Vol.2009-DPS-140 No.15
2009/9/11

論理的なディペンデンシーを使ったサーバ統合

塩谷知宏[†]　加納真[†]　串田高幸[†]

移行元の物理サーバを仮想マシンとして移行先の物理サーバに集約するサーバ統合は，通常，移行先の物理サーバにおける計算資源の使用率が最大になるように行われる．サーバ統合の問題点は，移行先の物理サーバが故障したときに広がる被害の影響が考慮されていないことである．この論文では，移行先の物理サーバの故障の影響を定式化し，論理的なディペンデンシーを持つ場合にはサーバ統合の配置計画によって故障の影響に差異があることを示す．また，焼きなまし法を故障の影響を小さくするサーバ統合の配置計画の算出に適用した．サーバ統合の配置計画立案をビンパッキング問題として定式化した手法と，その手法とサーバ統合の配置計画を求める焼きなまし法を組み合わせた手法を比較評価し，焼きなまし法を用いると故障の影響をより小さく抑えたサーバ統合の配置計画を求めることができることを示した．

Server Consolidation based on logical dependencies

Tomohiro Shioya[†], Makoto Kano[†]
and Takayuki Kushida[†]

Server consolidation which consolidates physical source servers as virtual machines on target servers is generally executed so that computational resource usage of source servers can be maximum. A problem of existing method to assign source servers to target servers is that it doesn't consider failure effects of physical target servers. In this paper, we formulate physical effects of physical target servers and describe there are differences of failure effects between source server allocation when servers have logical dependencies. In addition, we applied simulated annealing method to calculate source server allocation which is lower failure effects. We compared the simulated annealing method which is combined with a method which formulates source server allocation planning as bin-packing problem, with the bin-packing method. We showed that the simulated annealing method calculates source server allocation which failure effect is lower.

1. はじめに

仮想化技術の発展によって，複数の既存の物理サーバを，一台の物理サーバ上で稼働する複数の仮想マシン（Virtual Machine : 以下，VM と呼ぶ）に集約するサーバの物理統合（以下、サーバ統合と呼ぶ.）が，広く実施されている．複数の VM がメモリや CPU といったサーバの物理資源を効率的に共有して, 稼働率を向上させることができるため，物理サーバの総数が削減され，運用コストが削減される．

サーバ統合では，物理サーバの台数削減によって運用コスト削減が期待される一方，サーバの物理的故障に伴う被害の範囲が増大するリスクがある．一台の物理サーバ上で複数の VM が稼働するため，ハードウェアやハイパーバイザーの不具合が発生すると，その物理サーバ上で動作している全ての VM が正常なサービスを提供できなくなる．また，多くの場合，複数のサーバ上のプログラムコンポーネント（以下、構成要素）が連携してサービスを提供している．このため，正常にサービスを提供できなくなった VM 上の構成要素に依存するため，他の正常な物理サーバ上の構成要素も正常にサービスを提供できなくなる．サーバ統合の配置計画が不適切であると，物理的故障の被害の影響範囲が拡大するリスクがあるが，従来のサーバ統合の配置計画手法では考慮されてこなかった．物理サーバの台数削減によるコスト削減をしても，物理サーバの故障の影響で被害が広範囲に広がると復旧に大きなコストが発生する. よって，故障の影響を小さく抑えたサーバ統合の配置計画を求めることが必要である．

本論文では，サーバの物理的な故障の影響について定式化し，構成要素間に論理的なディペンデンシーが存在する場合には，サーバ統合の配置計画が異なると故障の影響量に差異が生じることがあることを単純な例を用いて示す．次いで，焼きなまし法[6]を構成要素間の論理的なディペンデンシーに基づいて物理サーバの故障の影響が小さくなる配置計画の算出に適用する．また，サーバ統合の配置計画立案をビンパッキング問題として定式化した手法（以下，ビンパッキングによる手法と呼ぶ）と，ビンパッキング法とサーバ統合の配置計画を求める焼きなまし法を組み合わせた手法（以下，サーバ統合の焼きなまし法と呼ぶ）を比較評価することにより，有効性を示す．

2. 関連研究

サーバ統合において，移行元の物理サーバを，どの移行先の物理サーバ上の VM として稼働させるか，統合の配置を計画する必要がある．[2][7][10]は，サーバ統合の配置計画立案を,「移行先の物理サーバ上の物理資源使用率が閾値を超えないという制約下で，移行先の物理サーバの台数を最小化するように，移行元サーバの移行先サーバ

[†] 日本アイ・ビー・エム株式会社　東京基礎研究所

への配置を決定する」，というビンパッキング問題として定式化し，ヒューリステックな解法を提案した．ここで，移行先の物理サーバの物理資源使用率は，移行後の VM のパフォーマンスの指標であり，移行先の物理サーバの台数は移行後の運用コストの指標である．これらの論文は，サーバ統合の配置計画を求める研究として本論文と関係しているが，サーバ統合の配置計画を求める上での目的が異なる．これらの論文では，物理サーバの台数が少なくなるサーバ統合の配置計画を求めることが目的であるが，本論文は物理サーバの故障の影響が小さくなるサーバ統合の配置計画を求めることが目的である．サーバ統合の焼きなまし法は，ビンパッキングによる手法によって求めたサーバ統合の配置計画をもとにして，物理サーバの故障の影響がより小さい配置を求める．こうすることで，移行先の物理サーバの台数を小さく抑えたうえで，故障の影響が小さくなるサーバ統合の配置計画を求めていくことができる．

[9]は，個々の移行元のサーバの物理資源の使用率は時間軸で変動することに着目し，特定の時間幅での物理資源の使用の時系列データに基づいて，適切な配置計画を立案する手法を提案した．この論文では，物理サーバの故障の影響を考慮した配置計画は行っていない．サーバ統合の配置計画では，移行先の物理サーバの台数を小さくすることと共に，構成要素が論理的なディペンデンシーを持つことによる物理サーバの故障の影響が広がらないように配置するべきである．なぜならば，故障の影響を小さく抑えることで，故障による被害の拡大を抑え復旧コストが削減できるからである．

構成要素間の論理的なディペンデンシーを分散システムでのデバッグに利用した研究として，[1][3]がある．[1]は，システムの動作をメッセージレベルでトレースして得たデータを解析することで，構成要素間の論理的なディペンデンシーを把握し，パフォーマンスのデバッグに利用した研究である．[3]は，本研究と同じように構成要素間の論理的なディペンデンシーは他の手法によって得られることを前提として，分散システムでの根本原因分析に利用した研究である．これらは，論理的なディペンデンシーを用いている点で本研究とアプローチが似ているが，分散システムでのデバッグを目的としている点で本研究とは異なる．

サーバ統合では，物理サーバの台数を削減することと共に，物理サーバが故障したときに構成要素間に論理的なディペンデンシーがあると広がる被害を小さくする配置を求めることが必要である．物理サーバの台数を小さくするサーバ統合の配置計画はビンパッキングによる手法で求められる．ビンパッキングによる手法で求めたサーバ統合の配置計画から再配置を試みて，構成要素間の論理的なディペンデンシーに基づいて物理サーバの故障の影響が小さくなるサーバ統合の配置計画を求める．物理サーバの台数を小さくした上で，故障の影響を小さくする．計算が複雑でないビンパッキングによる手法で求めたサーバ統合の配置計画をもとにすることで，故障の影響が小さくなるサーバ統合の配置計画を求める時間が短くなる．

3. ビンパッキングによる手法

サーバ統合の焼きなまし法では，物理サーバの台数を少なく抑えたうえで，故障の影響が小さくなるようにサーバ統合の配置計画を求めていく．ビンパッキング問題として定式化してサーバ統合の配置計画を求める手法[2]は複数の計算資源を考慮してサーバ統合の配置計画を求めることができるので，サーバ統合の焼きなまし法の入力として適している．この手法は，大きいものを先に詰めてから小さいものをその隙間に詰めると少ない箱で済むというヒューリステックを用いたビンパッキング問題のヒューリステックな解法である First-Fit Decreasing (以下 FFD) を改良した改良 FFD である．FFD は，移行元サーバのひとつの計算資源について大きい順にソートし，その順番でパッキングをしていく．パッキングできる移行先の物理サーバがなくなったときに始めて，物理サーバの台数を増やすことで，少ない台数でのサーバ統合の配置計画を求める．しかし，ソートに使用していない計算資源の影響でパッキングできず，台数が増えてしまうという問題がある．改良 FFD では，この問題を解決するために，ソートに使用していない計算資源の影響でパッキングできなかった場合，そのサーバを優先してパッキングするように最初からやり直す．何回までやり直すかは，パラメータ MAXR で決まる．改良 FFD のアルゴリズムは，m 台の移行元のサーバの集合を $S = \{s_1, s_2, s_3, \cdots, s_m\}$，移行先の物理サーバ台数 n と配置 X_j (j=1,2,3,…,n) とすると，

Step 1. 移行元のサーバの計算資源の合計が，移行先の物理サーバの計算資源の閾値を越えない理論的な最小の物理サーバ台数を求め，n とする．
Step 2. T' を空集合，T を S で初期化する．
Step 3. r を 1 で初期化する．
Step 4. r < MAXR であれば，r に 1 を加える．そうでなければ，Step 10 から実行する．
Step 5. X_j (j=1,2,3,…,n)を空集合で初期化する．
Step 6. $s \in T'$ について順に，s を配置しても計算資源が閾値を越えない Xj に s を配置する．もし，$^{\exists} s \in T'$ が配置できなければ，Step 8 から実行する．
Step 7. $s \in T$ について順に，s を配置しても計算資源が閾値を越えない Xj に s を配置する．もし，$^{\exists} s \in T$ が配置できなければ，その s を T'に加えて，Step 4 から実行する．そうでなければ，Step 11 から実行する．
Step 8. n に 1 を加えて，Step 3 から実行する．
Step 9. 結果として X_j と n を出力して終了する．

最初から配置をやり直す回数 MAXR は移行元サーバの台数 m の 10～30%が適切である．

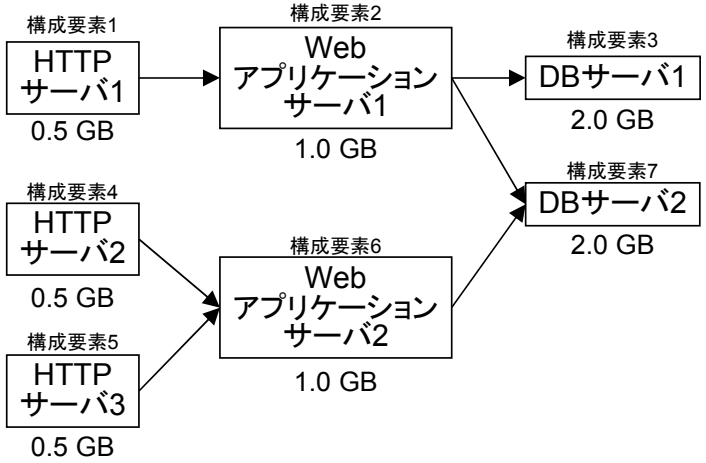

図 1　構成要素間の論理的なディペンデンシー
Figure 1　Logical dependency between configuration items

4. サーバ統合の配置計画における故障の影響の差異

4.1 サーバ上で動作する構成要素と構成要素間の論理的なディペンデンシー

サーバ上で動作するプログラムコンポーネントを構成要素と定義する. 例えば, Web アプリケーションサーバやデータベースサーバが構成要素である．通常，構成要素は単体で動作しているわけではなく，そのサーバ上で稼働している他の構成要素，他のサーバ上で稼働している他の構成要素と連携してユーザにサービスを提供する．構成要素 a が構成要素 b を利用している場合，構成要素 a は構成要素 b に対して論理的なディペンデンシーを有すると定義する．例えば，図 1 において Web アプリケーションサーバ 1 は DB サーバ 1 及び DB サーバ 2 を利用している．この場合，Web アプリケーションサーバ 1 は，DB サーバ 1 及び DB サーバ 2 に対して論理的なディペンデンシーを有するということになる．構成要素及び構成要素間の論理的なディペンデンシーを検出する手段としては，サーバ間のパケットをモニタリングする手法[1][4][5]，各サーバでスクリプトを実行し論理的なディペンデンシーを検出する手法[8]がある．サーバ統合の焼きなまし法では，構成要素及び構成要素間の論理的なディペンデンシーを入力とするが，検出手法に依らない．

サーバが物理的に故障すると，その影響は，そのサーバ上で動作する構成要素のみならず，構成要素間の論理的なディペンデンシーに沿って伝播する．図 1 において，DB サーバ 1 が故障すると，DB サーバ 1 に対して論理的なディペンデンシーを有する Web アプリケーションサーバ 1 が正常に動作しなくなり，さらに論理的なディペンデンシーに沿って影響が伝播し，HTTP サーバ 1 も正常に動作しなくなる．従って，構成要素間の論理的なディペンデンシーは，サーバの物理的故障の影響を考える上で必要な要素となる．

表 1　サービス構成グループと属している構成要素
Table 1　Service construct groups and configuration items which belong to .the service construct group.

サービス構成グループ	属している構成要素
g_1	HTTP サーバ 1，Web アプリケーションサーバ 1，DB サーバ 1，DB サーバ 2
g_2	HTTP サーバ 2，Web アプリケーションサーバ 2，DB サーバ 2
g_3	HTTP サーバ 3，Web アプリケーションサーバ 2，DB サーバ 2

4.2 構成要素のサービス構成グループ

複数の構成要素が，論理的なディペンデンシーを持って連携してユーザに一つのサービスを提供している場合，このような構成要素の集合を**サービス構成グループ**と定義し，それぞれ異なるサービスを提供していると仮定する．構成要素は少なくとも一つのサービス構成グループに属し, 複数のサービス構成グループに属することもある．論理的なディペンデンシーを持たない構成要素は，ひとつのサービス構成グループとして扱う．

図 1 では，表 1 のように 3 つのサービス構成グループ g_1，g_2，および g_3 が存在する．この複数のサービス構成グループを比較すると，提供しているサービスの種類によって，常に正常動作を期待されているサービスと，故障による停止をある程度許容できるサービスがある．本論文では，サーバの故障によるビジネスへの影響は，サービス構成グループごとに算出可能であると想定して, 図 1 のサービス構成グループ g_1，g_2，および g_3 の停止時の影響をそれぞれ I_1，I_2，I_3 と定義する．

4.3 サーバの物理統合とその配置計画

複数の既存物理サーバを VM として仮想化して，一台の物理サーバ上に集約するサーバ統合をサーバの物理統合と呼ぶ[11]．サーバの物理統合では，統合の前後で構成要素間の依存関係は維持されることが前提になる．一般に，一つのサーバ上では，複数の構成要素が動作しているが，本論文では簡単にするため，以下，一つのサーバ上

で一つの構成要素が動作しているケースを考える．即ち，サーバと構成要素を同義として取り扱う．

サーバの物理統合では, 複数の VM が, 移行先の物理サーバの物理資源を共有する．サーバの物理資源としては CPU，メモリ，ネットワークが挙げられる．図 1 で，各サーバ（構成要素）のノードの下に添えて記した数値が，各サーバのメモリ使用量を表す．サーバ統合の配置計画において，移行元のサーバの物理資源の使用量の総和が，移行先のサーバの物理資源の閾値を超えないように，サーバ統合の配置を決定する必要がある．ここでは，図 1 で表わされる移行元のサーバ群を，3 台の物理サーバ t_1, t_2, t_3 上に物理統合するケースを考える．t_1, t_2, t_3 の物理サーバのメモリ資源量はそれぞれ 3.0GB, 3.0GB, 2.0GB とする．図 2 は，二種類のサーバ統合の配置計画を表す．図 2(A)では，移行先サーバ t_1 に移行元サーバ s_1，s_2 を，移行先サーバ t_2 に移行元サーバ s_6，s_7 を，移行先サーバ t_3 に移行元サーバ s_1，s_4，s_5 を集約する配置を示している．またこの際に, 各移行先サーバ t_1, t_2, t_3 上での VM のメモリ資源の使用量の総和はそれぞれ 3.0GB，3.0GB，1.5GB であり，移行先サーバのメモリ資源量（( )内の値）を超えないように配置されている．

4.4 サーバ統合の配置計画によるサーバの物理的故障の影響の違い

物理サーバの故障やハイパーバイザーの故障が発生した場合，その物理サーバ上の全ての VM 及び，その VM 上で動作する構成要素が正常に動作しなくなる．さらに構成要素間の論理的なディペンデンシーに沿って，故障の影響が伝播する．構成要素間に論理的なディペンデンシーが存在する場合には，サーバ統合の配置計画によって故障の影響量に差異が生じる．

移行先の物理サーバがある一定期間の間に故障する確率を p とする．簡単のため，移行先物理サーバの故障確率は共通かつ独立とする．移行先の物理サーバの台数を M とすると，故障している移行先の物理サーバの組み合わせは 2^M-1 通りである．ある組み合わせが起こる確率を p_i とし，ある組み合わせが起きたときにサービス構成グループ g_i が正常なサービスを提供できるか否かを $x_{ij} \in \{0,1\}$ とすると, 移行先物理サーバの故障による影響度は式 1 で与えられる．

$$\sum_{i=1}^{2^M} p_i \sum_{j=1}^{Q} I_i x_{ij} \quad \cdots \text{式 1}$$

p を 0.01，I_1，I_2，I_3 をそれぞれ 30，20，10，とすると，図 2(A)のサーバ統合の配置計画の故障の影響度は 1.49, 図 2(B)のサーバ統合の配置計画の故障の影響度は 0.61 となり，図 2(B)の計画の方が，サーバの物理的故障の影響を低く抑えられていることが確認できる．

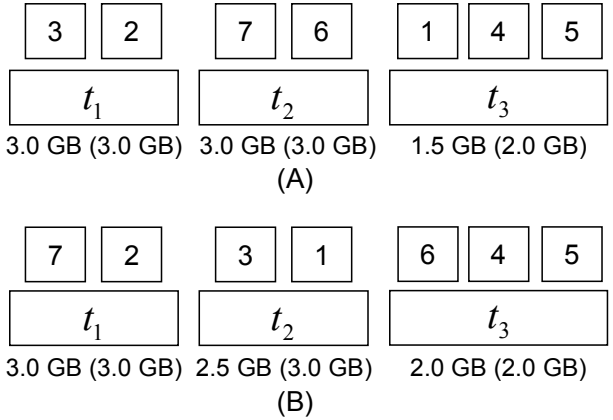

図 2　二種類のサーバ統合の配置計画
Figure 2　Two examples of plans of server consolidation

5. 故障の影響が小さいサーバ統合の配置計画

サーバ統合の焼きなまし法は，ビンパッキングによるサーバ統合の配置計画を初期状態とし，構成要素間の論理的なディペンデンシーに基づいて，被害の影響が小さくなるように計算してサーバ統合の配置計画を改善する．

5.1 入出力データ

サーバ統合の焼きなまし法に必要な与えられるべき入力データは, 1) 統合される移行元のサーバ(すなわち，VM)の数 N と移行先の物理サーバ数 M，2) 移行元の各サーバの CPU，メモリ，およびネットワーク使用量(s_cpu_i, s_mem_i, s_ntwk_i) (i=1,2,3,…,N) と移行先の各物理サーバの CPU，メモリ，およびネットワークの閾値(t_cpu_i, t_mem_i, t_ntwk_i) (i=1,2,3,…,M)，3) 構成要素間の論理的なディペンデンシー(CI_i，$dep(CI_i)$ (i=1,2,3,…N) 但し，$dep(CI_i)$は，構成要素 CI_i に直接依存している構成要素の集合を表す，4) サービス構成グループの重要度 w_i (i=1,2,3,…,Q)である．但し，サービス構成グループの数を Q とする．

3) 構成要素間の論理的なディペンデンシーは，構成要素に ID をふって，直接的に依存している構成要素の ID を列挙することで表現された表を与える. 図 1 を例にとり，入力として与える構成要素間の論理的なディペンデンシーの表を，表 2 に示す．

表 2　入力として与える構成要素間の論理的なディペンデンシー
Table 2　Logical dependencies of configuration items as an input data for proposed solution.

構成要素 ID	直接依存している構成要素 ID
1	2
2	3,7
3	-
4	6
5	6
6	7
7	-

4) は，構成要素間の論理的なディペンデンシーから自動的に作成されたサービス構成グループに対して手動で重要度を入力する．まず，表 2 のかたちで入力として与えられた論理的なディペンデンシーをもとにしてサービス構成グループを自動的に作成する．他の構成要素に依存されていない構成要素をルートとして，ルートから直接的および間接的に依存している構成要素をサービス構成グループとする．ルートの数は，サービス構成グループの数と同じ Q である．表 2 では，直接依存している構成要素の列に存在しない構成要素 1，4，および 5 がルートになる．それぞれのルートから表 2 をもとにサービス構成グループが作成される．そして，この作成されたサービス構成グループに対して手動で重要度を入力する．重要度とは，物理サーバの故障の影響を受けてサービス構成グループが正常なサービスを提供できなくなったときのインパクトの大きさである．重要度は，それぞれのサービス構成グループがサービスを提供できなくなったときの被害金額の割合により決める値で，整数値を取り，下限を 1，上限を 100 とする．表 2 から作成されるサービス構成グループと重要度の例を表 3 に示す．サービス構成グループ g_1，g_2，および g_3 がある期間サービスを提供できなくなったときの被害金額をそれぞれ 300 万円，200 万円，100 万円とし，重要度をそれぞれ 30，20，および 10 とした．

与えられた入力データからサーバ統合の焼きなまし法によって出力されるのは，サーバ統合の配置計画 X_i (i=1,2,3,…,M)である．

5.2 サーバ統合の焼きなまし法

サーバ統合の焼きなまし法では，M 台の物理サーバに集約するサーバ統合の配置計画 X_j (j=1,2,3,…,M) を状態と定義して，その状態における故障の重要度 d が小さくなるような次の状態へ遷移させることを繰り返す．
焼きなまし法は，確率的探索アルゴリズムである．現在の状態から次の状態の候補を生成し，決められた指標でみたときに改善された状態であれば，次の状態へ遷移させる．状態とは，あるサーバ統合の配置計画である．但し，ある確率で改悪された状態にも遷移させる．探索が進むにつれて小さくしていく値 T に依存して，改悪の状態へ遷移する確率を低くする．こうすることで，局所解に陥ることを減らしたアルゴリズムである．

表 3　サービス構成グループと重要度
Table 3　Service construct groups and degrees of importance.

サービス構成グループ	属している構成要素 ID	重要度
g_1	1，2，3，7	30
g_2	4，6，7	20
g_3	5，6，7	10

サーバ統合の焼きなまし法は，3 章で説明したビンパッキングによるサーバ統合の配置計画を初期状態とする．また，T の初期値を，T_START とする．同じ T において生成処理を行う回数を K とし，K 回の繰り返しを終えたときに T を更新する．指数型アニーリングを採用し，式 4 によってクーリング処理を行い，T を T'で更新する．指数型アニーリングにおいて γ は $0.5<\gamma<1$ が望ましい[12]．なぜならば，T を少しずつ小さくしていくことで，より最適解に近い解を求められるからである．

$$T'=\gamma T \quad \cdots \text{ 式 } 2$$

T が T_END 以下になったときに終了とする．

現在の状態から，次の状態の候補を生成する処理には，次の 2 つを用いる．1 つ目は，2 つの構成要素の入れ替え処理である．2 つ目は，構成要素を別の物理サーバへ移動する処理である．いずれの処理も，物理サーバの計算資源の値を越えない配置を次の状態の候補とする．これら 2 つの処理を 2 分の 1 の確率で適用し，次の状態の候補を生成する．

物理サーバの台数を M，サービス構成グループの数を Q とすると，影響度を計算する式 1 の計算量は $2^M Q$ である．例えば，M=100 のとき，式 1 の計算量は約 $1.16\times10^{30}\times Q$ となる．計算量が大きく時間がかかるので，ここでは式 1 を近似した式 2 により影響度 d を求める．式 2 は，サービス構成グループ g_i に属する構成要素がのっている物理サーバの数 l_i に故障確率 p と重要度 w_i をかけた値の和を求める．実際には，別々のサービス構成グループに属する構成要素が配置されている物理サーバもあるため，式 2 の計算結果は式 1 の計算結果よりも大きい値になる．しかし，式 1 よりも小さい計算量 Q であるので，影響度を早く求めることができるので良い．

$$d = \sum_{i=1}^{Q} l_i pw_i \quad \cdots \text{式 3}$$

生成された次の状態の候補を受理し，現在の状態を更新するかどうかの判定に Metropolis 基準を採用する．式 3 によって求められる確率 P(T)で，次の状態の候補を受理するかどうかを決定する．但し，現在の状態の影響度を d，次の状態の候補の影響度を d'とする．P(T)と，ランダムに生成した値 $y \in [0,1]$ を比べて，P(T) > y であれば受理する．

$$P(T) = \begin{cases} 1 & if\,(d'-d) > 0 \\ \exp(-\dfrac{d'-d}{T}) & otherwise \end{cases} \quad \cdots \text{式 4}$$

サーバ統合の焼きなまし法は焼きなまし法により，以下の Step でサーバ統合の配置計画を求める．

Step 1. T を T_START で初期化する．
Step 2. ビンパッキングによる手法で初期状態 X_j (j=1,2,3,…,M) を求める．X_j の影響度 d を求める．
Step 3. k を 1 で初期化する．
Step 4. 生成処理によって次の状態の候補 X_j'を求め，X_j'の影響度 d'を求める．
Step 5. P(T)と y によって X_j'を受理するか判定をし，受理するならば Step 5 から実行する．そうでなければ，Step 6 から実行する．
Step 6. X_j'を X_j とする．d'を d とする．
Step 7. k < K ならば，k に 1 を足し，Step 4 から繰り返す．そうでなければ，Step 8 から実行する．
Step 8. T にγをかけて T を更新する．
Step 9. T > T_END であれば，Step 3 から繰り返す．そうでなければ，Step 10 から実行する．
Step 10. 現在の状態 X_j を出力する．
Step 11. 式 1 により，X_j について正確な影響度 d を計算して出力し，終了する．

このアルゴリズムでは影響度を近似式によって求めているので，出力された結果に対して，式 1 により正確な影響度を計算する．

サーバ統合の焼きなまし法では，ビンパッキングによる手法で求めた物理サーバ台数を入力とせず，統合先の物理サーバの台数 M を入力から与えている．サーバ統合の焼きなまし法において次の状態の候補を生成するにあたって，M+α の台数で故障の影響が小さくなるサーバ統合の配置計画できるようにするためである．サーバの台数が増えると，生成される配置の組み合わせが増加するので，探索空間が広がり多くの中から故障の影響度が小さいものを探すことになり，良い結果が出やすくなる．

6. 評価

ビンパッキングによる手法で求めたサーバ統合の配置計画とサーバ統合の焼きなまし法で求めたサーバ統合の配置計画を比較した．

故障の影響度は，ビンパッキングによる手法よりもサーバ統合の焼きなまし法の方が小さく抑えられる．サーバ統合の焼きなまし法では，サービス構成グループがより少ない物理サーバにのる配置になると，故障の影響度を近似した式 2 の l_i が小さくなるので，故障の影響が小さい配置になる．ビンパッキングによる手法で求めたサーバ統合の配置計画を初期配置として改善していくので，より故障の影響度が小さい配置を求めることができる．

ビンパッキングによる手法は，物理サーバの台数を小さくする配置を求める手法である．一方，サーバ統合の焼きなまし法では，物理サーバの台数は入力として与えている．よって，移行元のサーバとその計算資源をもとに移行先の物理サーバの台数を小さくすることでコストを抑えたサーバ統合の配置計画を求めることができる．サーバ統合の焼きなまし法では，物理サーバの台数は入力として与えているので，ビンパッキングによる手法によって求まった物理サーバの台数を入力とすれば，物理サーバの台数を固定して，故障の影響が少ないサーバ統合の配置計画に改善することができる．よって，ビンパッキングによる手法の結果と，サーバ統合の焼きなまし法を組み合わせることで，物理サーバの台数と故障の影響が共に小さくなるサーバ統合の配置計画が求まる．

ビンパッキングによる手法では，繰り返し回数として MAXR を決める必要がある．一方，サーバ統合の焼きなまし法では，T_START，T_END，γ，K を決める必要がある．ビンパッキングによる手法はサーバ統合の焼きなまし法よりもパラメータ数が少なく，調整がしやすいという利点があるが，柔軟なチューニングをすることができない．サーバ統合の焼きなまし法は，パラメータがより多いのでチューニングに手間がかかるが，より詳細なチューニングが可能である．

サーバ統合の焼きなまし法は，次の状態の候補を上手く生成することで，より良い解を求めることができる．3 つ以上の構成要素を入れ替えたサーバ統合の配置計画を次の状態の候補として生成すると，構成要素の入れ替え方が増え，多くのサーバ統合の配置計画を探索できるようになる．

7. 議論

サーバ統合の焼きなまし法で求めたサーバ統合の配置計画が，ビンパッキングによる手法で求めたものよりどれ程改善されているかを定量的に示すことが必要である．

実験により定量的に比較するためには，ビンパッキングによる手法でも影響度を考慮したサーバ統合の配置計画が求められるように改良する．それによって，サーバ統合の焼きなまし法と定量的に結果を比較することができる．このためには，ビンパッキングによる手法の中で移行元のどのサーバをどの物理サーバに配置するかを，計算資源の閾値を越えないという制限の下で，物理サーバの台数を小さくすることと共に故障の影響を小さくすることを考えなくてはいけない．

サーバ統合の焼きなまし法では，4 つのパラメータを与える．構成要素間の論理的なディペンデンシーの特徴によって，これらのパラメータを調節する．構成要素間の論理的なディペンデンシーの特徴とは，例えば，構成要素間が密／粗にディペンデンシーを持っている，独立した構成要素が多い，またはルートが多く／少なくサービス構成グループの数が多い／少ないという特徴が考えられる．このような違い合ったパラメータを与えることで，より良いサーバ統合の配置計画が短い時間で求められる．構成要素間の論理的なディペンデンシーの入力とパラメータを変化させて実験を行ない，パラメータの値について考察を行う．

統合先サーバの台数が増加すると，サーバ統合の配置計画の組み合わせも増加するので，サーバ統合の焼きなまし法では生成される候補が増加する．よって，統合先サーバの台数を変化させたときに，サーバ統合の焼きなまし法によって求めたサーバ統合の配置計画にどのような傾向があるか解析する．

8. まとめ

サーバ統合は，資源を有効活用することによりコスト削減が期待できるが，同時に複数サーバをひとつの物理サーバで実行することによるリスクが発生する．このリスクは，構成要素が論理的なディペンデンシーを持ってサービスを提供していることによる．よって，構成要素間の論理的なディペンデンシーを考慮し，物理サーバの故障による被害の影響が小さくなるようにサーバ統合の配置計画を立案することが必要である．しかし，既存の研究では，物理サーバの故障による被害の影響については考慮されてこなかった．

本論文では，サーバ統合の配置計画によって物理サーバの故障による被害の影響が変化することを示した．また，故障による影響度を定義し，影響度が小さくなるサーバ統合の配置計画を求める手法を提案した．この手法は，ビンパッキングによる手法によって求められたサーバ統合の配置計画をもとに，焼きなまし法によってサーバ統合の再配置を行ない，影響度が小さくなる配置を見つける手法である．影響度がより小さいサーバ統合の配置計画が求まることで，物理サーバが故障したときの被害がより小さくなるため，運用・保守にとって有効性がある．

今後は，サーバ統合の焼きなまし法は，パラメータの値が大きく結果に影響する．そこで使用している各パラメータの値を変化させた実験により，より良い値を検討する．他には，構成要素間の論理的なディペンデンシーの特徴によるサーバ統合の焼きなまし法への影響を解析する．

参考文献

1) Aguilera, M.K., Mogul, J.C., Wiener, J.L., Reynolds, P. and Muthitacharoen, A.: Performance Debugging for Distributed Systems of Black Boxes, *Proc. SOSP'03*, pp74-89 (2003).
2) Ajiro, Y. and Tanaka, A.: Improving Packing Algorithms for Server Consolidation, *Proc. CMG'07*, pp399-406 (2007).
3) Brown, A., Kar, G. and Keller, A.: An Active Approach to Characterizing Dynamic Dependencies for Problem Determination in a Distributed Environment, *Proc. IM'01*, pp377-390 (2001).
4) Chen, X., Zhang, M. Mao, Z.M. and Bahl, P.: Automating Network Application Dependency Discovery: Experiences, Limitations, and New Solutions, *Proc. OSDI'08*, pp.117-130 (2008).
5) Kar, G., Keller, A. and Calo, S.: Managing Application Services over Service Provider Networks: Architecture and Dependency Analysis, *Proc. NOMS'00*, pp61-74 (2000).
6) Kirkpatrick, S., Gelatt, C. D. and Vecchi. M. P.: Optimization by Simulated Annealing, *Science*, No.220, pp.671-680 (1983).
7) Leinberger, W., Karypis, G, and Kumar, V.: Multi-Capacity Bin Packing Algorithms with Applications to Job Scheduling under Multiple Constraints, *Proc. ICPP'99*, pp404-412 (1999).
8) Magoutis, K., Devarakonda, M., Joukov, N. and Vogl, N.G.: Galapagos: Model-driven discovery of end-to-end application-storage relationships in distributed systems, *IBM J. RES & DEV.*, Vol.52, No.4, pp367-377 (2008).
9) Mehta, S. and Neogi, A.: ReCon: A tool to Recommend dynamic server Consolidation in multi-cluster data centers, *Proc. NOMS'08*, pp.363-370 (2008).
10) 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, *HiPC'01*, Vol.2228, pp182-191 (2001).
11) Spellmann, A., Erickson, K. and Reynolds, J.: Server Consolidation using Performance Modeling, *IT Professional*, Vol.5, pp31-36 (2003).
12) Van Laarhoven, P.J.M. and Aarts, E.H.L.: Simulated Annealing:Theory and Application, *Philips Research Laboratories* (1987).