

仮想マシン配置のためのパッキングアルゴリズム

網 代 育 大^{†1}

仮想マシンの配置問題に関して、オペレーションズリサーチの分野で研究されてきたベクトルパッキング問題に対する古典および最近のアルゴリズムの観点と、メインフレームやデータセンタの効率運用、サーバ統合のために研究されてきたアプリケーションやワークロード、仮想マシン配置の観点から関連研究を概観する。これに基づき、我々がサーバ統合用に開発したアルゴリズムの位置づけや設計思想について述べる。

Packing Algorithms for Virtual Machine Placement

YASUHIRO AJIRO^{†1}

Related work to virtual machine placement is surveyed from two points of view; one is conventional and recent algorithms for the vector packing problem in the field of operations research; the other is the placement of applications, workloads, and virtual machines not only for server consolidation but for efficient management of mainframe computers and data centers also. Based on this survey, the positioning and design concept of our technique of improving packing algorithms for server consolidation are described.

1. はじめに

本稿で述べる仮想マシン配置とは、異なるアプリケーションやワークロードをもつ複数の仮想マシン (VM) が与えられたときに、コストや性能、可用性の観点から、各 VM をどの物理マシンに配置すればよいかを求める問題である。仮想マシン配置が注目されるようになったのは、サーバ統合がビジネスとして大きく成長したためである。サーバ統合は、地理的に離れた場所に置かれた多数のサーバを 1ヶ所にまとめる位置統合、仮想化技術を用い

て既存のサーバ群を物理サーバ上の VM に集約する物理統合、同種のデータやアプリケーションを 1つにまとめるデータ・アプリケーション統合の 3 種類に分類されるが、多くの場合、物理統合の意味で用いられる³⁾。

サーバ統合が注目を浴びたのは、1990 年代に情報システムのオープン化の波が到来し、ハードウェア価格が大きく低下した結果、特に Windows Server を搭載した x86 サーバが次々に導入されて、企業内に大量のサーバが増殖してしまったことによる (海外ではこの状況を server sprawl とよぶ)。VMware をはじめとする仮想化製品を用いて、この増殖した Windows サーバを少数の物理サーバに集約するビジネスが 2000 年代前半から行なわれるようになった。我々のグループは、サーバ統合時の VM 配置問題を多次元のベクトルパッキング問題 (vector packing problem) として定式化し、これに対する新しいヒューリスティックアルゴリズムを開発した³⁾。開発したアルゴリズムは、2005 年、サーバ統合設計ツールにサーバ台数の見積り機能として組み込まれ、アルケマイスター (Alchmeister) とよばれる弊社のサーバ統合サービスで利用された。アルケマイスターというのは、錬金術 (alchemy) と名工 (meister) を組み合わせた造語である。

その後、カナダ PlateSpin 社 (現在は米 Novell によって買収) のサーバ管理ツールである PowerRecon に同様の機能が追加されたほか、VMware Capacity Planner、Microsoft Assessment and Planning Toolkit といった同様のコンセプトをもつ製品がリリースされた。また、サーバ台数の削減を目的とした VM 配置や集約効率に関する多くの論文も発表されている。我々の 3) を含め、これらの論文の多くは、VM やアプリケーションのワークロードを考慮し、時間帯によって VM の占有リソース量が変化するリソース共有型の統合を前提としているが、CPU のマルチコア化にともなう価格性能比の大幅な向上、iSCSI を用いた低価格な共有ストレージの台頭によって、現在は CPU コア 1 つにつき 1 VM のような、リソースの一部を固定的に割り当てて排他的に利用する排他型の統合が主流である。

排他型の統合では、サーバや VM のワークロードではなく、スペックが考慮される。たとえば、統合前のサーバが 2 GHz の CPU と 512 MB のメモリを搭載していれば、統合先のサーバから同等の VM を切り出して移行する。ディスクやネットワーク帯域は、アーキテクチャ上、共有される。統合対象となるサーバは、重要度や負荷の低いサーバであり、5 年以上前にリリースされた、いまとなっては低スペックなサーバである。また、現在はサーバ価格の大部分をメモリが占めており、CPU の重要性は相対的に低下している。ディスクやネットワーク I/O 処理の大きい例外的なサーバや VM は別途取り扱う必要があるものの、アプリケーションの重要性に基づいて、1 コアあたり 1 VM や 2 VM と方針を決め、あと

^{†1} NEC サービスプラットフォーム研究所
Service Platforms Research Laboratories, NEC Corporation

は統合前のサーバのメモリ搭載量を単純に積み上げていくだけで、多くの VM の配置先を決定できる。排他型の統合では、共有型の統合よりも集約効率が低下することになるが、1 コアあたりに多数の VM を統合した場合の仮想化オーバーヘッドの増大や性能問題の発生リスクを重くみて、排他型の統合が選択されている。

その一方で、仮想化技術の適用先は、サーバ統合から XaaS のようなクラウドコンピューティング基盤へと移行しつつある。現在、IaaS 事業の主流は、1 GHz CPU、512 MB メモリ、500 MB 分のディスク容量を一定期間貸し出すような、リソースを排他的に利用するモデルである。しかしながらクラウド環境においては、IaaS や PaaS 事業者が、リソース共有型の VM 配置を行なう、より集約度の高い、低価格な基盤を提供する可能性と合理性があると我々は考えている。企業が自前で所有する（オンプレミスな）サーバや VM と異なり、クラウド環境の VM は一定期間のリース契約で利用し、そのワークロードも定期的に報告されるため、重要度やワークロードの低い VM に関しては、集約効率を上げてリソースとコストを節約しようというインセンティブが働くからである。さらにクラウド環境では、問題が発生した場合に元の構成に戻すのが容易なため、積極的なリソース削減を行ないやすいという理由もある。

本稿では、以上に述べたようなサーバ環境の変化から、VM 配置が今後また重要となるだろうと考え、続く第 2 節でベクトルパッキング問題や VM 配置に関連する既存研究を概観する。第 3 節では、概観した関連研究に基づき、我々がサーバ統合用に開発したアルゴリズムの位置づけや設計思想について述べる。第 4 節はこれらのまとめである。

2. 関連研究

2.1 ベクトルパッキング問題

ベクトルパッキング問題は、アイテム i の d 次元目の要素を $a_{i,d}$ ($0 \leq a_{i,d} \leq 1$) としたとき、 D 次元ベクトル $(a_{i,1}, a_{i,2}, \dots, a_{i,D})$ をサイズとするアイテムの集合を、要素がすべて 1 の D 次元ベクトル $(1, \dots, 1)$ を容量とするベクトルの箱 (vector bin) に詰めるとすると、箱がいくつ必要になるかを求める問題である^{21),25)}。箱の負荷は、箱に詰めたアイテムのベクトル和で定義される。サーバ統合の場合、CPU 使用率が 15%、メモリ使用量が 230 MB の VM と、CPU 使用率が 20%、メモリ使用量が 150 MB の VM を同一の物理サーバ上に配置するとき、ハイパーバイザによるリソース消費や仮想化オーバーヘッドを除いて、物理サーバのリソース使用量はそれぞれ 35%、380 MB と見積もれる。つまり、 i 番目の VM によるリソース d の使用量を $\rho_{i,d}$ とし、VM を配置する物理サーバのリソース d の量、あ

るいは CPU 使用率 80% 等の閾値を c_d 、ベクトルの箱を (c_1, c_2, \dots, c_D) とすると、統合後の物理サーバ台数を最小化する問題はベクトルパッキング問題となる。リソース使用量 $\rho_{i,d}$ ($\leq c_d$) を $a_{i,d} = \rho_{i,d}/c_d$ と変換すれば、VM 配置の問題は冒頭で述べたベクトルパッキング問題に一般化できる。ただし、VM のワークロードに関する時系列を考慮すると次元数が非常に大きくなるが、これについては第 2.2 節で述べる。

オペレーションズリサーチの分野では、ベクトルだけでなく、2 次元の矩形や 3 次元の立方体をユークリッド空間上に詰めるビンパッキング問題 (bin packing problem) に関する多くの研究がなされている^{14),20)}。次元数 1 の問題に関しては、ビンパッキングとベクトルパッキングは同一の問題であり、3 次元までのビンパッキング問題は NP-complete であることが証明されている¹⁶⁾。多次元のベクトルパッキング問題は一般に NP-hard であり¹⁸⁾、厳密解法やヒューリスティックなアルゴリズムが研究されている。厳密解法に関する最近の研究としては、Spiessma³⁴⁾ や Caprara⁸⁾ らが、2 次元ベクトルパッキング問題に対して、分枝限定法 (branch-and-bound method) で用いる新しい下限 (lower bound) を提案している。ビンパッキング問題に関する最近の研究としては 12), 27) がある。また、ベクトルパッキング問題に対する多項式時間近似スキーム (PTAS) の理論的研究として、4), 10), 15), 31) がある。なお、パッキング問題には、ほかに円や球を詰める circle/sphere packing 問題^{*1}等、いくつかの亜種が存在する。

厳密解法はいまのところ、次元数やアイテム数の少ない問題を対象としているが、実用上はあらゆる問題に対して最適に近い解を求めるヒューリスティックなアルゴリズムが不可欠である。次元数 1 のパッキング問題に対する古典的なヒューリスティックアルゴリズムや、それによって得られる解の平均的な品質の解析 (average case analysis) は 13) にまとめられている。古典的なヒューリスティックアルゴリズムとして代表的なものに、First-Fit (FF), Best-Fit (BF), Worst-Fit (WF), Next-Fit (NF) の 4 つがある。FF では、まず箱がない状態から始め、より早く追加された箱へのアイテムのパッキングを試みて、どの箱にも詰められなければ新しい箱を追加して詰めるという操作を繰り返す。BF はできるだけ空き部分の小さい箱にアイテムを詰めようとし、WF は逆に空き部分のより大きな箱にアイテムを詰めようとするアルゴリズムである。NF はもっとも単純で、最後に追加された箱にアイテムを詰めようとし、だめならすぐに新しい箱を追加して詰める。アルゴリズムには、アイテム

*1 同半径の球の最密充填は面心立方格子であるというケプラー予想 (Kepler conjecture) が 1611 年に提起され、最近の 1997 年になってようやくコンピュータの力を借りて証明されたことを CMG Conference³⁾ の査読者に教わった。

を与えられた順番に処理するオンライン (online) アルゴリズムと、アイテムの順番を入れ替えるオフライン (offline) アルゴリズムとがあり、一般にオフラインアルゴリズムの方が性能がよい (より少ない箱に詰めることができる)。オフラインアルゴリズムとしては、FF アルゴリズムを適用する際、事前にアイテムを大きい順にソートしておく First-Fit Decreasing (FFD) がよく知られており、他の BF, WF, NF についても同様に BFD, WFD, NFD が定義される。次元数 1 の問題に対しては、FFD の性能が良いことが知られている。

Spieksma による 34) は、厳密解法のほかに、2 次元パッキング問題に対する FFD2 というヒューリスティックアルゴリズムを提案している (Caprara らによる文献⁸⁾ では、2FFD μ として比較されている)。次元数 2 以上の問題に FFD を適用する場合、アイテムをソートするための順序を規定する必要があるが、FFD2 は 1, 2 次元目の要素 $a_{i,1}$, $a_{i,2}$ に対する $\lambda a_{i,1} + a_{i,2}$ の値を基にアイテムをソートし、FFD でアイテムを詰めた結果、空きが一定以上小さい “well-filled” な状態になった箱とアイテムを解の一部として取り出す。その後、 λ の値を変化させながら、残ったアイテムに対する FFD の適用を well-filled な箱ができなくなるまで繰り返す。アイテム数 100 までの評価の結果、アイテムを 1, 2 次元目の要素の大きい方 $\max(a_{i,1}, a_{i,2})$ で単純にソートする FFD (FFGAR)¹⁸⁾ よりも FFD2 は良い性能を示している。

2.2 仮想マシン配置

次に、仮想マシン配置の観点からベクトルパッキング問題を扱っている既存の研究について述べる。大きくわけて VM の配置には、ワークロードが事前にわかっている VM やアプリケーションを最小台数の物理サーバ上に集約するのが主目的の静的な配置と、予測不可能なワークロードに対して、なんらかの評価関数の値を最小化または最大化するように VM を適応的に移動する動的な配置の 2 種類がある。ここでは静的な配置に関する研究を中心に紹介する。

私の知る限り、静的な配置に関する先駆的な研究としては 32), 35) がある。Urgaonkar³⁵⁾ らは、サーバ統合と同様の、アプリケーション数がノード数よりも多い shared hosting platform 上で、アプリケーションの配置アルゴリズムと集約効率との関係性を評価している。このときのアプリケーション配置は (明記されていないが) CPU とネットワークを考慮した 2 次元のベクトルパッキング問題として定式化されており、評価対象の配置アルゴリズムも Random, BF, WF の 3 つである。この研究では、アイテムサイズは 2 つのリソース、すなわち CPU とネットワーク使用率の平均として定義されている。また、リソース使用率として、リソース使用率の確率分布から求めたパーセンタイル値が利用されている。評価の

結果、35) は次の 2 点を明らかにしている。パーセンタイル値に基づくリソースの重複予約 (overbooking) を許容することでアプリケーションの集約効率が大幅に向上する点と、2 種類のリソース使用量にばらつきがある場合は、WF の集約効率が他のアルゴリズムよりも高い点である。後者の理由は、2 次元の問題において、BF や FF のように空きの小さい箱にアイテムを優先的に詰めようとする、空きリソースの断片化 (fragmentation) が発生することによる。この結果は、各次元のサイズが様々なアイテムを詰める場合、WF の方が両方のリソースをバランスよく使える可能性が高いことを示している。

一方、Shahabuddin³²⁾ らは、時系列を考慮した多次元のベクトルパッキング問題に対して、BFD を基にした 4 つのヒューリスティックアルゴリズムを評価している。第 2.1 節の冒頭で説明したサーバ統合の問題では、リソースの種類数が次元数に対応していたが、リソース使用量の時系列を考慮する場合は、時系列中のすべての時間帯に対してリソースが不足しないようにしなければならない。つまり、次元数は時系列長とリソースの種類数の積となる。たとえば、CPU、メモリ、ディスク、ネットワークの 4 種類のリソースに対し、1 時間間隔の時系列データが 1 日分あるとすると、次元数は一気に $4 \cdot 24 = 96$ に増加する。リソース使用量のデータは 15 分や 30 分間隔で採取されることが多いため、現実にはこの 2-4 倍の次元数となる。またリソースに関しても、たとえばディスクの転送量を読み込みと書き込みでわけて考えれば、その分、次元数が増加する。すなわち、時系列を考慮した VM 配置では、次元数が数百以上あると考えるのが合理的である。

そのほか、古典アルゴリズムを用いる研究として、Beck⁵⁾ らは、プロセッサの 6 種類のリソースとプロセッサ間の共有バスを考慮したタスク割当てをベクトルパッキング問題として定式化し、FFD を基にアイテムや箱の選び方を少しずつ変えた 256 種類のアルゴリズムを評価している。Leinberger²⁶⁾ は、同じく多次元問題に対して、箱 j の負荷 $b_{j,d}$ の大小関係がたとえば $b_{j,1} \leq b_{j,3} \leq b_{j,4} \leq b_{j,2} \leq b_{j,5}$ のとき、この箱にベクトル要素の大小関係が逆の $a_{i,5} \leq a_{i,2} \leq a_{i,4} \leq a_{i,3} \leq a_{i,1}$ なるアイテムを選択して詰める Permutation Pack (PP) と、 $a_{i,5}, a_{i,2}, a_{i,4} \leq a_{i,3}, a_{i,1}$ のように条件を緩和した Choose Pack (CP) を提案している。Gupta¹⁹⁾ は 2 つの VM を同一の物理サーバ上に配置しないという制約を導入し、これをグラフ彩色 (graph coloring) として前処理した上で、CPU、メモリ、ディスクの 3 つのリソース使用率の積でアイテムをソートする FFD を採用している。また、同じ所属の Agrawal¹⁾ は、同様の問題にグルーピング遺伝的アルゴリズム (grouping genetic algorithm) を適用している。

Roy³⁰⁾ らは、リソース/次元数 2 のベクトルパッキング問題に対して、2 リソースの使用

量の大きい方の値および和でアイテムや箱をソートする FFD, BFD, WFD を評価している。その結果は、前述の 35) とは異なり、BFD や FFD の方が WFD よりも集約効率が高いことを示している。この違いが発生した理由は、我々も実験で一部確認しているが³⁾、30) がアイテムを事前にソートしたためと考えられる。つまり、大小様々なアイテムを WF で詰めた場合は、結果として各次元の空きが小さくなるが、大きい順に並んだアイテムを WF で詰めると、逆に空きが大きくなってしまふ恐れがある。我々の 3) は 1 次元目の要素のみを使ってソートを行なっているため前提が異なるが、アイテムサイズの箱のサイズに対する比率や、1, 2 次元目の値の相関によって、FFD と WFD の間の優位性が変化することがわかっている。

厳密解法を用いた研究としては Rolia らによる 28) があり、リソース/次元数 1、時系列を考慮したアプリケーション配置問題に対して、厳密解法である整数計画法とヒューリスティックな遺伝的アルゴリズムの 2 つを適用し、多くの場合、遺伝的アルゴリズムでも整数計画法と同等の結果が得られると報告している。また、Rolia は 29) 上で、ある VM 群を同一の物理サーバに配置してはいけないという可用性に関する制約のほかに、同一の物理サーバ上に配置しなければならないというソフトウェアライセンス上の制約があることを指摘している。Bichler⁶⁾ は、ある VM やサービスを特定のサーバグループ内に配置しなければならないという制約を追加した問題を提起し、汎用サーバの適用結果を示している。これらの研究では、数十 VM の問題に厳密解法を適用すると、数時間から数十時間以上を要する場合があることが報告されている。

以上を整理すると、VM の静的な配置に関する研究は 4 つの軸で分類できる。第 1 に配置 (パッキング) アルゴリズムの種類、第 2 にリソース/次元数および時系列の考慮の有無、第 3 に、配置制約の有無やその種類、第 4 に、複数次元の問題に FFD 等の古典アルゴリズムを適用する場合は、アイテムや箱をソートするための順序関係の定義である。なお、動的な配置に関する研究に簡単に触れておくと、ワークロードや電力消費量、SLA を基に、独自に定義された profit 関数を最大化するようにリソース割当てを自動調整する 9) を始めとして、負荷分散を目的とした 22)–24) や、負荷予測に基づいて、静的な配置と同様、FFD による VM の集約を行なう 7) 等がある。

3. 開発したアルゴリズムの位置づけ

前節の関連研究を基に、本節ではサーバ統合用に我々が開発した改良パッキングアルゴリズム³⁾ の位置づけについて述べる。開発当時の 2005 年頃は第 1 節で述べた共有型のサーバ

統合を行なっており、問題としては、CPU を最重要リソースとしながらも複数のリソースと時系列を考慮する多次元のベクトルパッキング問題であった。統合対象の VM 数は少なくとも 100 以上あるため厳密解法は適用できず、また次元数が非常に多いため FFD²³⁴⁾ や PP, CP²⁶⁾ の適用も難しかった。このとき、もっとも単純なやり方は、優先的に考慮するリソースやリソース使用量に関する時系列データの最大値や合計値をもって VM をソートし、FFD や WFD 等でパッキングする方法である。しかし、Urgaonkar³⁵⁾ や Roy³⁰⁾ の結果が示しているように、各 VM のリソース消費量にばらつきがあると、空きリソースの断片化によって統合後の物理サーバ台数が増加してしまう問題があった。

そこで我々は、古典的なアルゴリズムに簡潔な再探索の仕組みを導入して最終的な箱の数を削減する改善法を開発した³⁾。この改良アルゴリズムを図 1 に示す。図 1 において、 s_1, \dots, s_n は n 個のアイテム、 X_1, \dots, X_m は m 個の箱、 $LB(\{s_1, \dots, s_n\})$ は与えられたアイテム集合に対して箱数の下限値を返す関数、 $pack(T)$ は集合 T 中のアイテムを m 個の箱に FFD や WFD 等を使って詰め、詰めるのに成功した場合は \emptyset 、失敗した場合は m 個の箱から最初に溢れたアイテムを T から削除して返す関数、 $MAXR$ はユーザの与える定数をそれぞれ表す。

簡単に説明すると、改良アルゴリズムはアイテムを格納する 2 つの順序つき集合 T' 、 T をもち、 T' のアイテムを優先的に詰める。初期状態では、アイテムはすべて T に所属する。箱数は 1 から始めてもよいが、計算時間の節約のために $\max([\sum_{i=1}^n a_{i,1}], \dots, [\sum_{i=1}^n a_{i,D}])$ のような簡単な下限値⁸⁾ から始め、既定数の箱にアイテムを詰められなければ、詰めるのに失敗した当のアイテムを T から T' に移動して、パッキングを最初からやり直す。ユーザの与える定数 $MAXR$ は、この試行回数に対応している。定数 $MAXR$ 回やっても詰められなかった場合は、箱を 1 つ増やしてから同様の処理を繰り返す。

我々の提案アルゴリズムと既存のアルゴリズムとの大きな違いは、詰めた結果に基づいてアイテムの順序を動的に入れ替えるところにある。複数のリソースや時系列を考慮した多次元のパッキング問題に FFD 等のアルゴリズムを適用する場合、どのリソースや時間帯の値を使ってアイテム (VM) や箱 (物理サーバ) の選択を行なえばよいかは、一意に決めることができない。なぜなら、アイテムの選択基準と、どの時間帯にどのリソースが競合するかは、相互依存の関係にあるからである。つまり、どの次元の値を重視すればよいかは、ヒューリスティックによってたまたま同一のサーバに集約される VM 同士が、どの時間帯にどのリソースを多く使っているかに依存するが、どの VM 同士が同一のサーバ箱に詰められるかは、どの次元の値を重視してアイテムや箱を選択したかに依存してしまう。さらに、

```
m ← LB({s1, ..., sn});  
while true do  
  T' ← {}; T ← {s1, ..., sn};  
  for r ← 1 to MAXR do  
    for j ← 1 to m do  
      Xj ← {} /* initialization */  
    end for;  
    s ← pack(T');  
    if s ≠ ∅ then  
      break  
    fi;  
    s ← pack(T);  
    if s ≠ ∅ then  
      T' ← T' ∪ {s};  
      continue  
    else /* all packed */  
      return m  
    fi  
  end for;  
  m ← m + 1  
end while
```

図 1 改良パッキングアルゴリズム

どの次元の値を重視すればよいかは、物理サーバごとに異なる可能性がある。

FFD 等の古典アルゴリズムの設計思想は、サイズの大きなアイテムを先に詰めてから、小さいアイテムを残った隙間に詰めることによって、最終的な箱の数を削減することにある。言い換えれば、サイズの大きなアイテムは、あとから詰めると先に詰めたアイテムと「調和」せず、箱の数を増やしてしまうアイテムとみなすことができる。提案アルゴリズムは、制約式の集合から矛盾する極小の部分集合 (minimal inconsistent subset) を求める長らのアルゴリズム¹¹⁾ から着想を得ている。矛盾する極小の集合とは、集合中の制約は充足

不可能 (unsatisfiable) であるが、集合中の任意の 1 制約を除くと、残りの制約が充足する (satisfiable) ような集合のことである。長らは、制約集合 c_1, \dots, c_n を先頭から順に処理していき、 c_k の処理中に矛盾が発生すると、制約の順序を c_k, c_1, \dots, c_n と入れ替えてから同様の処理を繰り返すことで、最終的に矛盾する極小部分集合を算出する。これは、 c_k の処理中に矛盾が発生した場合、 c_k が矛盾する極小部分集合の一部であり、かつ c_1, \dots, c_k の中に求める極小部分集合が必ず含まれていることを利用している。我々の改良アルゴリズムは、箱の数が増える契機となったアイテムを、本来なら重視すべきであった次元のサイズが大きいアイテムとみなし、これを他のアイテムよりも先に詰めることで、箱数の削減をはかっている。つまり、11) のことばでいえば、詰めるのに失敗したアイテムを他のアイテムとは 1 つの箱に収まらない「矛盾する」アイテムとみなしているわけである。

次元間の相関関係を考慮した問題のインスタンスを用いて、2 次元問題に対する基礎的な評価を行なった結果、我々の提案アルゴリズムは、単純な FFD や Least-Loaded (LL) に対し、サーバ台数を最大で約 28%削減することに成功している。LL は 負荷分散を想定した WFD と類似のアルゴリズムである。詳細については 3) を参照されたい。

4. まとめと今後の方向性

仮想マシンの配置問題に関して、オペレーションズリサーチの分野で研究されてきたベクトルパッキング問題の観点と、メインフレームやデータセンタの効率運用のために研究されてきたアプリケーション配置やワークロード配置、およびサーバ統合の観点から、2008 年頃までの関連研究を概観した。また、我々がサーバ統合用に開発したパッキングアルゴリズムの改善法を紹介し、その狙いや着想について述べた。提案した改善法に対し、3) において 2 次元のデータを用いた基礎的な評価を行なっているが、今後は時系列を考慮した多次元問題に対する評価や、PP, CP²⁶⁾ との比較、FFD²³⁴⁾ が用いている well-filled な箱を保存する手法が我々の手法に統合できるか等の検討がさらに必要と思われる。

サーバ統合の先にあるクラウドコンピューティング事業に目を向けると、近い将来、IaaS や PaaS 事業者が、VM のスペックや利用期間だけでなく、実際のワークロードに応じた従量課金モデルを導入する可能性が考えられる。米 Amazon の Vogels CTO³⁶⁾ は、Amazon が提供する EC2 の顧客が、ハードウェアリソースの取得サイクルが数ヶ月単位から分の単位に変化したと述べていることや、多くの会社の CIO が、サーバ統合によってリソースの余力が縮小したシステムでは、負荷変動に応じたワークロードの負荷分散が必要と考えていることなどを紹介している。これはつまり、ワークロードに応じた課金モデルへのニーズが

存在するが、そのような課金モデルの実現には、第1節で述べた共有型のVM配置や、負荷変動に応じた配置の見直しが必要となることを示している。我々はこの問題に対応するため、最小限のVMの移動で負荷の変動に対応する再配置アルゴリズムの開発²⁾や、再配置問題の厳密アルゴリズムに関する共同研究¹⁷⁾を行なっている。

また、今後クラウド環境で負荷や重要度の高いアプリケーションが稼働することを考えると、サーバ統合ではあまり考慮されていなかった物理マシン間のネットワーク機器や、物理マシンをまたいで共有される共有ストレージの負荷、さらには、データセンタの冷却能力の観点から、物理サーバから発せられる熱量等を考慮しなければならなくなるだろう。これらが本当に必要になるかどうかの予測は難しいが、現実のニーズをみずえたアルゴリズムの開発が今後の課題である。

参 考 文 献

- 1) Agrawal, S., Bose, S.K. and Sundarrajan, S.: Grouping genetic algorithm for solving the server consolidation problem with conflicts, *Proc. First ACM/SIGEVO Summit on Genetic and Evolutionary Computation (GEC'09)*, pp.1-8 (2009).
- 2) 網代育大：仮想マシンの組替えによる負荷変動への自律適応，人工知能学会第22回全国大会 (2008).
- 3) Ajiro, Y. and Tanaka, A.: Improving packing algorithms for server consolidation, *Proc. 33th Int. CMG Conference (CMG 2007)*, pp.399-406 (2007).
- 4) Bansal, N., Caprara, A. and Sviridenko, M.: Improved approximation algorithms for multidimensional bin packing problems, *Proc. 47th Annual IEEE Symp. on Foundations of Computer Science (FOCS'06)*, IEEE, pp.697-708 (2006).
- 5) Beck, J.E. and Siewiorek, D.P.: Modeling multicomputer task allocation as a vector packing problem, *Proc. Ninth Int. Symp. on System Synthesis (ISSS'96)*, IEEE, pp.115-120 (1996).
- 6) Bichler, M., Setzer, T. and Speitkamp, B.: Capacity planning for virtualized servers, *Proc. Workshop on Information Technologies and Systems (WITS 2006)* (2006).
- 7) Bobroff, N., Kochut, A. and Beaty, K.: Dynamic placement of virtual machines for managing SLA violations, *Proc. 10th IFIP/IEEE Int. Symp. on Integrated Network Management (IM'07)*, pp.119-128 (2007).
- 8) Caprara, A. and Toth, P.: Lower bounds and algorithms for the 2-dimensional vector packing problem, *Discrete Applied Mathematics*, Vol.111, pp.231-262 (2001).
- 9) Chase, J.S., Anderson, D.C., Thakar, P.N., Vahdat, A.M. and Doyle, R.P.: Managing energy and server resources in hosting centers, *Proc. 18th ACM Symp. on*

- Operating Systems Principles (SOSP 2001)*, ACM, pp.103-116 (2001).
- 10) Chekuri, C. and Khanna, S.: On multi-dimensional packing problems, *Proc. Tenth Annual ACM-SIAM Symp. on Discrete Algorithms (SODA'99)*, pp.185-194 (1999).
- 11) Cho, K. and Ueda, K.: Diagnosing non-well-moded concurrent logic programs, *Proc. Joint Int. Conf. and Symp. on Logic Programming (JICSLP'96)*, The MIT Press, pp.215-229 (1996).
- 12) Clautiaux, F., Carlier, J. and Moukrim, A.: A new exact method for the two-dimensional bin-packing problem with fixed orientation, *Operations Research Letters*, Vol.35, No.3, pp.357-364 (2007).
- 13) Coffman, Jr., E.G., Garey, M.R. and Johnson, D.S.: Approximation algorithms for bin packing: A survey, *Approximation Algorithms for NP-Hard Problems*, PWS Publishing, pp.46-93 (1996).
- 14) Dyckhoff, H.: A typology of cutting and packing problems, *European J. of Operational Research*, Vol.44, pp.145-159 (1990).
- 15) Epstein, L.: On variable sized vector packing, *Acta Cybernetica*, Vol.16, No.1, pp.47-56 (2003).
- 16) Fowler, R.J., Paterson, M. and Tanimoto, S.L.: Optimal packing and covering in the plane are NP-complete, *Information Processing Letters*, Vol.12, No.3, pp.133-137 (1981).
- 17) 福永アレックス，網代育大：仮想マシン組替えに対する厳密及びハイブリッドアルゴリズム，人工知能学会第23回全国大会 (2009).
- 18) Garey, M.R., Graham, R.L., Johnson, D.S. and Yao, A.C.: Resource constrained scheduling as generalized bin packing, *Combinatorial Theory Series A*, Vol.21, pp.257-298 (1976).
- 19) Gupta, R., Bose, S.K., Sundarrajan, S., Chebiyam, M. and Chakrabarti, A.: A two stage heuristic algorithm for solving the server consolidation problem with item-item and bin-item incompatibility constraints, *Proc. IEEE Int. Conf. on Services Computing (SCC '08)*, pp.39-46 (2008).
- 20) Hopper, E. and Turton, B. C.H.: A review of the application of meta-heuristic algorithms to 2D strip packing problems, *Artificial Intelligence Review*, Vol.16, No.4, pp.257-300 (2001).
- 21) Karp, R.M., Luby, M. and Marchetti-Spaccamela, A.: A probabilistic analysis of multidimensional bin packing problems, *Proc. Annual ACM Symposium on Theory of Computing*, ACM, pp.289-298 (1984).
- 22) Karve, A., Kimbrel, T., Pacifici, G., Spreitzer, M., Steinder, M., Sviridenko, M. and Tantawi, A.: Dynamic placement for clustered web applications, *Proc. 15th Int. Conf. on World Wide Web (WWW 2006)*, pp.595-604 (2006).
- 23) Khanna, G., Beaty, K., Kar, G. and Kochut, A.: Application performance manage-

- ment in virtualized server environments, *Proc. 10th IEEE/IFIP Network Operations and Management Symposium (NOMS 2006)*, pp.373–381 (2006).
- 24) Kimbrel, T., Steinder, M., Sviridenko, M. and Tantawi, A.: Dynamic application placement under service and memory constraints, *Proc. Fourth Int. Workshop on Experimental and Efficient Algorithms (WEA 2005)*, pp.391–402 (2005).
- 25) Kou, L.T. and Markowsky, G.: Multidimensional Bin Packing Algorithms, *IBM J. of Research and Development*, Vol.21, No.5, pp.443–448 (1977).
- 26) Leinberger, W., Karypis, G. and Kumar, V.: Multi-capacity bin packing algorithms with applications to job scheduling under multiple constraints, *Proc. Int. Conf. on Parallel Processing (ICPP'99)*, pp.404–412 (1999).
- 27) Lodi, A., Martello, S. and Vigo, D.: Recent advances on two-dimensional bin packing problems, *Discrete Applied Mathematics*, Vol.123, pp.379–396 (2002).
- 28) Rolia, J., Andrzejak, A. and Arlitt, M.: Automating enterprise application placement in resource utilities, *Proc. 14th IFIP/IEEE Int. Workshop on Distributed Systems: Operations and Management (DSOM 2003)*, LNCS, No.2867, Springer, pp.118–129 (2003).
- 29) Rolia, J., Cherkasova, L., Arlitt, M. and Andrzejak, A.: A capacity management service for resource pools, *Proc. Fifth Int. Workshop on Software and Performance (WOSP 2005)*, ACM, pp.229–237 (2005).
- 30) Roy, N., Kinnebrew, J.S., Shankaran, N., Biswas, G. and Schmidt, D.C.: Toward effective multi-capacity resource allocation in distributed real-time and embedded systems, *Proc. 11th IEEE Symp. on Object Oriented Real-Time Distributed Computing (ISORC 2008)*, pp.124–128 (2008).
- 31) Shachnai, H. and Tamir, T.: Approximation schemes for generalized 2-dimensional vector packing with application to data placement, *Proc. Sixth Int. Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2003)*, LNCS, No.2764, Springer, pp.165–177 (2003).
- 32) Shahabuddin, J., Chrungoo, A., Gupta, V., Juneja, S., Kapoor, S. and Kumar, A.: Stream-packing: resource allocation in web server farms with a QoS guarantee, *Proc. Eighth Int. Conf. on High Performance Computing (HiPC 2001)*, LNCS, No.2228, pp.182–191 (2001).
- 33) Spellmann, A., Erickson, K. and Reynolds, J.: Server consolidation using performance modeling, *IEEE IT Professional*, Vol.5, pp.31–36 (2003).
- 34) Spieksma, F. C.R.: A branch-and-bound algorithm for the two-dimensional vector packing problem, *Computers and Operations Research*, Vol.21, No.1, pp.19–25 (1994).
- 35) Uргаonkar, B., Shenoy, P. and Roscoe, T.: Resource overbooking and application profiling in shared hosting platforms, *Proc. Fifth USENIX Symp. on Operating Systems Design and Implementation (OSDI'02)*, ACM, pp.239–254 (2002).
- 36) Vogels, W.: Beyond server consolidation, *ACM Queue*, Vol.6, No.1, pp.20–26 (2008).