Vol.2010-OS-115 No.14
2010/8/4

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

広渕崇宏[†1] 中田秀基[†1]
伊藤智[†1] 関口智嗣[†1]

データセンタの資源利用効率を高めるため，資源消費量に応じた動的な仮想マシン（VM）再配置に関心が高まっている．仮想マシンに対して一定量の計算資源を割り当てつつも，資源消費量が少ないときには仮想マシンを集約して配置し運用効率を高める．しかし，これまで研究されてきた仮想マシン集約システムは，仮想マシンの移動に時間を要するプレコピー型ライブマイグレーション技術を使用している．仮想マシンの突発的な負荷上昇が発生すると，過負荷状態を素速く解消することが難しく，長時間にわたる仮想マシンの性能低下を招いてしまっていた．そこで，我々は，迅速な実行ホストの切り替えを可能とするポストコピー型ライブマイグレーション技術を，仮想マシン集約システムに対して適用することを新たに提案する．約一秒程度で仮想マシンの実行ホストを切り替えられるため，迅速に配置状態を最適化できる．突発的な負荷上昇が発生しても，仮想マシンの性能低下を短時間にとどめられる．評価実験を通して，提案機構のプロトタイプシステムが正しく動作することを確認した．ポストコピー型マイグレーションを利用したシステムは，プレコピー型を利用したシステムよりも，過負荷状態を迅速に解消し性能低下を軽減できた．

Dynamic Virtual Machine Packing Mechanism with Instantaneous Live Migration

Takahiro Hirofuchi ,[†1] Hidemoto Nakada ,[†1]
Satoshi Itoh [†1] and Satoshi Sekiguchi [†1]

Live migration of virtual machines is promising technology for IaaS (Infrastructure-as-a-Service) datacenters. By exploiting live migration, it is possible to realize over-commit assignments of virtual machines for improving resource usage. If virtual machines are not consuming their assured amounts of computing resources, these virtual machines are migrated into a fewer physical nodes to reduce running physical nodes. Existing virtual machine packing systems, however, incur large performance penalties if virtual machines are overloaded. Pre-copy live migration mechanisms, which are used in these packing systems, cannot relocate virtual machines quickly into new physical nodes. Overloaded states are not resolved in a short period, resulting in serious violation of performance agreement. In this paper, we propose an advanced virtual machine packing system that exploits a post-copy live migration mechanism. The execution host of a virtual machine is instantaneously switched to a new host just in one second. Virtual machines on an overloaded physical node are quickly rebalanced to other nodes thereby reducing performance degradation. Through experiments, we confirmed that our prototype system correctly worked; in comparison with a conventional virtual machine packing system based on pre-copy migration, our prototype system greatly alleviated performance degradation on sudden load rises.

1. はじめに

IaaS (Infrastracture-as-a-Service) データセンタの運用効率を高めるために，仮想計算機のライブマイグレーションを利用した，資源利用効率の向上に注目が集まっている．仮想マシンに対する資源割当量を保証しつつも，実際の資源消費量が小さければ，資源割当量から導かれる数よりも多数の仮想マシンを一つの物理ノードに対して配置（*overcommit*）する．また，仮想マシンの資源消費量が大きくなれば，約束した資源割当量を確保すべく仮想マシンを多数の物理ノードへ分散させる．顧客に対して一定の仮想マシン性能を提示しつつも，ハードウェア資源の稼動効率を向上させることで，一歩進んだ省エネルギー・効率化が可能になる．

しかし，これまで研究されてきた仮想マシン集約システムは，いずれもプレコピー型のライブマイグレーション[1)–3)] に基づいている．今日広く使われている仮想マシンモニタで標準的に利用できるマイグレーション機能であるものの，仮想マシンの実行ホストの切り替えに時間がかかり，また移動完了を見積もることが難しい．突発的な負荷上昇が発生すると，過負荷状態の解消に時間がかかるため，長時間にわたり仮想マシンの性能が低下してしまう．仮想マシンの集約を積極的に行うことが困難であった．

我々はこの問題を解決すべくポストコピー型のライブマイグレーション機構を開発してきた[4)]．仮想マシンのメモリページのコピーを実行ホストの切り替え後に行うことで，仮想マシンの実行ホストの切り替えを約 1 秒程度で可能にする．メモリページをコピーする際に

†1 産業技術総合研究所 / National Institute of Advanced Industrial Science and Technology (AIST)

は，重要領域からコピーすることで性能低下を抑制できる．またメモリデータを含めた仮想マシン全体のステートを再配置する時間も，プレコピー型よりも短くそして常に一定であるという利点がある．これまでの先行発表ではこのマイグレーション機構自体を話題としてきた．その応用である仮想マシン集約システムについては構想を述べた[5] にとどまってきた．

そこで本稿では，迅速な実行ホストの切り替えを可能とするポストコピー型ライブマイグレーションを，仮想マシン集約システムに適用することを提案する．仮想マシンの資源消費量をリアルタイムに監視しながら，過負荷状態を検知すると高速マイグレーションにより仮想マシンを分散配置する．仮想マシンの資源消費量が減少すると再度仮想マシンを集約配置する．仮想マシンの負荷にかかわらず常に短時間で仮想マシンの実行ホストを切り替えられるため，急激な負荷上昇が発生しても即座に過負荷状態を解消できる．プレコピー型およびポストコピー型ともにマイグレーションを行う仮想マシン自体にオーバーヘッドを伴う点は同様であるものの，極めて短時間で過負荷状態を解消できるポストコピー型の方が，マイグレーションする仮想マシンおよび過負荷状態を共有する仮想マシンに対する影響を小さくできる．

さらに，本稿での取り扱い範囲を越える発展的な観点では，仮想マシンの性能低下を抑制してできる限り性能を保証したいという要求と，仮想マシンを集約してできる限り稼働物理ノード数を削減したいという要求という，相矛盾する二つの要求を実環境においてより高いレベルで満たすことができるのは，ポストコピー型のマイグレーションであると考えている．

以降，2 節で既存手法の問題点を述べる．3 節で提案手法を説明し，4 節で我々の仮想マシン集約システムの概要を述べ，5 節で実験結果をまとめる．6 節で関連研究を説明し，7 節で本稿をまとめる．

2. 仮想マシン集約システムにプレコピー型マイグレーションを適用することの問題点

仮想マシンの動的な再配置機構においては，仮想マシンの処理性能を保証しながら，いかに集約率を高めるかという点が課題となる．仮想マシンのマイグレーションは，その動作自体が CPU やネットワーク資源を消費し，なおかつ時間を要してしまう．仮想マシンの資源消費量が少ないときに仮想マシンを集約しすぎると，負荷が上昇した時に仮想マシンを分散配置するコストが大きくなり，保証した仮想マシンの処理性能を下まわる期間が長くなってしまう．一方，分散再配置するコストを憂慮する余り，実際の資源消費量に応じた仮想マシンの集約を積極的に行わなければ，運用効率は向上しない．

既存研究においては，近い将来の資源消費量を予測して再配置を実行する手法[6] や，マイグレーションのコストを算入した配置決定手法[7] 等が提案されてきた．いずれも一般に利用できるプレコピー型の再配置機構を仮定している．また，我々のプロトタイプシステム[8] においても，当初はプレコピー型の再配置機構を利用してきた．

しかし，実際には，プレコピー型の再配置機構は仮想マシンの動的な再配置機構に適していないと考えている．プレコピー型は，実行ホストの切り替えに長時間（$MemorySize\ /\ NetworkSpeed + \alpha$）を要してしまう．仮想マシンのメモリサイズが 1GB であれば，GbE ネットワークを利用して少なくとも 10 秒程度かかる．さらに，実際には更新ページを再帰的にコピーするために，所要時間はメモリの更新速度に依存して増加する（α 部分）．マイグレーションの完了時間が一定ではなく，再配置計画を予定時間内に完了する保証がない．結果，既存システムではプレコピー型の再配置機構を採用しているがゆえに，急激な負荷上昇時において仮想マシンの処理性能を保証することが難しい．

顧客によってさまざまな利用がなされる IaaS サービスにおいて，個々の仮想マシンの負荷上昇を事前に予想することは一般的には困難であり，急激な負荷上昇時のペナルティが大きいことは好ましくない．一部のウェブサーバ等においては負荷変動に周期性があることが知られており，ある程度は事前に負荷の上昇を予測して配置状態を変更することはできる．しかし，一時間程度の時間粒度での不確実性をともなった予測であり，急激な負荷上昇時の問題を緩和こそすれど本質的に解決するものではない．

3. 提　　案

そこで，我々は仮想マシンの動的な再配置機構に対して，ポストコピー型マイグレーションを適用することを提案する．ポストコピー型再配置は，仮想マシンのメモリページのコピーを実行ホストの切り替え後に行うことで，仮想マシンの実行ホストの切り替えを約 1 秒程度で可能にする．急激な負荷上昇時には仮想マシンの実行ホストを短時間で切り替えることで，すぐさま過負荷状態を解消できる．

ポストコピー型マイグレーションについては，我々の実装を紹介した文献[4] で詳細を述べている．仮想マシンの実行状態の大半を占めるメモリページデータのコピーを，実行ホスト切り替え後に遅延する点が特徴である．基本的な動作としては，マイグレーション開始直後に移動元で仮想マシンを停止し，CPU レジスタとデバイス状態（最小 140KB 程度）を移動先にコピーする．そしてすぐさま仮想マシンを移動先で再開する．その後，仮想マシン

がメモリページにアクセスしようとすると，移動元からメモリページをオンデマンドに取得する．オンデマンドなメモリページ取得動作と平行して，メモリアクセスが頻出する領域から残りのメモリページも取得する．最終的にすべてのメモリページがコピーされると，移動元に依存性がなくなり，マイグレーションが完了する．

仮想マシン集約システムに対する利点としては，

- 一秒程度での実行ホスト切り替えが可能である．急激な負荷上昇に対応できる．
- メモリデータを含めた仮想マシン全体のステートを再配置する時間も，更新ページの再送が不要なためプレコピー型よりも短くそして常に一定である．所要時間は $MemorySize / NetworkSpeed$ である．再配置計画が予定時間内に必ず完了する．
- マイグレーションにともなうネットワーク転送量も，更新ページの再送が不要なためプレコピー型よりも短い．ほぼ仮想マシンのメモリサイズと同等である．マイグレーションにともなうネットワーク資源消費量が少ない．

という点が挙げられる．一方，欠点には，

- 実行ホスト切り替え後からメモリコピーを行うために，未だ転送されていないメモリページにアクセスすると性能が若干低下してしまう．

という点がある．しかし，我々は利点の方が欠点を補ってあまりあるのではないかと考えている．

これらポストコピー型マイグレーションの特徴が仮想マシン集約システムに対して有効に働くのかという点を，プレコピー型との比較において議論することが本研究の主題である．本研究の構成は，

- 最初に負荷上昇時の挙動について焦点を当てる．研究の土台となる仮想マシン集約システムの基本的な動作も確認する．
- 以後の研究発表にて，仮想マシン集約システムにおける性能保証と集約率向上の達成度を評価し，ポストコピーの優位性を詳しく議論する．

という 2 段階を予定している．本稿では前者を取り扱い，後者は今後の課題とする．

4. 仮想マシン集約システム

本研究で用いる仮想マシン集約システムについて述べる．概要を**図 1** に示す．資源消費量モニタ，再配置計画エンジン，VM コントローラから成る．資源消費量モニタでは，仮想マシンおよび物理ノードの資源消費量を一秒ごとに収集しデータベースに記録する．再配置計画エンジンでは，データベースを参照して再配置の必要性を検知すると，新たな仮想マシンの配置状態を決定する．VM コントローラは，新たな配置状態となるように仮想マシンのマイグレーションを実行する．

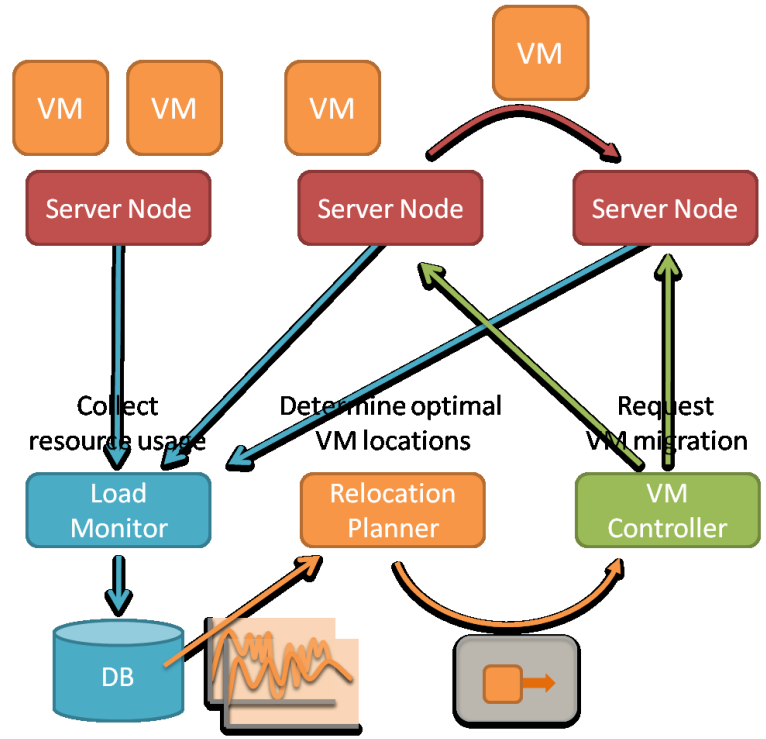

図 1 仮想マシン集約システムの概要

監視対象の資源消費量としては，CPU，メモリ，ディスク I/O，ネットワーク I/O を予定している．現状では，物理ノードおよび仮想マシンの CPU 使用率をホスト OS 上で取得している．

再配置計画エンジンとしては，遺伝的アルゴリズムおよび線形計画法による最適化手法[9),10)]を開発した．本稿では，過負荷状態の解消動作に焦点を当てるためにそれらを用いていない．代わりに単純化したアルゴリズムを用いる．一定時間以上，物理ノードの CPU 使用率が閾値を上まわると分散配置を，下まわると集約配置を行う．

VM コントローラでは，KVM[11] 本来のプレコピー型マイグレーションおよび我々が開発したポストコピー型マイグレーションを選択的に利用可能である．各ホスト OS 上で動作する管理デーモンに対して，XML-RPC 経由で仮想マシンの作成・破棄，マイグレーションを命令する．

仮想マシンを一台もホストしていない物理ノードは，サスペンド・レジューム（ACPI S3）機能を利用して，一時的にハードウェアを停止する．このときの消費電力は機種によって変わるが，我々のテストベッドでは数 W 程度になることを確認した．仮想マシンを復帰する際には，Wake-on-LAN よりも信頼性が高い Intel AMT (Active Management Technology) を用いている．数秒程度で，停止した物理ノードを復帰し再び仮想マシンをホストできる状態に戻すことができる．仮想マシン集約システムによる省電力化の効果は，本稿での取り扱い範囲を超えるため，今回は省電力機能を無効にしている．

5. 評　　価

本節では，ポストコピー型あるいはプレコピー型のマイグレーションを使用した仮想マシン集約システムが，急激な負荷上昇に対してどのように対処できるのか比較を行う．我々の仮想マシン集約システムの基本的な動作も確認する．

実験環境を**図 2** に示す．各仮想マシンに対しては 1 物理 CPU コアおよび 2GB のメモリを割り当てる．ただし，固定的に常に 1 物理 CPU コアを割り当てるのではなく，CPU 消費量が少ないときには複数の仮想マシンで一つの物理コアを共有する．実験では 2 台の仮想マシンを用いる． 仮想マシンをホストするサーバノード*1は 2 コアの CPU を有している．実験では，このうち 1 コアのみを仮想マシンに用いるようにホスト OS のスケジューラを明示的に設定することで，複数の仮想マシンが一つの物理コアを共有する状況を再現する．また，仮想マシンのストレージを提供するストレージサーバノードや，仮想マシン配置を管理するヘッドノードも用意した．

本稿では，簡単なアルゴリズムを用いて再配置を実行する．

- ある物理ノードの CPU 使用率が 90%以上であり，なおかつ，その物理ノード上に active な仮想マシン（CPU 使用率が 5%以上の仮想マシン）が複数台存在する状態になると，その物理ノードは過負荷状態になったと定義する．
- また，物理ノードの CPU 使用率が 30%未満の状態を，その物理ノードは idle 状態で

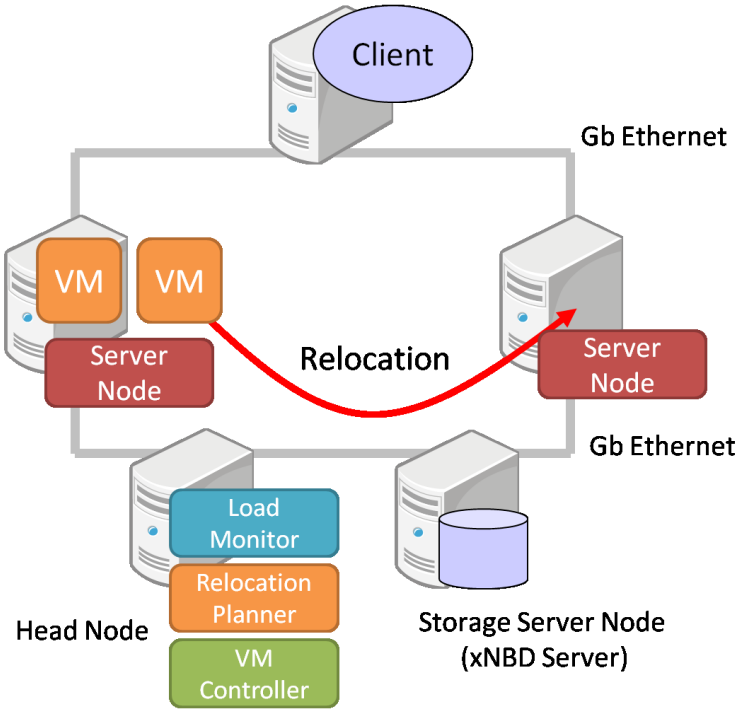

図 2　実験環境

⋆1 Intel Core2 Duo E6350，20GB RAM

あると定義する．

再配置計画エンジンは，物理ノードおよび仮想マシンの CPU 使用率を監視し，その値によって次の判断を行う．

- 物理ノードの過負荷状態を検知すると，その物理ノード上の active な仮想マシンに対して，1 台をその物理ノードに残し，残りを idle 状態の物理ノードに移動することを決定する．このとき CPU 使用率が最も高い仮想マシンをその物理ノードに残す．CPU 使用率が高い仮想マシンほどメモリページの更新頻度が高い，つまりマイグレーション時のオーバーヘッドが大きいとみなし，その再配置をできるだけ避けるようにしている．
- idle 状態かつ仮想マシンをホストしている物理ノードが複数存在することを検知すると，それら idle 物理ノード上の仮想マシンを 1 台の idle 物理ノード上に集約することを決定する．このとき CPU 使用率が最も高い idle 物理ノードを集約先とする．

CPU 使用率の一時的な変化に過度に反応しないように，再配置計画エンジンが CPU 使用率を用いる際には，過去 10 秒間の平均値を用いる．また，再配置計画エンジンは，一秒ごとに再配置の判断を行う．ただし，再配置を実行してから 10 秒間経過するまでは次の再配置を実行しない．同様に過度な反応を抑えるためであり，さらに，前回の再配置にともなって発生したマイグレーション自体の資源消費が再配置判断に影響しないようにするためである．

5.1 CPU 負荷プログラム

5.1.1 再配置動作の確認

最初に再配置動作を確認するために，簡単な負荷生成プログラムを仮想マシン内部で動かして，上記アルゴリズムどおりに再配置が実行されることを確認した．負荷生成プログラムは，短時間の計算とスリープを繰り返すことで，任意の CPU 使用率をおおよそ再現する[*1]．仮想マシン VM0 では常に CPU 使用率 20%の負荷を生成し，仮想マシン VM1 では CPU 使用率を 60 秒ごとに 0%，50%，100%，50%，0%と階段状に変更する．実験開始時には仮想マシンは両方とも物理ノード PM0 上で起動した．

ポストコピー型マイグレーションを用いた場合における，物理ノードおよび仮想マシンの CPU 使用率の時間経過を**図 3** から**図 6** に示す．グラフ中の縦線は実行ホストが切り替わった時刻を表す．

図 6 で明らかなように，時刻 37:20 付近から VM1 上の負荷生成プログラムが CPU 使用率 100%の負荷生成を試みている．しかし，VM0 および VM1 ともに物理ノード PM0 上に存在しているため，VM1 は 80%程度の CPU 時間しか得られていない．その後，10 秒程度経過して，再配置計画エンジンの過負荷状態の判定基準に達すると，再配置が実行されている．このとき，約 20%の CPU 時間を消費する VM0 の方が，約 80%の CPU 時間を消費する VM1 よりもマイグレーションのオーバーヘッドが小さいと見なして，VM0 を idle 状態の物理ノード PM1 に移動している．**図 5** では，VM0 が PM1 で CPU 時間を消費し始めたことが見て取れる．この再配置によって，VM0 および VM1 ともに，必要な CPU 時間を確保できた．その後，時刻 39:30 付近で VM1 の負荷が 0%に戻ると，約 10 秒後に再配置計画エンジンの判定基準に達して，再び 2 つの仮想マシンが 1 つの物理ノード上に集約される．判定時点で CPU 負荷平均が低かった VM0 をマイグレーション対象としている．VM1 の負荷が 0%に戻ってから，40 秒程度ですべてのメモリページの転送を完了し，PM1 への依存性を解消できた．

以上から，本再配置システムが正しく動作していることが確認できた．物理ノードおよび仮想マシンの CPU 使用率を監視しながら，一定のスレッショルドを超えたことを検知すると，再配置により仮想マシンの分散および集約を実行できる．

5.1.2 プレコピー型との比較

次に，この負荷生成プログラムを用いて，ポストコピー型およびプレコピー型の比較を行う．プレコピー型を用いた際の結果を**図 7** から**図 10** に示す．KVM 本来の設定値ではワークロードによってはマイグレーションが完了しないことがあるため，マイグレーションの際の許容ダウンタイムを 1 秒に設定している．図 10 に注目すると，時刻 16：00 付近で VM1 上の負荷生成プログラムが CPU 使用率 100%の負荷生成を試みていることがわかる．約 10 秒後に再配置計画エンジンの判断基準に達して，VM0 を物理ノード PM1 上に移動すべくマイグレーションが開始されている．しかし，プレコピー型ではすべてのメモリページを移動先に転送するまで実行ホストを切り替えることができないことから，実行ホストを切り替えるまでに 60 秒程度を費やしてしまった．時刻 17:00 付近まで VM0 は物理ノード PM0 上で動き続けたため，VM1 は 100%の CPU 時間を確保することができなかった．その後，時刻 18:00 付近で，VM1 の負荷が 0%に戻っているものの，PM1 への依存性が解消されるまで 60 秒程度を要している．

以上から，プレコピー型は実行ホストを切り替えるのに時間を要するため，過負荷状態を解消するまでに要する時間はポストコピー型よりも長いことが確認できた．この実験においては 60 秒程度の間にわたって過負荷状態を解消することができなかった．

⋆1 厳密には，他の仮想マシンの存在等により，若干変動する．

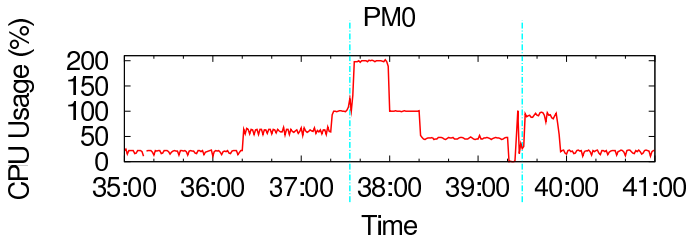

図 3 物理ノード PM0 の CPU 使用率（ポストコピー型）

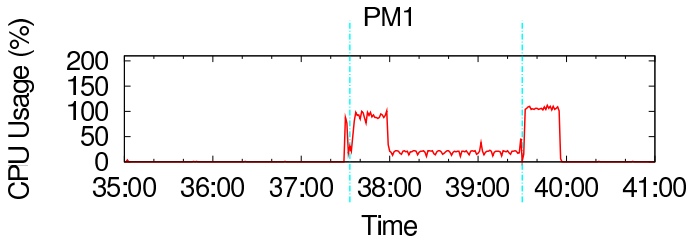

図 4 物理ノード PM1 の CPU 使用率（ポストコピー型）

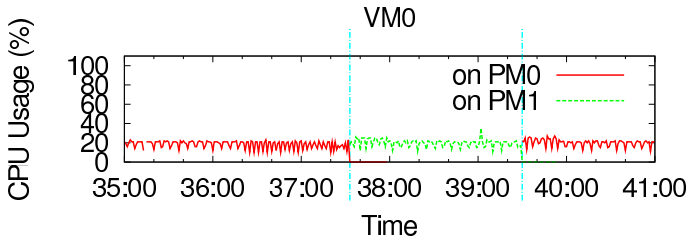

図 5 仮想マシン VM0 の CPU 使用率（ポストコピー型）

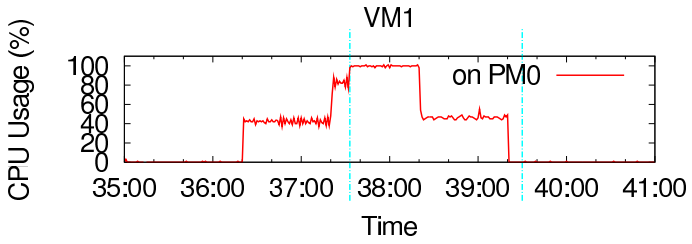

図 6 仮想マシン VM1 の CPU 使用率（ポストコピー型）

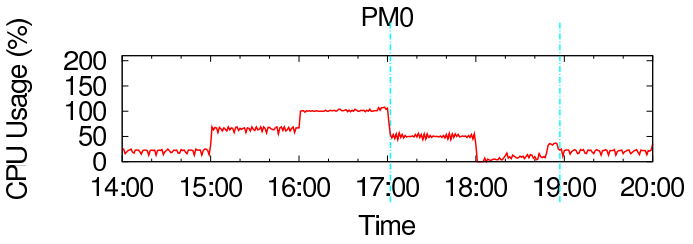

図 7 物理ノード PM0 の CPU 使用率（プレコピー型）

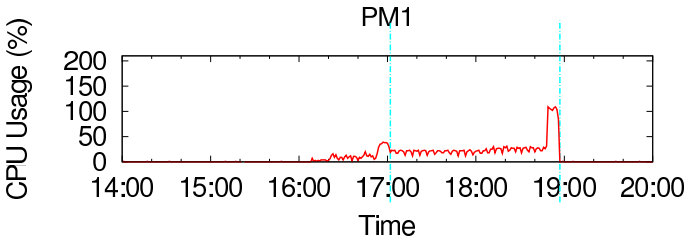

図 8 物理ノード PM1 の CPU 使用率（プレコピー型）

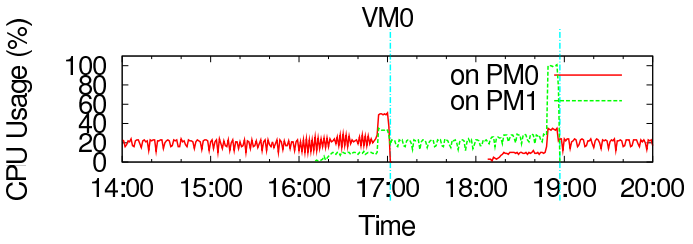

図 9 仮想マシン VM0 の CPU 使用率（プレコピー型）

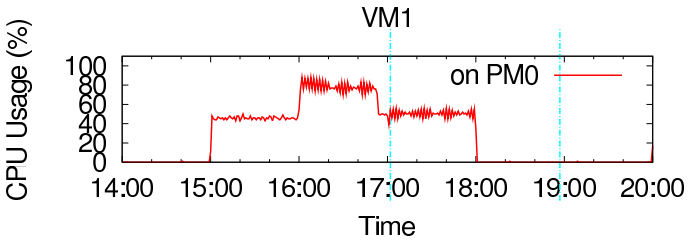

図 10 仮想マシン VM1 の CPU 使用率（プレコピー型）

5.1.3 補　　足

この実験結果について付加的な議論をする．負荷生成プログラムのメモリフットプリントは 40KB 程度と非常に小さく，ポストコピー型において実行ホスト切り替え後の性能低下が発生しにくい場合である．プログラムの使用メモリ領域がすべて転送されれば，未だ転送されていないページにアクセスして移動元ホストから取得することは発生しにくい．このようなワークロードに対しては、ポストコピー型のデメリットはほぼ存在しない。

ポストコピー型の方がマイグレーションにともなう CPU 資源消費が大きい．現在の実装で 1 ページごとにメモリ取得要求を移動元へ送信している影響だと思われる。今後の実装の最適化により改善できると考えている．また，プレコピー型では仮想マシンの QEMU プロセス自身がすべてのマイグレーション処理を担うのに対して，ポストコピー型では仮想マシンとは別のプロセスがメモリページの転送を担う．このプロセスは仮想マシンが動作する物理 CPU コアとは別のコアでも動作できたため，物理ノードの CPU 使用率が 200%近くになったときもあった．仮想マシンへのマイグレーションの影響を最小限にするという点で，別の物理コアでメモリページ転送用プロセスが動作することには利点があると考えている．一方で，ポストコピー型とプレコピー型の方式自体の利点を比較するために，今後の実験においては，ポストコピー型におけるメモリページ転送用プロセスも仮想マシンが動作する物理コアで動作するように，ホスト OS のプロセススケジューリングを調整することを検討している．

KVM のプレコピー型マイグレーションの実装では，メモリページの内容がすべて 0x00 であるページは転送しない．ポストコピー型でも同様に実装できる機能であるものの，未実装であることからプレコピー型の実験においては，両者の比較のためこの機能を無効にしている．また，実際に適切なメモリ量を搭載してワークロードを処理している仮想マシンにおいては，大半のメモリページに一度はデータが書き込まれているはずであり，仮にこの機能を有効にしたとしてもデータ転送量の削減効果は限定的である．この機能を有効にした場合の結果を参考のため**図 11** から**図 14** に載せる．仮想マシンが起動してから本実験を行うまでに書き込みを行ったメモリ領域は小さく，0x00 で埋まっているメモリページが大半であることから，転送時間は大幅に短くなっている．ポストコピー型においても同様の機能を実装すれば，この場合メモリページの転送期間を短縮できる．

5.2 カーネルコンパイル

アプリケーションの実効性能に対する影響を調べるため，簡単な実験を行った．Linux の

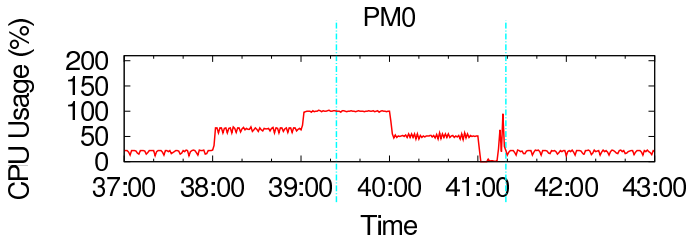

図 11 物理ノード PM0 の CPU 使用率（プレコピー型，転送ページ削減機能あり）

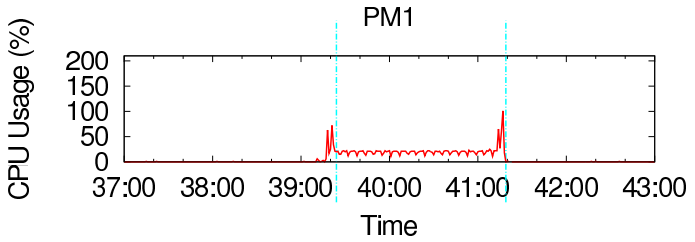

図 12 物理ノード PM1 の CPU 使用率（プレコピー型，転送ページ削減機能あり）

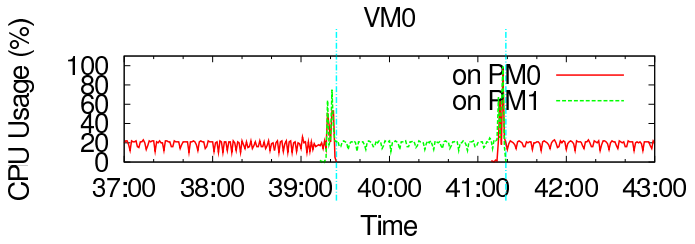

図 13 仮想マシン VM0 の CPU 使用率（プレコピー型，転送ページ削減機能あり）

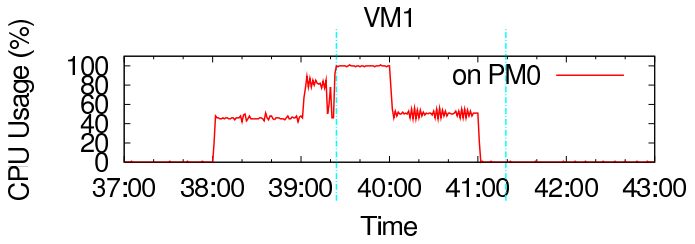

図 14 仮想マシン VM1 の CPU 使用率（プレコピー型，転送ページ削減機能あり）

表 1 カーネルコンパイル時間の比較（秒）

	VM0	VM1
単独配置	220.40	-
ポストコピー型	225.15	225.70
プレコピー型	257.21	259.45
再配置なし	345.26	345.90

カーネルコンパイル[*1]を仮想マシン上で行い，その完了時間を比較する．最初，2 つの仮想マシンは両方とも物理ノード PM0 上に存在している．はじめに仮想マシン VM0 上でコンパイルを開始し，90 秒経過してから仮想マシン VM1 でもコンパイルを開始した．

カーネルコンパイル時間を**表 1** に，ポストコピー型およびプレコピー型のマイグレーションを利用して再配置を行った場合における物理ノードおよび仮想マシンの CPU 使用率の変化を**図 15** から**図 22** までに示す．

再配置を行わなかった場合，カーネルコンパイルには 345 秒程度要してしまっている．物理ノード PM0 の CPU 使用率は常に 100%であり，双方の仮想マシンが同時にコンパイルを実行していた約 250 秒間には，各仮想マシンは 50%の CPU 時間しか得られなかった．一方，ポストコピー型マイグレーションを利用して再配置を行った場合，**図 18** 等が示すように，仮想マシン VM1 でコンパイルを開始した直後（時刻 09:30 付近）に，すぐさま再配置が実行されている．双方の仮想マシンが 50%の CPU 時間しか得られなかった時間は，再配置計画エンジンが過負荷状態を検知するまでの十秒弱にとどまっている．あらかじめ 2 つの物理ノードに各仮想マシンを単独で配置していた場合に対して，コンパイル時間は約 2%程度の増加したにすぎない．

プレコピー型を用いた場合，**図 21** や図 22 で明らかなように，60 秒程度にわたって，各仮想マシンが 50%程度の CPU 時間しか得られない期間が存在している．結果として，単独配置の場合と比較して，約 17%もコンパイル時間は増加してしまった．

さらに，2 つの物理ノードが必要であった時間（この場合物理ノード PM1 を使用していた時間）を**図 16** および**図 20** で比較すると，ポストコピー型においては約 260 秒であったのに対し，プレコピー型では約 320 秒であった．ポストコピー型の方が物理ノード PM1 の稼働時間を短縮できており，ハードウェアのスタンバイ機構（ACPI S3）と組み合わせれば，消費電力の削減がより多く可能になると考えられる．

⋆1 `make allnoconfig` した後，`time make -j 2`

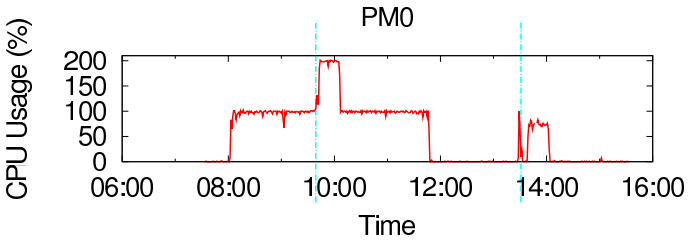

図 15 物理ノード PM0 の CPU 使用率（ポストコピー型，カーネルコンパイル）

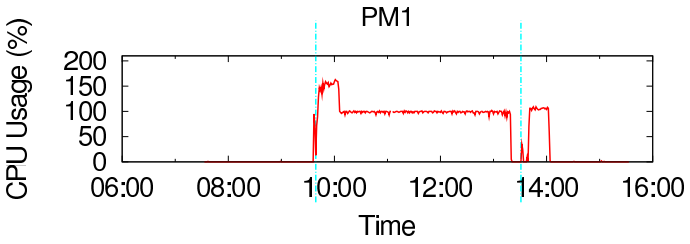

図 16 物理ノード PM1 の CPU 使用率（ポストコピー型，カーネルコンパイル）

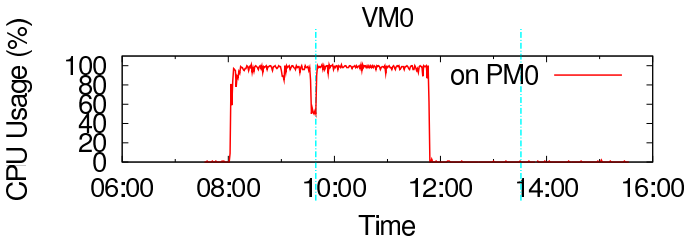

図 17 仮想マシン VM0 の CPU 使用率（ポストコピー型，カーネルコンパイル）

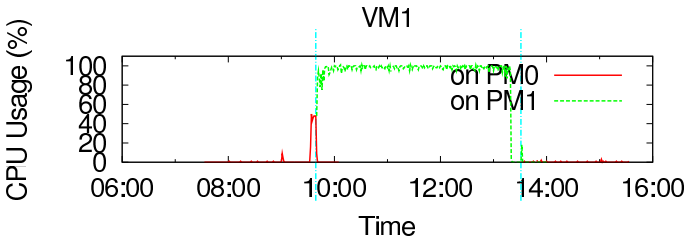

図 18 仮想マシン VM1 の CPU 使用率（ポストコピー型，カーネルコンパイル）

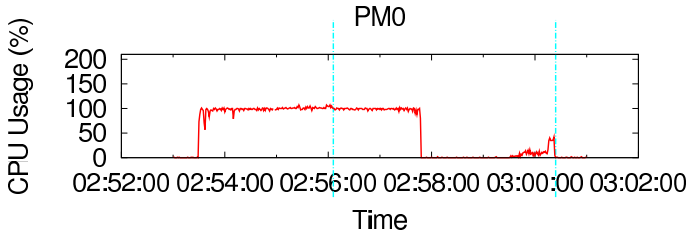

図 19 物理ノード PM0 の CPU 使用率（プレコピー型，カーネルコンパイル）

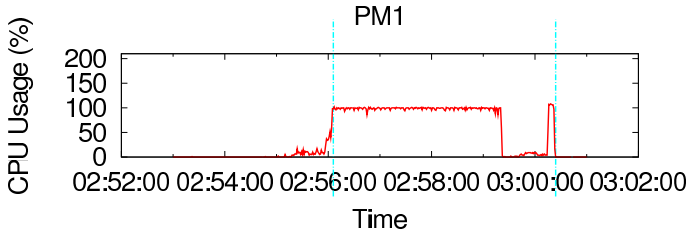

図 20 物理ノード PM1 の CPU 使用率（プレコピー型，カーネルコンパイル）

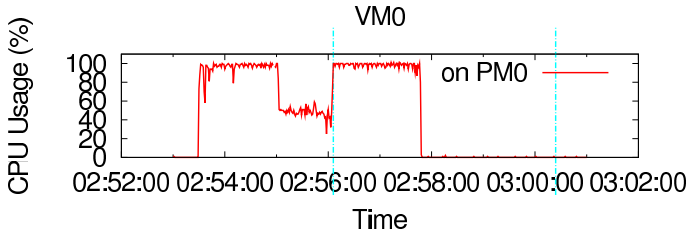

図 21 仮想マシン VM0 の CPU 使用率（プレコピー型，カーネルコンパイル）

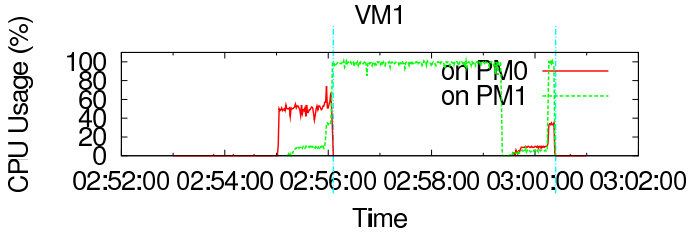

図 22 仮想マシン VM1 の CPU 使用率（プレコピー型，カーネルコンパイル）

5.3 SPECweb

ウェブサーバのベンチマークプログラムである SPECweb2005 Banking[12] を用いて，再配置動作にともなう性能低下を評価した．仮想マシンの内部にウェブサーバおよびサーバ側ベンチマークプログラムをインストールし，仮想マシン VM0 に対しては同時セッション数を 100 に設定して負荷をかけ続ける．一方，仮想マシン VM1 に対しては，当初の同時セッション数 20 を途中で 200 に変更した．変更する際には 10 秒ごとに同時セッション数を 30 ずつ増加させている．最初，2 つの仮想マシンをともに物理ノード PM0 で起動する．その後，同時セッション数を増加させていくと，再配置計画エンジンが過負荷状態を検知し，一方の仮想マシンを別の物理ノードへマイグレーションする．

ポストコピー型およびプレコピー型のマイグレーションを用いた場合において，物理ノードおよび仮想マシンの CPU 使用率の変化をそれぞれ**図 23** から**図 30** までに示す．また SPECweb が計測した応答品質の変化を**図 31** から**図 34** に示す．応答品質とは，SPECweb Banking がエミュレーションする銀行サイトについて，そのサービスを利用しているユーザがどの程度ストレスを感じているのかの指標である．応答時間に応じて，各セッションの状態が Good，Tolerable，Failed に分類される．

最初に，ポストコピー型の結果を確認する．図 23 および**図 26** が示すように，仮想マシン VM1 に対する同時セッション数を増加させていくにしたがって，VM1 および物理ノード PM0 の CPU 使用率も増加していく．VM1 の同時セッション数が 170 に設定された時刻 16:25 付近で再配置計画エンジンが過負荷状態を検知して，その時点で CPU 使用率が低かった仮想マシン VM0 を物理ノード PM1 に移動している．図 31 および**図 32** で明らかなように，実行ホスト切り替え直後には若干応答品質が悪化しているものの，すぐに回復している．なお，同時セッション数を徐々に増加させている期間において，CPU 使用率の山型の増加や応答品質の悪化が存在する．これはベンチマークにおいて，一度に 30 人ユーザが同時刻に銀行サイトにログインしようとした状況が再現されたためである．ポストコピー型およびプレコピー型双方の実験結果において発生している．

プレコピー型の結果においても，VM1 の同時セッション数が 170 に達した段階で，再配置計画エンジンが過負荷状態を検知して，VM0 のマイグレーションを開始している．しかし，**図 29** 等が示すように，実行ホストを切り替えることができたのは，約 60 秒が経過してからである．その間，物理ノード PM0 の CPU 使用率はほぼ 100%を推移し（**図 27**），十分な CPU 時間を得られなかった VM1 の応答品質は大幅に悪化していた（図 34）．

表 2 は，前述グラフ描画期間中における応答品質の総計を示している．過負荷状態に陥っ

表 2　応答品質総計の比較（240 秒間）

	Good	Tolerable	Failed
ポストコピー型 (VM0)	3622 (99.15%)	23 (0.63%)	8 (0.22%)
ポストコピー型 (VM1)	4119 (89.56%)	299 (6.50%)	181 (3.94%)
プレコピー型 (VM0)	3522 (98.11%)	19 (0.53%)	49 (1.36%)
プレコピー型 (VM1)	3130 (76.51%)	412 (10.07%)	549 (13.42%)

たことによる応答品質の悪化は，仮想マシン VM1 に対して，ポストコピー型の結果においては 10%程度と軽微である一方で，プレコピー型の結果においては 23%程度も存在した．過負荷状態においては，ホスト OS のプロセススケジューラが同等に CPU 時間を割り当てるため，負荷が高い仮想マシン VM1 の方で応答状態の悪化が顕著である．

以上の結果をまとめると，ポストコピー型の方が過負荷状態をすばやく解消できたため，応答品質の低下を抑えることができた．プレコピー型は過負荷状態が 60 秒間にわたって継続したため，その間応答品質は悪化してしまった．

6. 関 連 研 究

ポストコピー型マイグレーションを仮想マシン集約システムに適用した例は我々の知る限り存在していない．いずれもプレコピー型を利用している．

Sandpiper[6] では，資源消費量に応じた動的な仮想マシン配置を実装している．資源消費量の観察を仮想マシンの内部および外部で行う場合それぞれを比較している．前者の場合には，ゲスト OS 上でリクエストの到着頻度や応答時間を計測して，キューイング理論より SLA を満たすために必要な資源量を直接算出できる．後者の場合には，過去の資源消費量の推移から間接的に必要な資源量を見積もる．ワークロードの状態を直接観察できる前者の方が，過負荷状態となる前に再配置を実行できることを確認している．また，過去の資源消費量の履歴から自己回帰分析により直後の消費量を予測して，再配置状態の決定に用いている．

Entropy[7] では，制約問題を解くことで仮想マシンの配置状態を決定している．最初に，各仮想マシンの必要資源量を満たす最小の物理ノード数を求める．次に，その物理ノード数をもとに再配置のコストが最も小さくなる再配置計画を求める．仮想マシンの再配置がすべて完了するまでの時間，すなわち過負荷状態が継続する時間を，再配置のコストとして見積もっている．可能であれば複数のマイグレーションを同時に行って，再配置の完了時間を短くしている．仮想マシンの CPU 資源を考慮する際には，CPU 使用率を用いるのではなく，

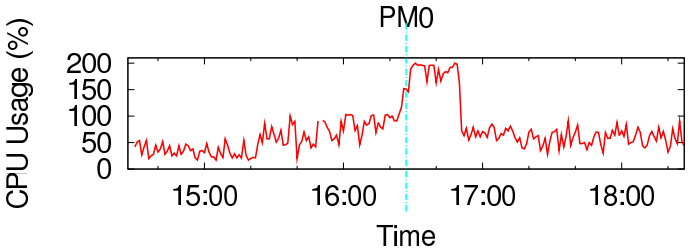

図 23　物理ノード PM0 の CPU 使用率（ポストコピー型，SPECweb）

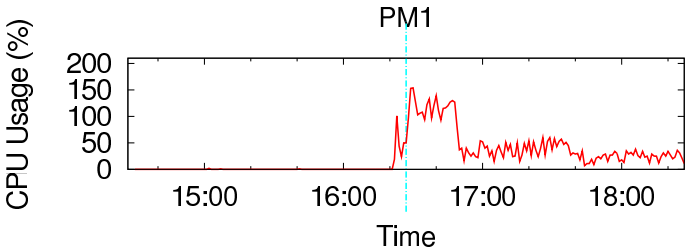

図 24　物理ノード PM1 の CPU 使用率（ポストコピー型，SPECweb）

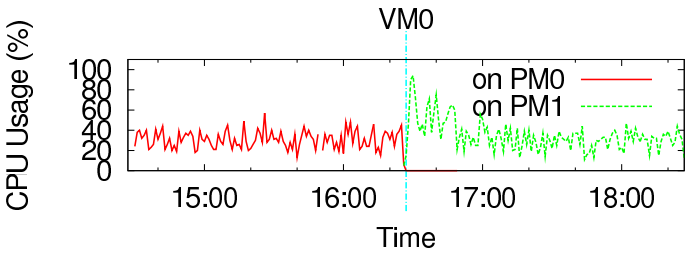

図 25　仮想マシン VM0 の CPU 使用率（ポストコピー型，SPECweb）

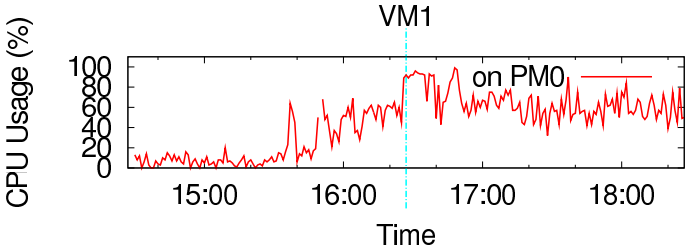

図 26　仮想マシン VM1 の CPU 使用率（ポストコピー型，SPECweb）

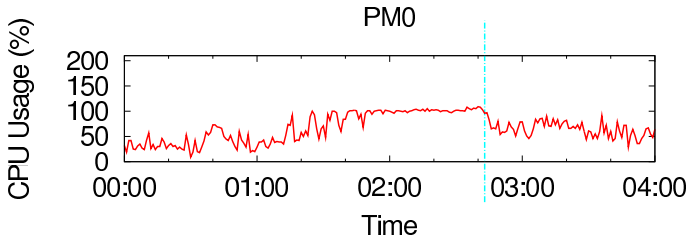

図 27 物理ノード PM0 の CPU 使用率（プレコピー型，SPECweb）

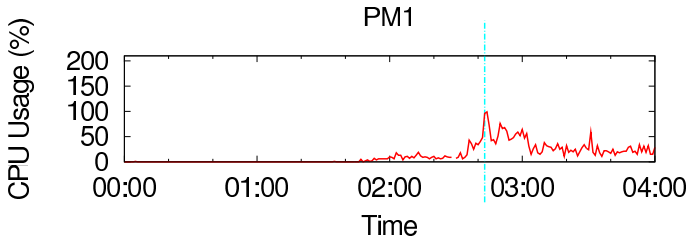

図 28 物理ノード PM1 の CPU 使用率（プレコピー型，SPECweb）

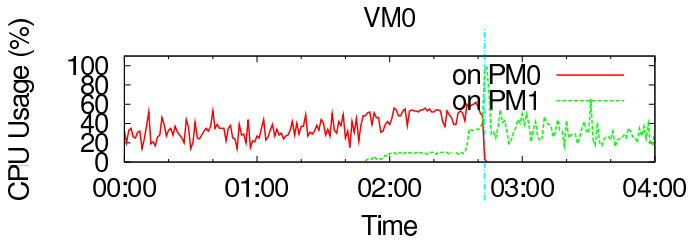

図 29 仮想マシン VM0 の CPU 使用率（プレコピー型，SPECweb）

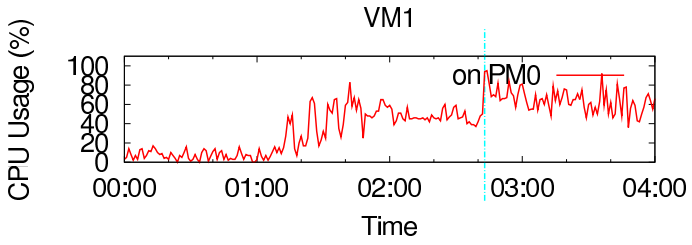

図 30 仮想マシン VM1 の CPU 使用率（プレコピー型，SPECweb）

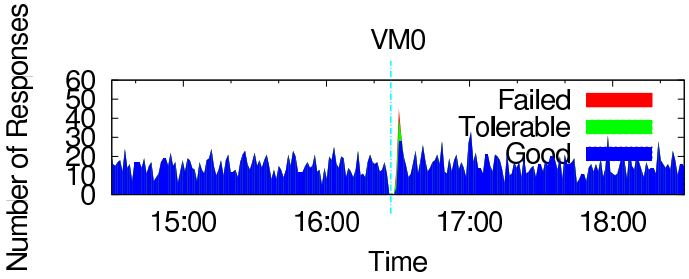

図 31 仮想マシン VM0 の SPECweb 応答品質（ポストコピー型）

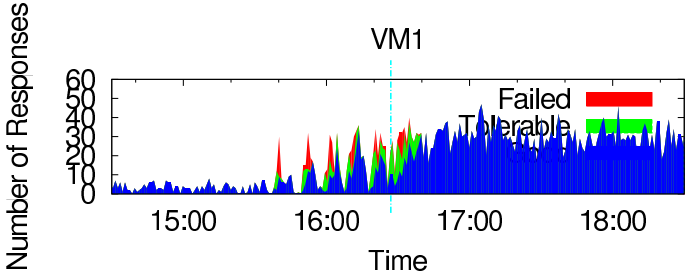

図 32 仮想マシン VM1 の SPECweb 応答品質（ポストコピー型）

仮想マシンが活動的（active）な状態であるか否かという 2 値のみを用いる．active 状態の仮想マシンに対して物理 CPU コアを割り付けるよう調整する．

その他，仮想マシンのマイグレーションのオーバーヘッドに着目した研究を挙げる．文献 13) では，マイグレーション時に発生するデータ転送量を考慮した再配置スケジューリングを検討している．マイグレーション完了時間のデッドラインを考慮しつつ使用帯域を最小にすることを試みている．文献 14) では，エンタープライズ分野のアプリケーションサーバの負荷の履歴を用いて，異なる仮想マシン配置ポリシーをシミュレーションにより評価している．4 時間ごとに過去の負荷変動履歴に基づいて再配置を実行するモジュールと，5 分ごとに過負荷状態を検知して再配置を実行するモジュールの 2 つを組み合わせている．マイグレーションが移動元・移動先ホストにともなう CPU 消費量は，マイグレーション時間に比例するとして算定している．文献 15) では，性能低下をともなうマイグレーションの実行回数を減らすためには，仮想マシン群全体の資源消費傾向の大きな変化が発生した際に再配置を検討すべきであるとしている．そして線形代数的手法によってその変化を把握する方法を

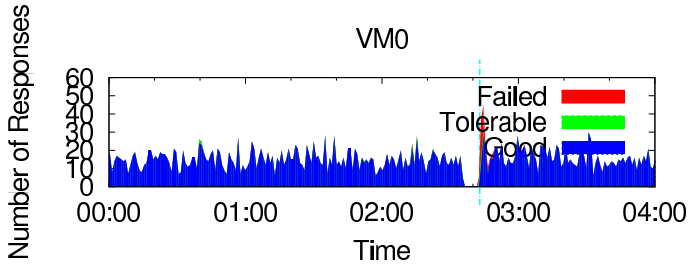

図 33 仮想マシン VM0 の SPECweb 応答品質（プレコピー型）

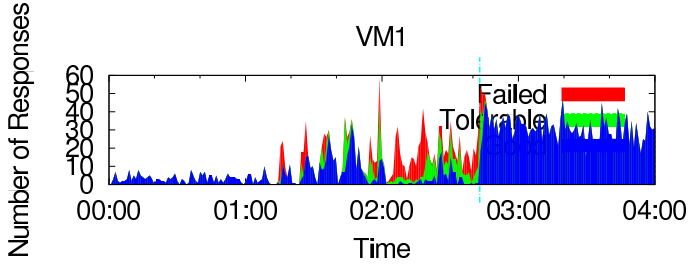

図 34 仮想マシン VM1 の SPECweb 応答品質（プレコピー型）

議論している．

これら関連研究で議論された手法は，仮想マシン集約システムに対してポストコピー型マイグレーションを使用するという我々の提案手法とは，互いに補完的な関係にあると考える．既存の仮想マシン集約システムにおいて，プレコピー型マイグレーションが用いられている部分をポストコピー型に置き換えるだけでも，性能低下が改善されるはずである．この点は今後検証に値すると考えている．

7. ま と め

本稿では，仮想マシン集約システムに対して、ポストコピー型マイグレーションを用いることを提案した．ポストコピー型マイグレーションは，実行ホストを 1 秒程度で切り替えることができるため，仮想マシンの配置を迅速に変更できる．ポストコピー型マイグレーションを仮想マシン集約システムに応用すれば，物理ノードの過負荷状態をすばやく解消して仮想マシンの性能低下を抑制できる．評価実験において，既存のプレコピー型マイグレーションを応用した場合に対する優位性を検証した．プレコピー型よりもポストコピー型を用いた場合の方が，過負荷状態を早く解消できたためワークロードの性能低下が軽微であった．また我々の仮想マシン集約システムが仮想マシンの負荷変動に追随して動的に配置を最適化できることも確認できた．

今後の課題を以下にまとめる．まず，我々のポストコピー型マイグレーション実装において，メモリページ転送時の CPU 資源消費量が大きい点を改善する．ディスク I/O やネットワーク I/O 等，CPU 使用率以外の資源消費量を監視対象として，再配置決定アルゴリズムに加味できるようにする．より多数の仮想マシンを対象にシステムを構築するとともに，さまざま再配置決定アルゴリズムを実装できる基盤を整える．そしてポストコピー型マイグレーションが，仮想マシンの性能保証と稼働物理ノード数の削減要求を，高いレベルで満たせることを検証していく．

謝辞 本研究は科研費（20700038）および 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）の助成を受けたものである．

参 考 文 献

1) Christopher Clark, Keir Fraser, Steven Hand, JacobGorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live migration of virtual machines. In *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation*, pp. 273–286. USENIX Association, 2005.
2) Michael Nelson, Beng-Hong Lim, and Greg Hutchins. Fast transparent migration for virtual machines. In *Proceedings of the USENIX Annual Technical Conference*, pp. 25–25. USENIX Association, 2005.
3) Andrey Mirkin, Alexey Kuznetsov, and Kir Kolyshkin. Containers checkpointing and live migration. In *Proceedings of the Linux Symposium*, pp. 85–92. The Linux Symposium, Jul 2008.
4) 広渕崇宏, 中田秀基, 伊藤智, 関口智嗣. 既存 VMM への適用が容易でゲスト透過なポストコピー型仮想マシン再配置機構. 情報処理学会論文誌:コンピューティングシステム（採録決定済み）, Vol. ACS31, , Aug 2010.
5) 広渕崇宏, 中田秀基, 伊藤智, 関口智嗣. 仮想計算機メモリの遅延再配置による高速ライブマイグレーション. 情報処理学会研究報告 (2009-OS-112). 情報処理学会, Jul 2009.
6) Timothy Wood, PrashantJ. Shenoy, Arun Venkataramani, and MazinS. Yousif. Black-box and gray-box strategies for virtual machine migration. In *Proceedings of the 4th Symposium on Networked Systems Design and Implementation*, pp. 229–242. USENIX Association, 2007.
7) Fabien Hermenier, Xavier Lorca, Jean-Marc Menaud, Gilles Muller, and JuliaL. Lawall. Entropy: a consolidation manager for clusters. In *Proceedings of the 5th In-*

ternational Conference on Virtual Execution Environments, pp. 41–50. ACM Press, 2009.

8) 広渕崇宏, 中田秀基, 小川宏高, 伊藤智, 関口智嗣. 仮想マシン技術とサーバ一時停止技術を利用した省エネデータセンタシステムの開発. 先進的計算基盤システムシンポジウム SACSIS 2010, pp. 99–100, May 2010.

9) Hidemoto Nakada, Takahiro Hirofuchi, Hirotaka Ogawa, and Satoshi Itoh. Toward virtual machine packing optimization based on genetic algorithm. In *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living (Proceedings of International Symposium on Distributed Computing and Artificial Intelligence 2009)*, Vol. 5518 of *Lecture Notes in Computer Science*, pp. 651–654. Springer, Jun 2009.

10) 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コンピュータシステム（SWoPP2010 発表予定）. 電子情報通信学会, Aug 2010.

11) Avi Kivity, Yaniv Kamay, Dor Laor, and Anthony Liguori. kvm: the Linux virtual machine monitor. In *Proceedings of the Linux Symposium*, pp. 225–230. The Linux Symposium, 2007.

12) Standard Performance Evaluation Corporation. SPECweb2005. `http://www.spec.org/web2005/`.

13) Alexander Stage and Thomas Setzer. Network-aware migration control and scheduling of differentiated virtual machine workloads. In *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, pp. 9–14. IEEE Computer Society, 2009.

14) Daniel Gmach, Jerry Rolia, Ludmila Cherkasova, and Alfons Kemper. Resource pool management: Reactive versus proactive or let's be friends. *Computer Networks*, Vol.53, No.17, pp. 2905–2922, 2009.

15) Thomas Setzer and Alexander Stage. Decision support for virtual machine reassignments in enterprise data centers. In *Proceedings of the 5th IEEE/IFIP International Workshop on Business-driven IT Management*, Apr 2010.