Vol.2010-OS-113 No.2
2010/1/27

仮想マシンのオーバヘッドおよび負荷評価方法の提案

金田 典久[†]　飯塚 剛[†]　金木 佑介[†]
[†]三菱電機株式会社　情報技術総合研究所

仮想化によりサーバ統合する場合，仮想化のオーバヘッドやリソース競合があるためこの点を考慮してサイジングを行う必要がある．そこで統合前に取得した負荷データを統合後のサーバでシミュレーションして再生することにより性能評価を行い，サイジングおよび負荷評価の精度を向上する方法を提案する．

A Proposal of the Method of Overhead and Load Evaluation of Virtual Machine

Norihisa Kaneda[†], Tsuyoshi Iizuka[†] and Yusuke Kaneki[†]
[†]Information Technology R&D Center, Mitsubishi Electric Corporation

When servers are consolidated by the virtualization technology, it is necessary to do sizing in consideration of overhead and resource competition of the virtualization. This paper describes a proposal of the method of improving the accuracy of sizing and load evaluation by simulating with the load data collected from the servers.

1. はじめに

近年，企業内にある数百台のサーバをより少数の高性能サーバに集約するサーバ統合が注目されている．仮想化技術を利用することによって，複数のサーバで動作していたシステムを 1 つの物理サーバに集約して稼動することが可能となっており，消費電力，サーバコスト，運用管理コストの削減が期待されている．

仮想化環境では複数の仮想マシンが 1 台の物理マシン上で動作することにより，スケジューリングによるオーバヘッド，ディスクやネットワークアクセス時に発生する I/O（Input/Output）エミュレーションによるオーバヘッドなど，仮想化による負荷が発生する．また，各仮想マシンは，物理サーバの CPU，メモリ，ディスク，ネットワーク等のハードウェアリソースを時分割に使用するため，これらのリソースの競合が発生する．

したがって，仮想化のオーバヘッドやリソース競合を考慮したサーバ統合が必要であり，統合後の試験では実際の運用環境にできるだけ近い条件で性能を評価することが不可欠である．そこで，統合前物理サーバから取得した負荷データを元に仮想環境で負荷をシミュレーション再生することにより，実際の運用環境に近く，かつ効率的に性能評価を行うことができる方法について提案する．

以下，2 章では本方式を提案するに至った背景と課題について示す．3 章では本方式について説明する．

2. 背景と課題

2.1 仮想化によるサーバ統合の流れ

仮想化によるサーバ統合の一般的な流れを図 1に示す．

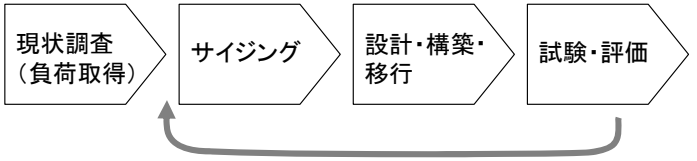

図 1　仮想化によるサーバ統合の流れ

まず，サーバ統合の対象となる既存サーバの現状調査として，統合前各サーバの負荷を取得する．負荷は，CPU，メモリ，ディスク容量，ディスクアクセス数，ネットワークアクセス数などを，ある一定期間（例えば数週間）取得する．

次に，取得した負荷データと統合後のサーバ仕様を元にサイジング（詳細は後述）を行い，統合後の物理サーバの台数を求める．

その後，統合後のサーバの設計・構築を行い，統合前サーバから統合後の仮想マシンへ移行を行う．

最後に試験や評価を行う．試験や評価でリソース不足や性能不足，逆に性能が余りすぎているなどの問題が発生した場合は，サイジングからやり直すことになる．したがってサイジングの精度を高めることが重要である．

2.2 サイジング

仮想化技術を利用したサーバ統合では，統合先のサーバに必要な性能や台数の決定，仮想マシンと物理サーバの組み合わせの算出が必要となる．このため，統合対象のサーバからサーバ負荷の統計情報を取得し，それを基に仮想化後のサーバ負荷を見積もる．

サイジングは，CPU 性能を元に行われることが多いが，CPU 性能によるサイジングでは，統合前サーバの CPU 負荷情報から CPU 使用量を求め，統合後の CPU 性能から統合台数を算出する．以下に算出方法を示す．

まず，統合前の各サーバの CPU 使用量を以下の式から求める．

$$\text{統合前}CPU\text{使用量} = \text{統合前サーバ}CPU\text{コア数} \times \text{周波数} \times \text{ピーク使用率}$$

統合後のサーバで使用可能な CPU 量は以下の式となる．

$$\text{統合後サーバ使用可能}CPU\text{量} = \text{統合後サーバ}CPU\text{コア数} \times \text{周波数} \times \alpha$$

ここで α は安全率であり，仮想化によるオーバヘッド（詳細は後述）などを考慮し通常 60%程度を指定する．統合後にサーバが何台必要かは，統合対象とした統合前サーバの値を集計し，以下の式から計算する．なお，CPU 使用率についてピーク時間が重ならないことが分かっている場合は，システム全体としてピークになる時間での各サーバの CPU 使用率を使用する．

$$\text{統合後サーバ台数} = \frac{\sum^{\text{統合前サーバ}}(\text{必要}CPU\text{使用量})}{\text{統合後サーバ使用可能}CPU\text{量}}$$

ただし，ここで求めた統合後サーバ台数では足りない場合がある．例えば，5 台のサーバを統合する場合に，それぞれの必要 CPU 使用量がすべて 2 とする．統合後サーバ使用可能 CPU 量を 5 とすると，統合後サーバ台数=2×5 台÷5=2 となり，2 台と計算できる．これは統合後サーバ 1 台あたり，統合前サーバの 2.5 台を割り当てたことになるが，サーバは分割できないため統合後のサーバには 2 台の仮想マシンしか実装できない．この場合は統合後のサーバは 3 台必要ということになる．

統合前のサーバで使用されている CPU と，統合後のサーバで使用する CPU のアーキテクチャが異なる場合は，動作周波数が同じでも性能が異なる場合がある．より精度よくサイジングを行うためにはアーキテクチャの違いによる CPU の性能差も考慮する必要がある．

2.3 仮想化オーバヘッド

従来から行われてきた非仮想化環境でのサイジング方式において，例えば複数のアプリケーションを 1 つのサーバで動作させる場合のサーバ負荷見積もりでは，各アプリケーションが使用するシステム負荷を合算することにより，サーバ全体のシステム負荷を見積もることができた．しかし，仮想化環境では，複数のサーバが 1 台の物理マシン上で動作することによる競合やスケジューリングによるオーバヘッド，ディスクやネットワークアクセス時に発生するI/Oエミュレーションによるオーバヘッドなど，仮想化による負荷がある．文献1)では，Xen[2]による仮想化のオーバヘッドを測定した結果が示されている．また，文献3)では，CPUリソース，ディスクリソースのオーバヘッドを，ベンチマークテストを用いて測定し，仮想マシンの稼働数と性能の関係や非仮想化の場合との性能比を算出し，サーバ統合時の性能設計や性能管理に役立てようという研究がなされている．このように，仮想化環境では仮想化オーバヘッドを考慮する必要がある．

I/O が多いシステムであればあるほど統合後の CPU 負荷は大きくなる．そのため，サイジングを行う際には，負荷の単純な合算ではなく，仮想化によるオーバヘッドを考慮した算出が必要となる．

2.4 リソース競合

各仮想マシンは，物理サーバの CPU，メモリ，ディスク，ネットワーク等のハードウェアリソースを時分割に使用するため，これらのリソースの競合が発生する．ある仮想マシンの動作が他の仮想マシンの動作に影響を与える場合がある．したがって，試験ではこれらリソースの競合についても評価することが重要となる．

2.5 課題

以上述べたように仮想環境では，仮想化のオーバヘッドおよび，リソース競合があるため，実際の運用環境にできるだけ近い条件で試験および性能評価を行うことが不可欠である．

統合前のサーバ環境すべてを統合後の仮想環境に移行し，実際のアプリケーションを実行すれば精度よく評価を行うことができる．しかし，統合前のサーバ環境を統合後の仮想環境に再構築するのは手間がかかるという問題があり，試験や評価の効率化が課題となる．

3. 負荷シミュレータによるサイジング評価方法

性能評価を精度よく，かつ効率的に行う方法として，統合前サーバから取得した負荷データをシミュレーション再生することにより性能評価を行う方法を提案する．

図 2に負荷シミュレータを使用する場合のサーバ統合の流れを示す．2.1節で示した従来の方法では，試験・評価の段階で性能問題が発生すると，設計・構築・移行を何度も繰り返す必要がある．負荷シミュレータを使用することにより実際のサーバ移行の前に十分評価が行えるため，最後のフェーズの試験・評価での手戻りを少なくすることが可能となる．

図 2　負荷シミュレータによるサーバ統合

以下，負荷シミュレータについて述べる．負荷シミュレータは負荷データを元に，負荷をシミュレートして再生する機能である．ここでは，Xen を使用した場合の例として示す．

3.1 負荷データ収集

負荷データは，統合対象となる統合前サーバからLAN経由で定期的に収集する．図 3を参照．

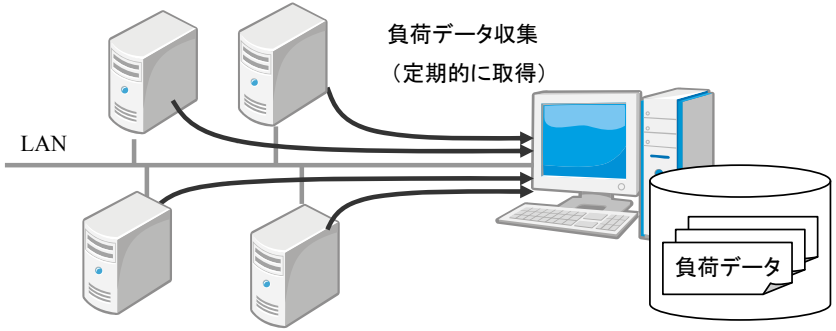

図 3　負荷データ収集

統合対象のサーバ OS が Linux 2.6 系の場合は，負荷取得対象サーバの/proc/stat, /proc/diskstats, /proc/ meminfo, /proc/net/dev ファイルから負荷データを採取することが可能である．これらのファイルを一定時間毎に読み込み，値の差分が負荷データとなる．今回は 1 秒毎に読み込むこととしている．

サーバの OS が Windows 系の場合もパフォーマンスデータから同様の負荷データを収集可能である．

3.2 負荷シミュレーション（ゲスト内で実行）

3.2.1 構成

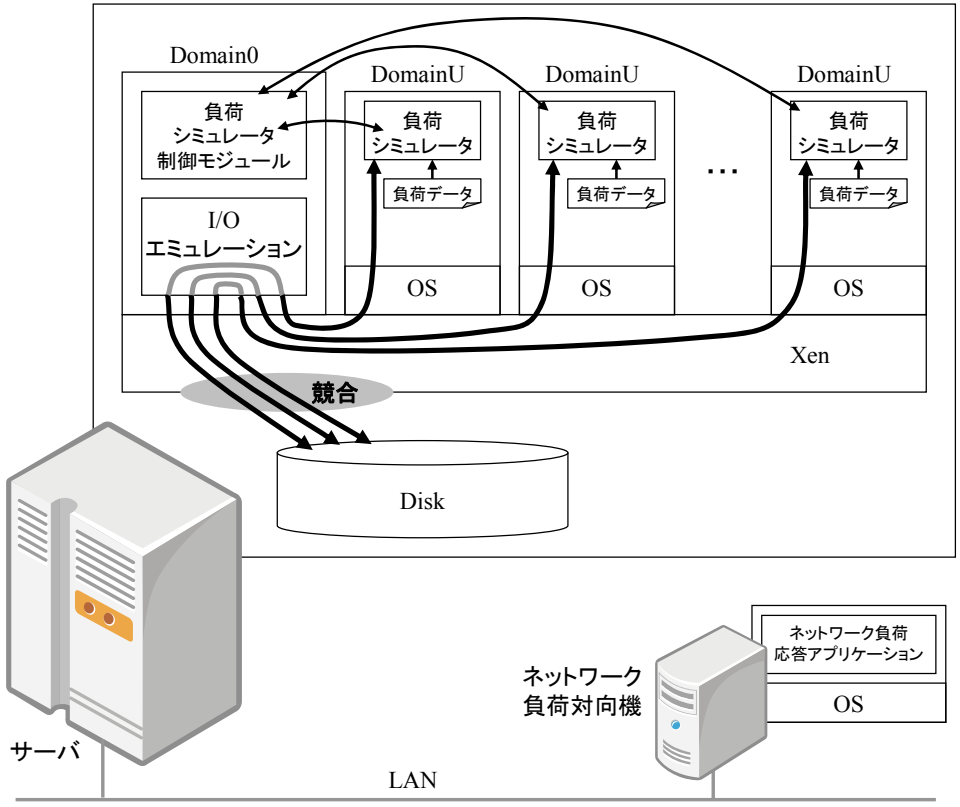

図 4　ゲスト内での負荷シミュレーション構成

図 4にゲスト内で負荷をシミュレーションする場合の構成を示す. 統合後のサーバ内にXen環境を構築し，統合前サーバに相当するゲストのDomainUを作成する．Domain0 に負荷シミュレータの制御用モジュール，各DomainU内に負荷シミュレータのモジュールを実装する．なお，ネットワーク負荷シミュレーション用に外部に対向機を用意しておき、対向機内にネットワーク負荷応答アプリケーションを動作させておく．

3.2.2 動作

各 DomainU 内の負荷シミュレータは, Domain0 の負荷シミュレータ制御モジュールからの指示により実行を開始する．負荷シミュレータの詳細動作について次に示す．

負荷のシミュレーションは，一定時間毎の時系列で記録された負荷データファイルを読み込み，そのデータにしたがって一定時間負荷をシミュレーションする．これを 1 サイクルとし, 1 サイクル終了後, 次の負荷データを読み込み次の負荷をシミュレーションする．今回は負荷データの収集を 1 秒毎に行っているため，負荷シミュレーションの 1 サイクルも 1 秒となる．以下，1 サイクル 1 秒として説明する．

負荷シミュレーションのフローチャートを図 5に示す．CPU負荷，ディスク読み込み負荷，ディスク書き込み負荷，ネットワーク送信負荷，ネットワーク受信負荷はそれぞれ別のスレッドとし，同時に実行する．CPU負荷については，仮想CPU数毎にスレッドを生成する．メモリ負荷については，1 サイクルの期間中使用し続けるシミュレーションとするため，メインスレッド内でサイクルの初めにメモリ割り当てを実行することとする．各負荷スレッドはシグナルにより 1 サイクル毎に同期実行される．

メインスレッドでは，各スレッド作成後，Domain0 上に実装されている制御モジュールからの指示待ちとなる. 複数の仮想マシン間での同期を取るため, 開始は Domain0 上の制御モジュールからの指示で行う. 制御モジュールから開始の指示を受信すると，現在時刻を取得し，負荷データを読み込む．各スレッドに対し負荷を設定し，各スレッドにシグナルを送り，各負荷を実行させる．メモリ負荷を生成した後，現在時刻を取得し，次のサイクル実行時刻まで sleep する．

各スレッド内では，設定された負荷を生成して実行する．負荷実行終了時刻を記録し，次のサイクルの実行待ちに戻る．

メインスレッドでは sleep 終了後，メモリ負荷を解放する．各スレッドの実行終了時刻をチェックし，結果を出力する．

各項目の負荷シミュレーション方法について，以下に示す．

(1) CPU 負荷

スピンループを指定された負荷に応じた回数実行する. 統合前サーバの CPU 性能と負荷データからスピンループ回数を決める．

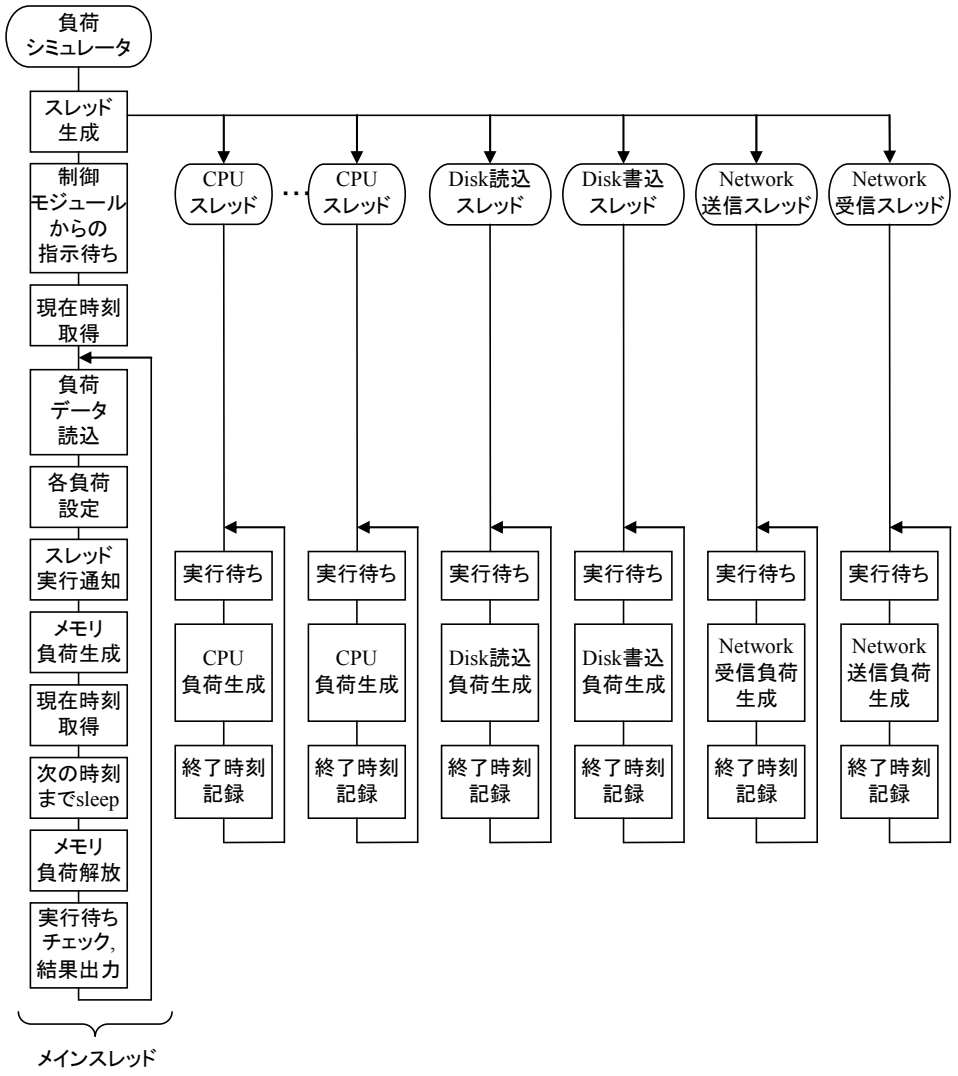

図 5　負荷シミュレータフローチャート

(2) **メモリ負荷**
サイクルの最初に指定されたメモリ容量を割り当てる．サイクル終了時にメモリを解放する．ただし，メモリを割り当てただけでは実際にメモリが使用されないため，例えばメモリブロックの先頭のみデータを書き込むなどの処理を行う．

(3) **ディスク読み込み負荷**
指定されたディスク読み込み量のデータをディスクから読み込む．読み込むためのデータはあらかじめディスクに作成しておく．ただし，実際に物理ディスクからの読み込みが行われるようにするため，以前に読み込んだ場所とは違う場所のデータを読み込むようにする．

(4) **ディスク書き込み負荷**
指定されたディスク書き込み量をディスクに書き込む．実際に物理ディスクへの書き込みが行われるように同期書き込みとする．

(5) **ネットワーク送信負荷**
ネットワークの送受信には相手先が必要となるため，外部の対向機としてネットワーク負荷サーバを用意し，ネットワーク負荷対向用応答アプリケーションを実装する．指定されたネットワーク送信量をネットワーク負荷サーバへ送信する．ネットワーク負荷サーバへの送信のみでありネットワーク負荷サーバからの送信はない．

(6) **ネットワーク受信負荷**
まず，指定されたネットワーク受信量の値をネットワーク負荷サーバへ送信する．ネットワーク負荷サーバの応答アプリケーションは，この値のデータ量を，負荷シミュレータに向けて送信する．負荷シミュレータはこのデータを受信する．

3.2.3 負荷評価
負荷シミュレータを実行した結果，1 秒以内に CPU，ディスク，ネットワークそれぞれの処理が完了するかをチェックする．もし，1 秒以内に終わらなければ，負荷が高すぎることを意味する．つまり，統合数が多すぎたことを意味する．統合サーバ数を減らすか，ハードウェアを増強するかが必要である．また，1 秒間の処理内でアイドル状態が多ければ，サーバ統合数を増やすことができる．

3.3 負荷シミュレーション（VM管理ドメイン上で実行）
Xenの場合Domain0 上で負荷シミュレータを動作させることにより，仮想マシン内でシミュレートせず実行することも可能である．図 6に構成を示す．負荷シミュレータは，Domain0 上で複数のアプリケーションとして動作させる．Domain0 上で負荷シミュレータを動作させる場合，仮想マシンを起動する必要がないため手間を少なくすることが可能である．

ただし，Domain0 上で動作させる場合，リソース競合の評価は可能であるが，仮想化のオーバヘッドが加味されなくなってしまうため，負荷評価が不正確になる．そこで，CPU 負荷をシミュレートする際，仮想化オーバヘッドも考慮して行う．シミュレートする CPU 負荷は，負荷データファイルに指定された CPU 使用量に，I/O 使用量に応じたオーバヘッド分を加えたものとなる．以下の式で，CPU 負荷を補正する．

$$
\begin{aligned}
\text{生成}CPU\text{負荷} = {} & CPU\text{負荷} \\
& + \text{ディスク書き込み量} \times \text{ディスク書き込み負荷係数} \\
& + \text{ディスク読み込み量} \times \text{ディスク読み込み負荷係数} \\
& + \text{ネットワーク送信量} \times \text{ネットワーク送信負荷係数} \\
& + \text{ネットワーク受信量} \times \text{ネットワーク受信負荷係数}
\end{aligned}
$$

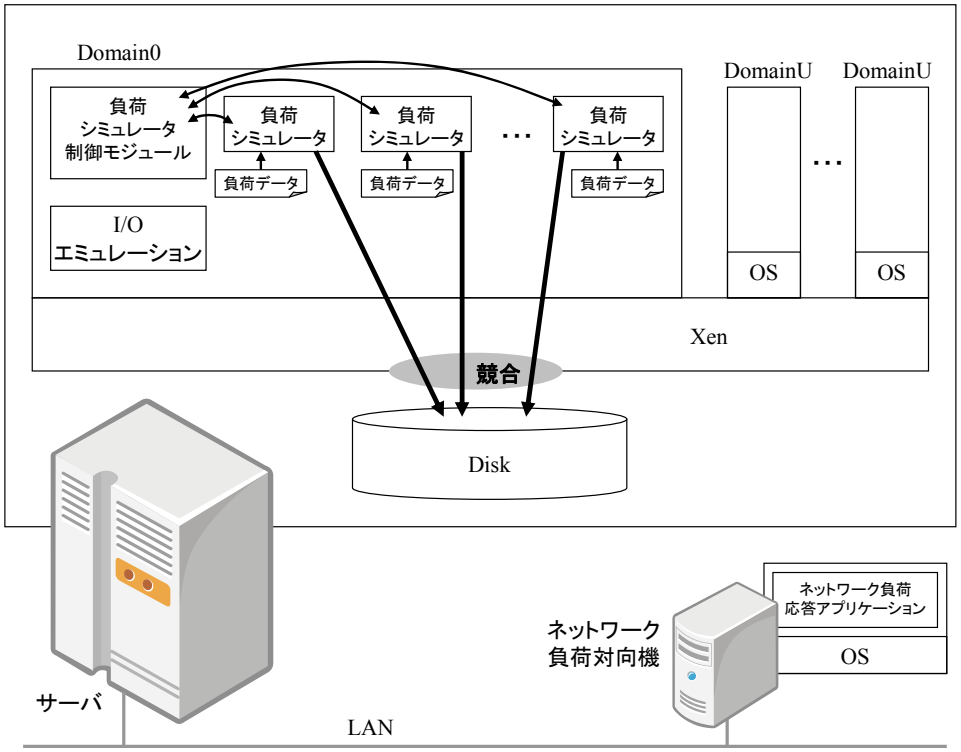

図 6　Domain0 内での負荷シミュレーション構成

ここで，各負荷係数はあらかじめ求めておく必要がある．文献1)での測定のようにあらかじめ，各I/O量とそのときのCPUオーバヘッドの関係を調査しておき，その値を負荷係数とする．

Domain0 内でシミュレーションを実行する場合，Xen のスケジューリングの違いなどにより DomainU 内でシミュレーションを実行するときと比較すると実際の運用環境との差は大きくなる．しかし，仮想マシンの構築および起動が必要なく，評価の手間を少なくすることができ，評価がより効率化される．

4. おわりに

仮想化によるサーバ統合では，仮想化のオーバヘッドおよび，リソース競合があるため，実際の運用環境にできるだけ近い条件で試験および性能評価を行うことが不可欠である．統合前のサーバ環境すべてを統合後の仮想環境に移行し，実際のアプリケーションを実行すれば精度よく評価を行うことができるが，統合前のサーバ環境を統合後の仮想環境に再構築するのは手間がかかるという問題があり，試験や評価の効率化が課題となっていた．

本課題を解決するため，統合前のサーバから取得した負荷データを統合後のサーバでシミュレートすることにより，統合前のサーバ環境をすべて移行して仮想環境を構築する前に，性能評価を行う方法を提案した．今後は，実システムへの適用に向けた本方式の評価を行う予定である．

参考文献

1) Ludmila Cherkasova, Rob Gardner: Measuring CPU Overhead for I/O Processing in the Xen Virtual Machine Monitor, 2005 USENIX Annual Technical Conference pp 387-390
2) Paul Barham 他 Xen and the Art of Virtualization
3) 網代育大, 田中淳裕. 仮想計算機環境における資源管理オーバヘッドの評価. 情処システム評価研究会研究報告(EVA-17), pp. 17-22, 2006