \sum_{i \in P^+ \cup P^-}(cost_i(\sum_{k \in V^i} vc_k) - cost_i(\sum_{k \in \hat{V}^i} vc_k)) \\ & + \sum_{i \in P^-} base_i(\min\{1, |V^i|\} - \min\{1, |V^i| - 1\}) \\ & + \sum_{i \in P^+} base_i(\min\{1, |V^i|\} - \min\{1, |V^i| + 1\}) \\ = & \sum_{i \in P^+}(cost_i(\sum_{k \in V^i} vc_k) - cost_i(\sum_{k \in V^i \cup \{k_i\}} vc_k)) \\ & + \sum_{i \in P^-}(cost_i(\sum_{k \in V^i} vc_k) - cost_i(\sum_{k \in V^i \setminus \{k_i\}} vc_k)) \\ & + \sum_{i \in P^-} base_i \cdot \max\{0, 2 - |V^i|\} \\ & + \sum_{i \in P^+} base_i \cdot \min\{0, |V^i| - 1\} \\ = & \sum_{(i,j) \in \hat{M}} \tilde{c}(i,j,k_i) \end{aligned} \tag{14}$$

となる．明らかに $k_i \in V^{i \to j}$ なので，$\tilde{c}(i,j,k_i) \le c(i,j)$ となる．一方，M^* は最大重みの U-マッチングなので，$\sum_{(i,j) \in \hat{M}} c(i,j) \le \sum_{(i,j) \in M^*} c(i,j)$ であり，M^* から得られる再配置の方が消費電力量の削減量が小さくなることはない．つまり，M^* から得られる再配置も最適な再配置である．　□

マッチングを用いた仮想計算機パッキング問題に対

する厳密アルゴリズムを以下に示す．

マッチングベースアルゴリズム **(MBA)**

Step 1. 頂点を物理計算機とする有向完全グラフ (P, A) を構成し，各枝 $(i, j) \in A$ の重み $c(i, j)$ を求める．資源制約を満たしていない物理計算機の集合を $U = \{i \in P \mid \sum_{k \in V^i} vc_k > c_i\} \cup \{i \in P \mid \sum_{k \in V^i} vm_k > m_i\}$ とする．

Step 2. $(P, \tilde{A})$ を無向グラフとみなして，最大重みの U-マッチング M^* を求める．

Step 3. 各 $(i^*, j^*) \in M^*$ に対して，$\sigma(i^*, j^*)$ を i^* から j^* に再配置する．

Step 2 で最大重みの U-マッチングを求めるのには，$U = \emptyset$ のときの Edmonds の多項式時間アルゴリズム[8] を利用して．O($|P|^3$) 時間，または，O($|P||\tilde{A}| + |P|^2 \log |P|$) 時間で解くことができる[8)~10)]．

定理 2. 仮想計算機パッキング問題は $O(|P|^3 + |V||P|)$ 時間で解ける．

Proof. Step 1 で各枝 (i, j) の重み $c(i, j)$ を求めるのに O($|V^i|$) の手間が必要であるから i から出る枝の重みをすべて求めるのに O($|V^i||P|$) 時間かかり，Step 1 の計算量は O($\sum_{i \in V} |V^i||P|$) = O($|V||P|$) となる．また，$|\tilde{A}|$ =O($|P|^2$) なので，Step 2 は上に述べたように O($|P|^3$) 時間で最大重みの U-マッチングを求める．Step 3 はマッチングの各枝に対する操作が必要なのでその計算量は O($|P|$) となる． □

4.2 グリーディアルゴリズム

仮想計算機パッキング問題に対するある程度良い解を高速に求める，ヒューリスティクとしてグリーディアルゴリズム (GREEDY) を提案する．GREEDY はできるだけ 1 台の物理計算機にたくさんの VM を割当て物理計算機の稼働台数を削減する．

グリーディアルゴリズム **(GREEDY)**

Step 1 各物理計算機 $i(\in P)$ に割当てられている VM の CPU 使用量の総和とメモリ使用量の総和を計算し，$\sum_{k \in V^i} vc_k > c_i$ もしくは $\sum_{k \in V^i} vm_k > m_i$ となるもの $High(\subseteq P)$ とそうでないもの $Low(\subseteq P)$ に分類する．

Step 2 $High$ および Low を各物理計算機 i の資源使用率 $util_i$ に応じてソートする．ここで，$util_i$ を次のように定義する．

$$util_i = \sum_{k \in V^i} vc_k/c_i + \sum_{k \in V^i} vm_k/m_i \qquad (15)$$

Step 3 物理計算機 $h(\in High)$ に対して，$util_i$ の高い順に次の操作を行う．物理計算機 h に割当てられている VM の集合 V^h から適切な k' を選択し，Low のいずれかの物理計算機 $l(\in Low)$ にマイグレートする．各物理計算機 $h(\in High)$ に対して，CPU およびメモリ使用量の超過した分を $oc_h = \max\{0, \sum_{k \in V^h} vc_k - c_h\}$，$om_h = \max\{0, \sum_{k \in V^h} vm_k - m_h\}$ とすると，以下の優先順位でマイグレートする VMk' を決定する．

(a) $vc_k \geq oc_h$ かつ $vm_k \geq om_h$ で *diff1* が最小の $k(\in V^h)$

(b) $vc_k < oc_h$ かつ $vm_k \geq om_h$ で *diff2* が最小の $k(\in V^h)$

(c) $vc_k \geq oc_h$ かつ $vm_k < om_h$ で *diff3* が最小の $k(\in V^h)$

(d) $vc_k < oc_h$ かつ $vm_k < oc_h$ で *diff4* が最小の $k(\in V^h)$

ここで，*diff1-4* は以下のように算出する．

$$\textit{diff1} = (vc_k - oc_h)/c_h + (vm_k - om_h)/m_h \qquad (16)$$

$$\textit{diff2} = (oc_h - vc_k)/c_h + (vm_k - om_h)/m_h \qquad (17)$$

$$\textit{diff3} = (vc_k - oc_h)/c_h + (om_h - vm_k)/m_h \qquad (18)$$

$$\textit{diff4} = (oc_h - vc_k)/c_h + (om_h - vm_k)/m_h \qquad (19)$$

マイグレート先の物理計算機 $l(\in Low)$ は資源使用率の高い物理計算機から順に $k'(\in V^h)$ が割当て可能なものを探し，割当てる．$Low \setminus \{l\}$ とする．

Step 4 Low に含まれる物理計算機のうち，資源の使用率が低い物理計算機 l_{send} から順に，マイグレート先 l_{recv} を決定していく．l_{recv} は資源使用率の高い物理計算機から $V^{l_{send}}$ に含まれる VM を割当て可能なものを探し割当てる．$Low \setminus \{l_{recv}\}$ とする．

以上の処理により，VM を割当てる物理計算機数の最小化を試みる．GREEDY では，負荷の急激な上昇により再配置しても資源制約を満たさない場合も扱うことができる．

5. 評価実験

各 VM の CPU 使用量とメモリ使用量が期ごとに変化するとき，1 期ごとに仮想計算機パッキング問題を解いて，VM の再配置を行う設定で，200 期間での性能を測る．評価は，MBA，GREEDY と VMPP を商用ソルバを用いて解く IP の各期間における平均消費電力量および平均計算時間で比較する．

5.1 実験環境

物理計算機数と VM 数は等しいとし，それぞれ 50 から 300 までの 50 刻みの 6 つの問題サイズを用いる．

表 1 パラメータ設定

パラメータ	環境設定
アルゴリズム，ソルバ	MBA, GREEDY IP(CPLEX)
物理計算機数，VM 数	50 から 300 までの 50 刻み
期間	200
物理計算機の CPU キャパシティ	1.0
物理計算機のメモリキャパシティ	4096 [MB]
VM の CPU 利用量の avg. / max	0.5 / 1.0
VM のメモリ利用量の avg. / max	2048 / 4096 [MB]
VM の CPU，メモリ利用量の変動	Sine カーブ (位相: 一様分布，平均周期: $1/12\pi$, 一様分布)
$base_i$ / $cost_i(z)$	53.0 / 47.0z

物理計算機の CPU のキャパシティ（コア数 x 利用率の上限）は 1.0 とし，メモリキャパシティは 4096[MB] とする．各期における VM の CPU とメモリの使用量の平均値 / 最大値はそれぞれ，0.5 / 1.0，2048 / 4096 [MB] とする．VM の資源使用量の変動は，Sine カーブを用いる．VM のデータについては，実環境での観測データが望ましいが，データ収集は困難なため，今後の課題とする．また，Sine カーブに基づくデータ生成は期間ごとに資源使用量が変動するため，実際の運用時に比べて厳しい条件である．消費電力に関するパラメータは，文献[7] の調査結果をもとに，すべての $i \in P$ で $base_i = 53.0$，$cost_i(z) = 47.0z$ とする．初期配置は，各物理計算機に 1 台ずつ VM をランダムに割当てたものとする．

実験は，Intel Core i7 CPU (3.4GHz, 8 コア), MacOSX 10.7 64bit, メモリ 16GB の計算機上で実行する．商用ソルバとして，CPLEX(IBM ILOG CPLEX Optimization Studio_Academic V12.3, 64bit 版) を用い，スレッド処理を 8 スレッド使用する．また，タイムアウト時間を 600 秒とし，600 秒で最適解が出ない場合は暫定解を出力する．実験は，MBA と GREEDY 及び，CPLEX を用いて整数計画問題を解く IP の性能を比較する．CPLEX のインターフェースとして，Python の pulp ライブラリを利用する．また、参考までに無償ソルバ の一つである GLPK(Ver.4.47)[14] を用いた場合の計算時間も調査する．他のアルゴリズムについては，Python 2.6.5 で実装した．MBA の step 2 で最大重みのマッチングを求めるプロシージャは，文献[9] で示されている O($|P|^3$) アルゴリズムを J. van Rantwijk[11] が実装し，公開しているものを，U-マッチングを出力するように修正したものを用いる．

5.2 実験結果

まず，各アルゴリズムの 1 期間あたりの平均電力消費量と平均計算時間を比較する．結果を表 2 に示す．但し，MBA はマッチングベースアルゴリズム，GREEDY はグリーディアルゴリズム，IP は VMPP ((3)-(10) 式) を CPLEX で解いた結果である．ただし，VMPP の制約 (8) は x_{ik} と x^0_{ik} の積が 1 ならば物理計算機 i 上に VM k が再配置されたことを意味する性質を利用し，

$$\sum_{k \in V}(x_{ik} + x^0_{ik} - 2x_{ik}x^0_{ik}) \leq 1,\ \forall i \in P \qquad (20)$$

としている．また，FIX は，各期で再配置を行わなかった場合の結果を表す．問題が実行不能のとき，MBA や IP では，再配置を行わず，1 期前の配置を用いる．一方，GREEDY では，実行不能時でも出来るだけ消費電力量が小さくなるように再配置を行っている．IP でも実行不能時にも再配置を行うように VMPP の制約 (5) と (6) をそれぞれ，制約を超過した量を表す非負変数 p_i, q_i を用いて

$$\sum_{k \in V} vm_k x_{ik} - m_i \leq p_i,\ \forall i \in P \qquad (21)$$

$$\sum_{k \in V} vc_k x_{ik} - c_i \leq q_i,\ \forall i \in P \qquad (22)$$

と置き換え，目的関数 (3) を

$$\sum_{i \in P}(cost_i(\sum_{k \in V} vc_k x_{ik}) + base_i y_i) + \sum_{i \in P} p_i + \sum_{i \in P} q_i \qquad (23)$$

の最小化とした整数計画問題とし，これを CPLEX で解いた結果を IPr に表す．サイズ 300 の問題については，実験が終了しなかったため，結果を載せていない．MBA は IP や GREEDY に比べ，多くの場合で平均消費電力量が小さい事がわかる．MBA と IP の平均消費電力量が異なっているのは，最適解が複数有る場合，異なる再配置になることがあり，かつ IP では，指定時間 (600 秒) 内に最適解が求められない場合，暫定解を用いているためである．IP と IPr を比較すると，IPr の方が平均消費電力量が小さいことから，実行不能時にも再配置を行う意義がわかる．計算時間は，GREEDY が圧倒的に短く，MBA と IP を比較すると，MBA の方が計算時間が短い．CPLEX は整数計画問題や線形計画問題に対する汎用ソルバとして企業の生産計画などに使われているが，問題構造を把握し，適切な多項式時間アルゴリズムを構築する事で，計算時間を抑え，最適解を得られる事がわかった．表 3 にさらに大きなサイズの問題を解いたときの MBA と GREEDY を比較したものを示す．MBA は，問題サイズ 700 で平均計算時間は約 4 秒で，GREEDY の計算時間には劣るものの，十分に実用的な時間であることがわかる．

さらに，整数計画問題 VMPP を解くのに GLPK を用いると，物理計算機数 (=VM 数) が 10 のとき，平均計算時間が 0.168(sec.) であり，物理計算機数 (=VM 数) が 20 のときは，200 期間中，600 秒以内で解ける期はなく，暫定解もあまり良い解とは期待できない結

表 2　1 期間あたりの平均電力消費量と平均計算時間の比較

物理計算機数 (=VM 数)	平均消費電力量					平均計算時間 (sec.)			
	FIX	MBA	IP	GREEDY	IPr	MBA	IP	GREEDY	IPr
50	3814.716	1951.501	2817.311	2892.781	2736.334	0.018	0.659	0.002	0.484
100	7623.835	3959.415	5925.185	5643.225	5400.173	0.076	13.243	0.004	2.753
150	11469.435	5841.630	9395.015	8369.995	8101.466	0.157	12.360	0.007	12.107
200	15259.594	7752.674	11755.764	11087.964	10736.924	0.280	22.335	0.010	48.862
250	19055.052	9579.447	14552.893	13688.537	13313.994	0.433	21.358	0.014	239.831
300	22840.991	11423.201	17479.776	16385.326	15976.583	0.625	48.815	0.017	530.083

表 3　1 期間あたりの平均電力消費量と平均計算時間の比較

物理計算機数 (=VM 数)	平均消費電力量			平均計算時間 (sec.)	
	FIX	MBA	GREEDY	MBA	GREEDY
400	30507.199	15273.144	21826.859	1.145	0.026
500	38130.399	19116.119	27164.169	1.890	0.037
600	45808.771	22914.891	32493.581	2.771	0.048
700	53408.342	26723.107	37835.882	3.892	0.061

表 4　各アルゴリズムを用いた場合の消費電力削減率

物理計算機数 (=VM 数)	MBA	IP	GREEDY	IPr
50	0.49	0.26	0.24	0.28
100	0.48	0.22	0.26	0.29
150	0.49	0.18	0.27	0.29
200	0.49	0.23	0.27	0.30
250	0.50	0.24	0.28	0.30
300	0.50	0.23	0.28	0.30

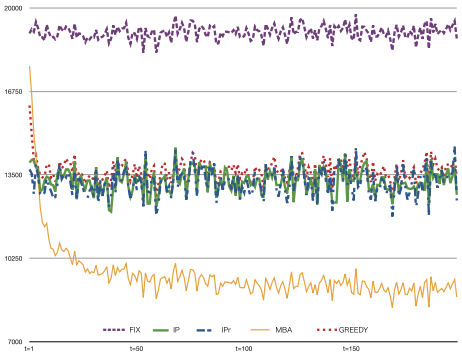

図 2　物理計算機数 (=VM 数)250 のときの FIX, MBA, GREEDY, IP, IPr の消費電力量の推移

果であった．

次に図 2 に問題サイズ 250 の時の FIX と MBA, GREEDY，IP，IPr の消費電力量の推移を示す．横軸に期間を，縦軸にその期での消費電力量を表す．ほとんどの期間でどのアルゴリズムも消費電力量の増加減の傾向は同じである事がわかる．また，FIX と比べると MBA, GREEDY, IP, IPr ともに大幅に消費電力量を削減できている事がわかる．表 4 は各アルゴリズムの消費電力の削減率を表していて，再配置をしなかった場合に比べ，どのアルゴリズムも消費電力を 18%から 50%削減できている事がわかる．

最後に，再配置による性能劣化を比較する．ユーザ

表 5　アルゴリズムごとの性能劣化数の平均

物理計算機数 (=VM 数)	性能劣化数			
	MBA	IP	GREEDY	IPr
50	1.34	8.05	3.67	1.43
100	1.21	18.39	8.52	2.87
150	1.16	23.47	13.80	4.21
200	0.65	34.66	18.44	5.27
250	0.41	36.68	24.32	6.57
300	0.61	48.11	29.14	7.67

が感じる性能劣化には様々な要素が絡んでいるが，ここでは，性能劣化を表す指標として，CPU およびメモリの使用量の何れかが容量を超えている割合を表す (24) 式で定義される性能劣化数を用い，数値が大きくなるほどユーザ体験が劣化する事を意味している．各物理計算機 $i(\in P)$ に対して，保持している VM の集合を V^i とする．このとき，性能劣化数は，

$$\sum_{i \in P} \max\{0, \sum_{k \in V^i} vc_k - c_i\}/c_i + \sum_{i \in P} \max\{0, \sum_{k \in V^i} vm_k - m_i\}/m_i \quad (24)$$

で定義される．表 5 に各アルゴリズムを用いた場合の再配置後の性能劣化数の平均を示す．表 5 を見ると，MBA 以外のアルゴリズムでは問題サイズの増加に比例して性能劣化数が増加している．一方で，MBA は最も性能劣化数が小さかった．これは，各期ごとに最適な配置を求めた結果と思われる．また，IP と IPr では，IPr の方が性能劣化数が小さいが，これは実行不能時の再配置の重要性を示唆している．以上のことより，MBA に対しても，実行不能時にも再配置を行うようにアルゴリズムを修正することは性能劣化も改善すると予想される．

提案するマッチングベースアルゴリズムは物理計算機がマルチコアのモデルなどにもそのまま拡張できる．一方で，通信通信衝突を回避するために 1 つの計算機が送受信する VM 数の上限を 1 としており，マッチングベースアルゴリズムはこの性質を利用していた．送受信数を 2 以上としてより消費電力の削減を試みる方針も考えられるが，送受信数が 2 回以上を許す場合，仮想計算機パッキング問題は，3-分割問題を帰着させる事で NP-困難性を示せる．よって，マッチング

ベースアルゴリズムのような効率的な解法は望めず，3章で与えた整数計画問題をソルバで解いたり，グリーティーアルゴリズムのようなヒューリスティックアプローチが良い．また，提案手法は VM 上のシステムのダウンタイムを最小限にするために高速ポストコピーマイグレーションを用いた VM パッキングシステムへの適用を前提しているが，プレコピーマイグレーションを前提としたシステムにおいても適用できる．プレコピーマイグレーションでは，メモリイメージの転送時間が VM 上のシステムのダウンタイムに直接反映されるため，提案する通信衝突を回避した再配置により高速なメモリイメージ転送が可能となり，ダウンタイムを短縮することができる．

6. おわりに

本研究では，仮想計算機パッキング問題に対して，多項式時間アルゴリズムのマッチングベースアルゴリズムと，ヒューリスティックアルゴリズムのグリーディアルゴリズムを提案した．評価実験では，最適化ソルバと比較し，マッチングベースアルゴリズムはグリーディアルゴリズムには実行時間では劣るものの，実用的な時間でより効果的な消費電力の削減が可能であり，性能劣化も小さい事を示した．また，商用ソルバよりも短時間で最適解を得られる事を示した．

今後は，計算機実験のモデルを，実システムに近づけアルゴリズムの効果を検証すること，大域的な消費電力の削減効果を評価するために多期間仮想計算機パッキング問題に拡張し，提案アルゴリズムの大域的な性能評価を行う．また，提案手法を実 VM パッキングシステムに適用し，CPU やメモリ等の資源の使用量に基づく性能劣化に加えて複数の実アプリケーションにおける性能劣化について調査し，提案手法の実用性を示す．

謝辞　本研究の一部は，CREST(情報システムの超低消費電力化を目指した技術革新と統合化技術) および，科研費 (22510135) の助成を受け実施した．

参考文献

1) T. Hirofuchi, H. Nakada, S. Itoh and S. Sekiguchi. Reactive Consolidation of Virtual Machines Enabled by Postcopy Live Migration, *Proc. of the 5th Int. Workshop on Virtualization technologies in distributed computing*, pp.11–18, (2011).
2) H. Nakada, T. Hirofuchi, H. Ogawa and S. Itoh. Toward Virtual Machine Packing Optimization Based on Genetic Algorithm, *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living (Proc. of Int. Symposium on Distributed Computing and Artificial Intelligence), LNCS*, Vol. 5518, pp.651–654, Springer, (2009).
3) 中田秀基，竹房あつ子，広渕崇宏，伊藤　智，関口智嗣．仮想計算機パッキングへの最適化手法の適用，電子情報通信学会技術研究報告 (CPSY: コンピュータシステム)，Vol.110, No.167, pp. 55–60, (2010).
4) 竹房あつ子，中田秀基，広渕崇宏，伊藤智，関口智嗣．省電力化にむけた仮想計算機パッキングアルゴリズムの提案，電子情報通信学会技術研究報告 (CPSY: コンピュータシステム)，Vol.110, No.167, pp.73–78, (2011).
5) T. Hirofuchi, H. Nakada, S. Itoh and S. Sekiguchi. Enabling Instantaneous Relocation of Virtual Machines with a Lightweight VMM Extension, *The 10th IEEE/ACM Int. Conference on Cluster, Cloud and Grid Computing (CCGrid2010)*, pp. 73–83, (2010).
6) 広渕崇宏，中田秀基，伊藤智，関口智嗣．高速マイグレーションを利用した仮想マシン配置最適化システムの検討，情報処理学会研究報告 (2010-OS-115), pp. 1–13, (2010).
7) T. Hirofuchi, H. Nakada, S. Itoh and S. Sekiguchi. Making VM Consolidation More Energy-efficient by Postcopy Live Migration, *Proc. of the 2nd Int. Conference on Cloud Computing, GRIDs, and Vidualization*, pp.195–204, (2011).
8) B. H. Korte, and J. Vygen. *Combinatorial optimization: theory and algorithms*, Springer, (2004).
9) Z. Galil. Efficient Algorithms for Finding Maximum Matching in Graphs. *ACM Conputing Serveys*, Vol.18, 1, pp.23–38, (1986).
10) J. Edmonds. Paths, trees, and flowers, *Canadian Journal Mathematics*, Vol. 17, pp449–467, (1965).
11) J. van Rantwijk. “Joris_VR”, http://jorisvr.nl/maximummatching.html
12) M.R. Garey and D.S. Johnson. *Computeres and Intractability - A Guide to the Theory of NP-Cpmpleteness*, A Series of Books in the Mathematical Science, 22nd printing, W.H. Freeman and Company, New York, (2000)
13) “IBM ILOG CPLEX”, http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.
14) “GLPK”, http://www.gnu.org/software/glpk/
15) F. Hermenier, X. Lorca, J-M. Menaud, G. Muller and J. Lawall. Entropy: a consolidation manager for clusters, *Proc. of the 5th Int. Conference on Virtual Execution Environments*, pp.41–50, (2009).
16) M.L. Stillwell, F. Vivien and H. Casanova. Dynamic Fractional Resource Scheduling for HPC Workloads. *Proc. of Int. Parallel and Distributed Processing Symposium (IPDPS)*, pp.1–12, (2010).