Vol.2013-DPS-154 No.17
Vol.2013-CSEC-60 No.17
2013/3/14

仮想化環境における キャッシュヒット率を考慮した VM メモリ割り当て

日名川 幸矢 [†1]　竹内 洸祐[†2]　山口 実靖[†1]

―近年，サーバの消費電力増加，設置スペース肥大化が問題となっており，その解決策の一つとして，仮想化技術を用いて複数のサーバを一台の物理計算機に集約する手法がある．仮想化環境では，仮想計算機を停止させることなくメモリの割当量を変更することが可能である．一つの物理計算機に複数の仮想計算機を稼働させ，それぞれの仮想計算機の負荷が時間により変化する場合は，静的なメモリ割り当てを行うとメモリを効果的に活用できないことになる．負荷変動に適切に対応するためには，動的に仮想計算機のメモリ割当量を変更する必要がある．本稿では，仮想計算機のキャッシュヒット率を考慮した動的メモリ割当量の制御手法を提案する．そして，性能評価を行い提案手法の有効性を示す．

Dynamic VM Memory Allocation using Cache Hit Ratio

KOUYA HINAGAWA[†1]　KOUSUKE TAKEUCHI[†2]
SANEYASU YAMAGUCHI[†1]

In virtualized environment, allocation memory size of virtual machines can be dynamically changed. For making use of physical memory, dynamic VM memory size adjustment, which is called *ballooning*, for dynamically changing loads is required. In this paper, we propose a new ballooning system which considers disk cache and cache hit ratio in guest operating system running on virtual machines. In addition, we introduce our implementation using Xen virtual machine monitor and our experimental results. The results have demonstrated that our proposed method have been able to significantly improve I/O performance.

1. はじめに

近年，情報技術が普及し，データセンター等において多数のサーバ計算機が稼動するようになった．これに伴い，サーバの消費電力の増加，設置スペースの肥大化が問題となっている．この問題に対する解決策の一つとして，仮想化技術を用いて複数のサーバ OS を一台の物理計算機に集約する手法がある[1].

仮想化環境では，仮想計算機（VM）のメモリ割当量の動的変更が可能であり，VM を停止させることなくメモリ割当量を変更することが可能である．一つの物理計算機に複数の VM を稼働させ，それぞれの VM の負荷が時間により変化する場合は，静的なメモリ割り当てを行うとメモリを効果的に活用できないことになる．負荷変動に対応するためには，動的に VM メモリ割当量を変更する必要がある．

仮想化ソフトウェアの Xen には動的に VM メモリ割当量を変更させる機能として，xenballooning がある．しかし xenballooning のメモリ割当量の算出方法は，プロセスの推定メモリ使用量のみを考慮し，ページキャッシュとして利用されるメモリを考慮していない．よって I/O 性能の最適化には適さないと考えられる．

本稿では，Xen によって提供される VM で，メモリ割当量，キャッシュヒット率，I/O 性能の関係性について調査を行い，キャッシュヒット率を考慮した VM メモリ割り当て量の変更手法を提案し，性能評価により有効性を示す．

†1 工学院大学工学部情報通信工学科
Department of information and Communications Engineering, Kogakuin University
†2 工学院大学大学院工学研究科電気電子工学専攻
Electrical Engineering and Electronics, Kogakuin University Graduate School.

2. Xenballooning

xenballooning は，Xen が持つ機能の一つで，VM に割り当てるメモリ割当量の動的な変更を行う機能である．[2] xenballooning はプロセスが使用しているメモリ量 (Commited_AS) のみを考慮して割当メモリ量の調整を行うため，ページキャッシュなどに使用されているメモリ量は考慮されない．よって xenballooning は I/O 性能の向上に対して効果がないと予想される．

また，異なるゲスト OS 間でのメモリ量調整の機能が無く，ホスト OS へ物理メモリ以上のメモリ量を要求してしまうことがある．この場合でも仮想計算機は動作を継続することが可能であるが，先にホスト OS にメモリを要求したゲスト OS が優先的にメモリを得てしまう．そのため，特定のゲスト OS がメモリを占有してしまい，他のゲスト OS のメモリ割当量が増えないという事象が発生する．

3. キャッシュヒット率の解析

Xen 仮想化環境におけるゲスト OS とホスト OS のキャッシュヒット率を測定するために，図 1 のようにホスト OS，ゲスト OS のカーネル内でページキャッシュ前後の I/O 量を監視した．

具体的には，キャッシュ前の I/O 量としてページキャッシュへのアクセス量を測定するために Linux カーネルの改変を行った．ゲスト OS ではカーネル構成ファイルの `mm/filemap.c` 内にある「`find_get_page`」関数の実行

回数とブロックサイズの監視，ホスト OS では drivers/xen/blkback/blkback.c 内にある「dispatch_rw_block_io」関数の実行回数とブロックサイズの監視をするための改変を行った．

キャッシュ後の I/O 量として VM の仮想ブロックデバイス(XVD)における I/O 量，物理 HDD における I/O 量を測定するためにカーネルの改変を行った．ゲスト OS では，「do_blkif_request」関数の実行回数とブロックサイズの監視，ホスト OS では，drivers/scsi/sd.c 内にある「sd_prep_fn」関数の実行回数とブロックサイズの監視をするための改変を行った．

測定したキャッシュ前後の I/O 量の比よりキャッシュヒット率を求めた．具体的には，（キャッシュ後の I/O 量）／（キャッシュ前の I/O 量）をキャッシュミス率とし，1－（キャッシュミス率）をキャッシュヒット率とした．

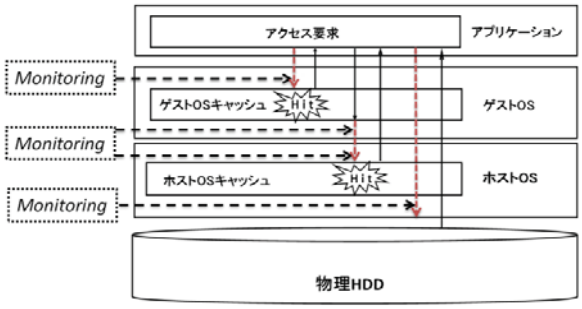

図 1　キャッシュヒット率の測定

4. 基本性能調査

本章にて，VM の基本性能調査を行う．キャッシュヒット率と I/O 性能の関係，ホスト OS とゲスト OS のメモリ割当量と I/O 性能やキャッシュヒット率の関係について調査を行うため，VM 上でファイルアクセスベンチマークを実行し，スループットとキャッシュヒット率を測定した．

4.1 キャッシュヒット率と I/O 性能

VMにおけるキャッシュヒット率とI/O性能の関係を調査するために，VM上でファイルアクセスベンチマークを実行し，スループットとキャッシュヒット率を測定した．ベンチマークでは，VM内ファイルシステム上に1MBのファイルを5000個(5000MB)作成し，これらのファイルに対して各種読み込み方法にてアクセスを行った．読み込みサイズは1MB（ファイル全体）であり，ランダムアクセス時におけるファイル選択の乱数として，一様分布乱数と平均2500の指数分布乱数を用いた．VM のメモリ割当量を1000MB~7000MBの範囲で変更し，キャッシュヒット率の変化と，それに伴うI/O性能の変化について調査した．物理計算機，仮想計算機の仕様は表1, 2, 3の通りである．また，キャッシュヒット率はVM上で測定し，3章の方法を用いて算出した．

表 1　物理計算機の仕様

CPU	AMD Athlon 1640B 2.8[GHz]
CPU Core	4VCPU
Memory	8[GB]
HDD	500[GB]，7200[rpm]

表 2　仮想計算機（ホスト OS）の仕様

Host OS	CentOS 6.3 x86_64
Host Kernel	Linux 2.6.32
Xen Version	4.1.2
Virtual CPU Core	1
Virtual Memory	1024[MB]

表 3　仮想計算機（ゲスト OS）の仕様

Guest OS	CentOS 5.4 x86_64
Guest Kernel	Linux 2.6.18
Virtual CPU Core	1
Virtual Memory	測定によって変動

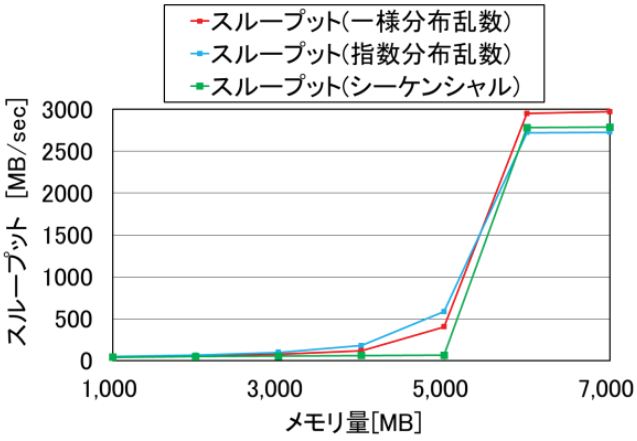

図 2　スループットの測定

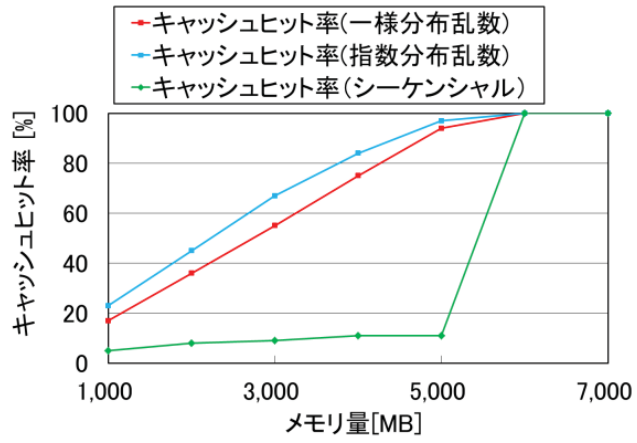

図 3　キャッシュヒット率の測定

Vol.2013-DPS-154 No.17
Vol.2013-CSEC-60 No.17
2013/3/14

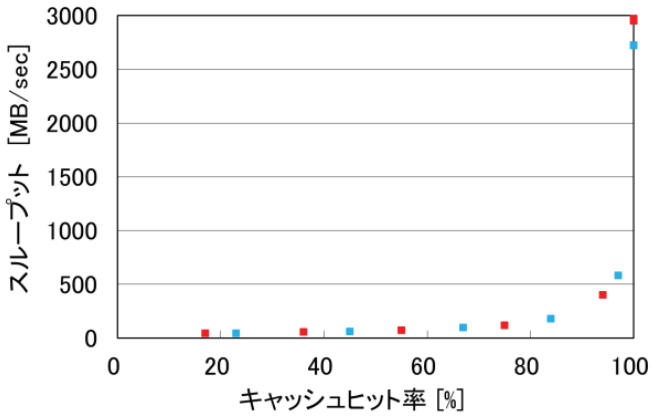

図 4　キャッシュヒット率とスループットの関係

測定結果を図 2，3 に示す．図 2，3 より，繰り返しシーケンシャルリードの場合ベンチマークに使用した合計データサイズより少ないメモリ割当量だとキャッシュヒット率が 0%に近いことが分かる．ランダムリードの場合はベンチマークに使用した合計データサイズより少ないメモリ割当量でも，キャッシュヒットしていることが確認できる．

一様分布乱数にてランダムリードを行ったときのキャッシュヒット率とスループットの関係を図 4 に示す．図 4 より，キャッシュヒット率が低い環境では，メモリ割当量を増やしても性能向上が小さく，メモリ割当の効果が小さいことが分かる．一方キャッシュヒット率が高い環境では，メモリ割当による性能向上が大きく，メモリ割当の効果が大きいことが分かる．

4.2 ホスト OS とゲスト OS のメモリ割当量

ゲスト OS でのキャッシュヒットしたときの I/O 性能とホスト OS でのキャッシュヒットしたときの I/O 性能を比較するために，VM 上でランダムアクセスベンチマークを実行し，性能を測定した．ベンチマークでは，VM 内ファイルシステム上に 1MB のファイルを 10,000 個(10,000MB)作成し，これらのファイルに対してランダムに読み込みアクセスを行った．ベンチマークに使用するファイルの数を 100 個～10000 個の間で変更した．読み込みサイズは 1MB(ファイル全体)であり，ファイル選択の乱数として，一様分布乱数を用いた．測定は，ゲスト OS に多くのメモリを割り当てた環境(ゲスト 7,000MB，ホスト 1,000MB)と，ホスト OS に多くのメモリを割り当てた環境(ゲスト 1,000MB，ホスト 7,000MB)で行い，キャッシュヒット率と I/O 性能を比較した．測定環境は前節の環境と同じである．

測定結果を図 5，6 に示す．図 5，6 を比較すると，ゲスト OS に多くメモリを割り当てた方がホスト OS に多くメモリを割り当てるよりスループットが良いことが分かる．このことから，アクセス要求がホスト OS のキャッシュヒットとなっても，その処理（ゲスト OS からホスト OS へのアクセス要求の転送）により I/O 性能は大きく低下してしまうことが分かる．

本実験により，VM における I/O 性能の向上を行うにはゲスト OS に最適なメモリ量を割り当てることが重要であることが分かる．

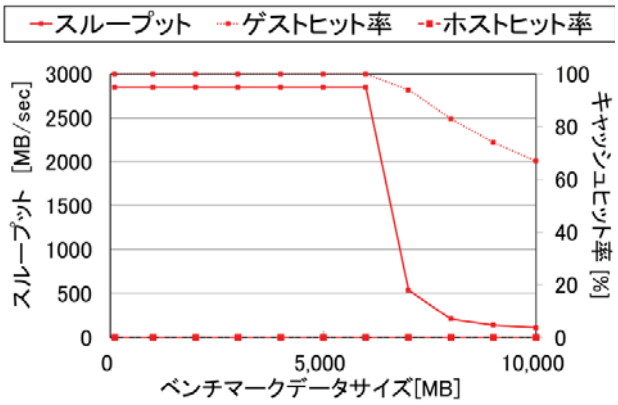

図 5　ゲスト OS メモリ 7000MB, ホスト OS メモリ 1000MB におけるキャッシュヒット率と I/O 性能

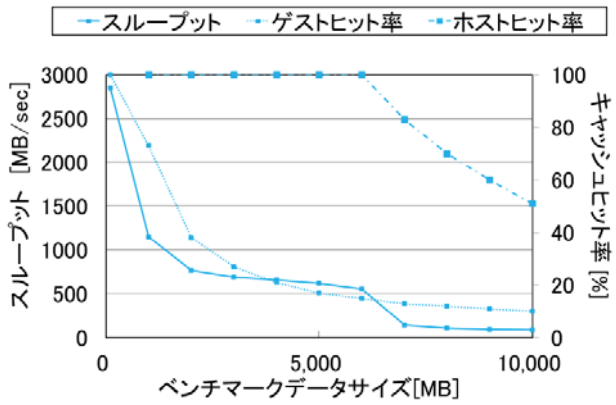

図 6　ゲスト OS メモリ 1000MB, ホスト OS メモリ 7000MB におけるキャッシュヒット率と I/O 性能

5. 提案手法

本章で，キャッシュヒット率を監視し，それにより VM に割り当てるメモリ量の変更を行う手法を提案する．

本手法では，各 VM のキャッシュヒット率の値をホスト OS にて集計し，以下のルールで VM メモリ割当量を決める．

1. キャッシュヒット率が γ %以上の VM のメモリ割当量を α %減少させ，これを再配分用メモリとする．
2. 全 VM のメモリ割当量を β %減少させ，これも再配分用メモリとする．
3. 上記 1,2 にて得た再配分用メモリを下記の A または B に従い，各 VM に再配分する．

Vol.2013-DPS-154 No.17
Vol.2013-CSEC-60 No.17
2013/3/14

A. キャッシュヒット率がγ%以上の VM を除く全ての VM に，VM のキャッシュヒット率により比例配分する．これにより，キャッシュヒット率が高い VM ほど多くのメモリが得られることになる．

B. キャッシュヒット率がγ%以上の VM を除く全ての VM に，VM のキャッシュミスヒット率により比例配分する．これにより，キャッシュヒット率が低い VM ほど多くのメモリが得られることになる．

メモリ割当量変更処理のオーバーヘッドを削減するため，上記 1，2 の段階ではメモリ割当量の変更は実行せず，上記 3 の終了後に変更を行う．また，キャッシュヒット率は 3 章の手法で算出する．

キャッシュヒット率が 100%未満になると，スループットが急激に悪くなることが 4 章の実験から分かっている．そのためメモリ割当に重み付けする必要があり，そのための手法として，手法 A と手法 B を用意した．ただし，手法 B は比較のために用意した手法であり，前章の考察に従わない合理的でない手法となっている．

また，閾値の近傍にて頻繁なメモリの増減が行われて大きなオーバーヘッドによる性能劣化が発生することを避けるために，本手法ではキャッシュヒット率が閾値を超える場合でも，すぐにメモリ割当量の変更は行わず，3 回連続で閾値を超えた場合にメモリ割当の変更を行うこととした．これにより 100%近いキャッシュヒット率を維持する時間を長くすることが可能となる．

キャッシュヒット率が 100%近い VM のメモリ割当量を増やしても I/O 性能は向上しないことが 4 章の実験の結果から分かっている．先ほどの重み付けを利用し，キャッシュヒット率が 100%近い状態が続くほど，メモリ量の減少率を上げるため，（閾値を超えた回数×0.1＋1）を減少率とした．これにより，メモリを必要としていない VM のメモリ減少速度を速くすることが可能となる．

提案手法は，xenballooning の実装に変更を加えることにより実現した．具体的には，ゲスト OS とホスト OS に 3 章で述べたキャッシュヒット率解析機能を追加し，ゲスト OS 上に定期的にキャッシュヒット率を調査してホスト OS に報告するサービスを実装し，ホスト OS に各ゲスト OS から報告されたキャッシュヒット率をもとに各 VM の割当メモリ量を定めるサービスを実装した．

6. 性能評価

性能評価は，単一物理計算機上に複数の VM を稼働させ，各 VM 上でランダムアクセスベンチマークを実行することにより行った．性能評価では，既存手法と提案手法の性能比較，提案手法の分配量変更による性能比較，書き込みが発生するベンチマークを用いた場合の既存手法と提案手法の比較を行った．

6.1 実験環境

実験は，Xen を用いて 1 台の物理計算機上に 3 台の VM を起動させ，全 VM 上で同時にベンチマークを実行し I/O 性能を測定した．ベンチマークでは，4 章の実験と同様で VM 内ファイルシステム上に 1MB のファイルを 10,000 個(10,000MB)作成し，これらのファイルに対してランダムに読み込みアクセスを行った．読み込みサイズは 1MB(ファイル全体)であり，ファイル選択の乱数として，一様分布乱数を用いた．

ベンチマークファイル数は，0 個(0MB)，100 個(100MB)，1,000 個(1,000MB)，2,000 個(2,000MB)，3,000 個(3,000MB)，4,000 個(4,000MB)，5,000 個(5,000MB)，7,500 個(7,500MB)，10,000 個(10,000MB)とし，600 秒毎にファイル数を変更させ，各 VM の負荷に時間変化を与えた．実験はファイル数の順番をランダムに変更し 10 回行い，スループットの相加平均と相乗平均により評価した．また，30 秒毎に各 VM のメモリ量を取得し，VM に与えられた I/O 負荷量とメモリ割当量の時間変化を調査した．実験に用いた計算機の仕様は 4 章の実験と同様で，表 1，2，3 の通りである．

6.2 節，6.4 節の実験に用いた提案手法のパラメータは A，B の両手法とも α を 10%，β を 10%，γ を 99%とした．

6.2 I/O 性能測定(既存手法と提案手法の比較)

既存手法と提案手法の比較を行うため，既存手法(改善前の xenballooning)，各 VM のメモリ割当量が均一になるように静的に割り当てた(2357MB)，提案手法 A，B の性能を比較した．測定結果を図 7，8 に示す．

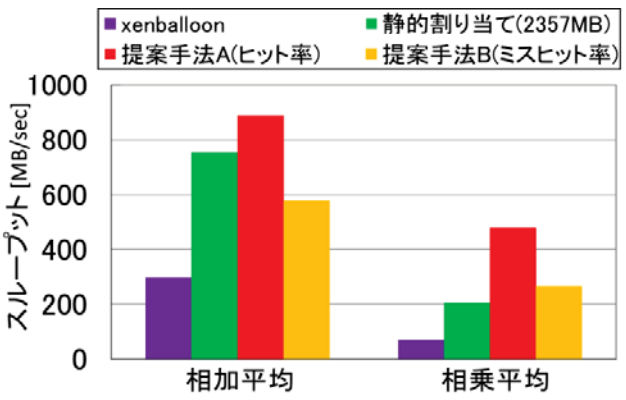

図 7　既存手法と提案手法の比較

Vol.2013-DPS-154 No.17
Vol.2013-CSEC-60 No.17
2013/3/14

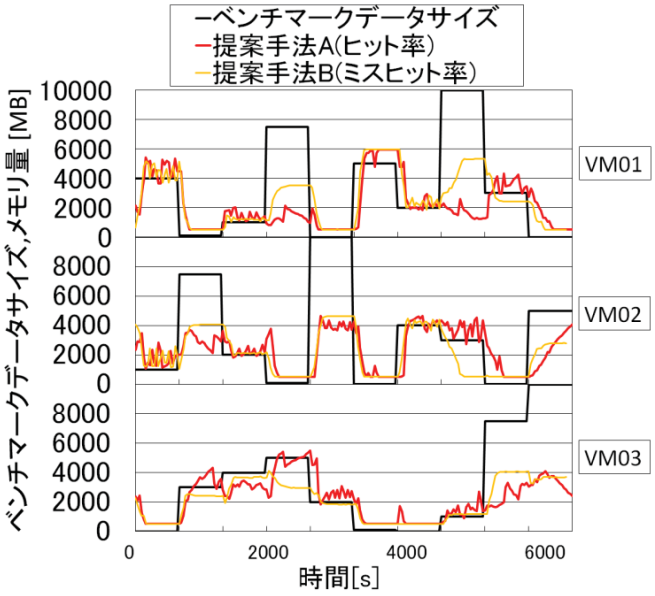

図 8　VM メモリ量変化の推移

図 7 がスループットの相加平均，相乗平均であり，図 8 が VM に与えられた I/O 負荷量とメモリ割当量の時間変化である．図 6 より，提案手法の方が既存手法より性能が高く，提案手法が有効であることが分かる．また提案手法 A のスループットが B より良かった理由はキャッシュヒット率が高い VM に優先的にメモリを割り当てることで，4 章で示した効果の大きい割当が多く行えたからだと考えられる．相乗平均を比較すると，相加平均に比べ性能差が大きい結果となった．これは，静的割り当ての場合にベンチマークデータサイズがメモリ割当量を上回るとスループットが大幅に低下するのに対し，提案手法はベンチマークデータサイズに合わせてメモリ割当量が変化するため，10 回のスループットが平均的に既存手法より高く，大幅なスループット低下が起こらなかったためだと考えられる．

図 7 より，ベンチマークデータサイズの変化によりメモリ割当量が変化していることが分かる．提案手法 A ではベンチマークデータサイズが大きい場合にメモリ割当量が下がるのに対し，提案手法 B ではベンチマークデータサイズが大きい場合にメモリ割当量が上がることが分かる．キャッシュヒット率が低い VM にメモリを割り当てても大きな性能向上が望めないため，提案手法 A のメモリ割当量が好ましい．

6.3　I/O 性能測定(提案手法の分配量変更による比較)

提案手法のパラメータ α と β の値を変更し，一様分布乱数のランダムリード性能を測定し，分配メモリの値変更による比較をした．実験結果を図 9，10，11，12，13，14 に示す．

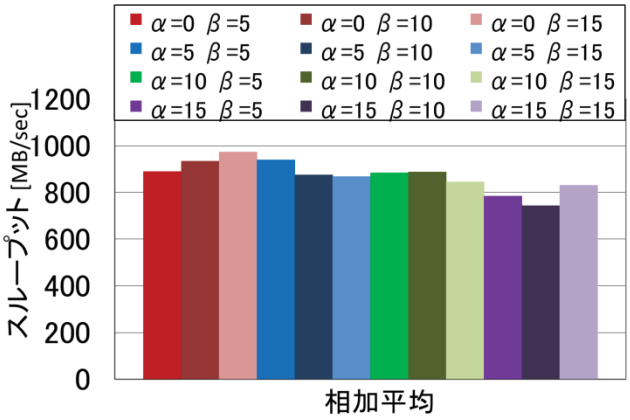

図 9　提案手法の分配量変更による比較（相加平均）

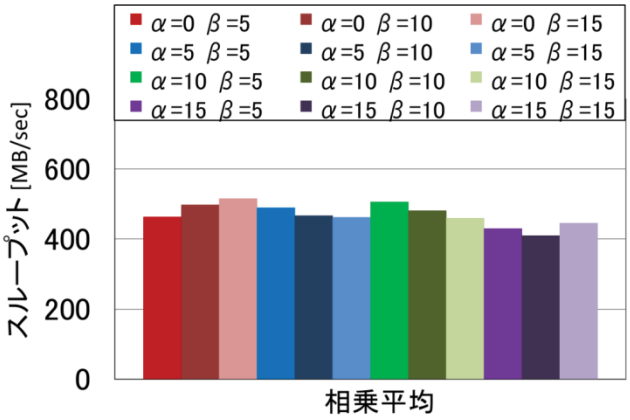

図 10　提案手法の分配量変更による比較（相乗平均）

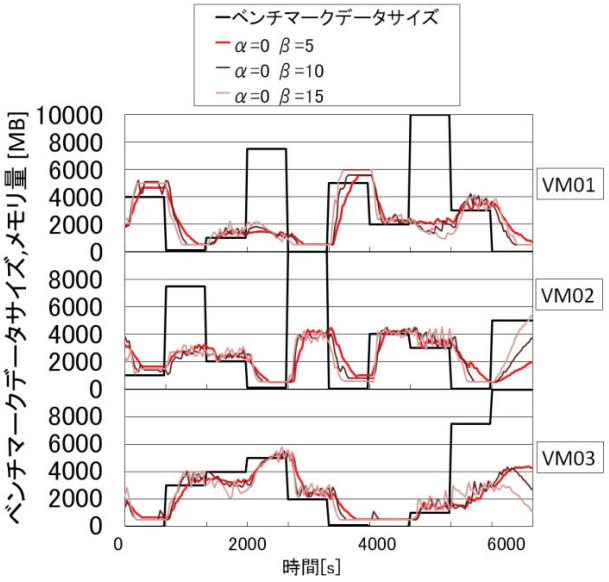

図 11　VM メモリ量変化の推移（分配量変更 α＝0）

Vol.2013-DPS-154 No.17
Vol.2013-CSEC-60 No.17
2013/3/14

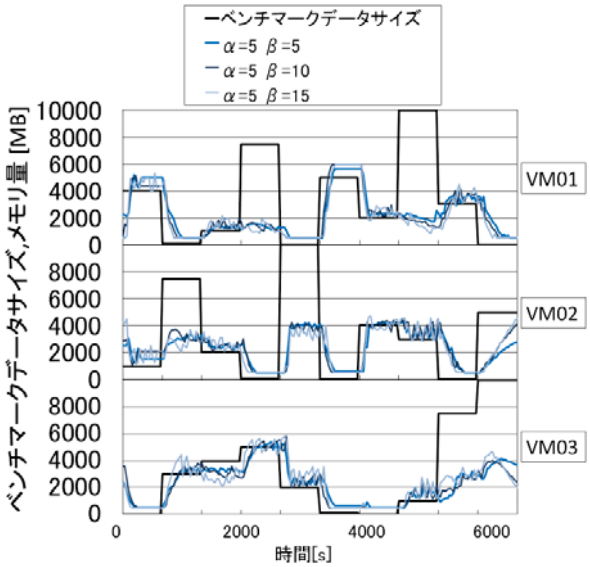

図 12　VM メモリ量変化の推移（分配量変更 α＝10）

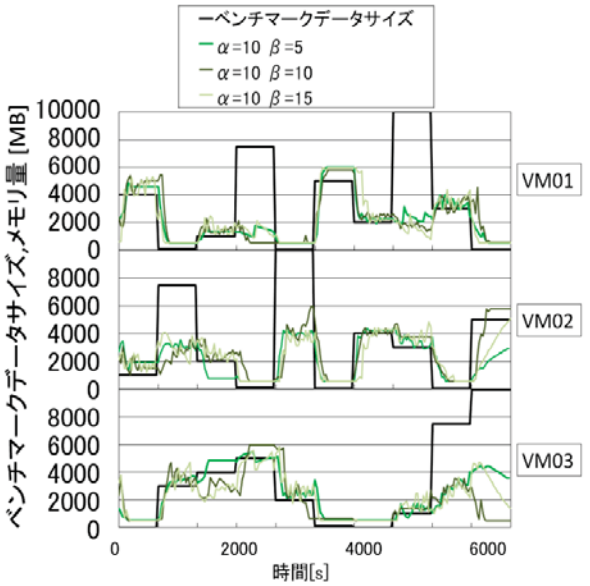

図 13　VM メモリ量変化の推移（分配量変更 α＝15）

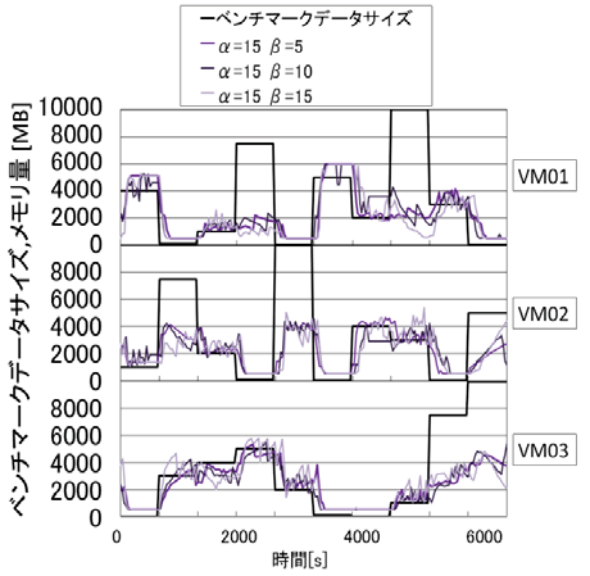

図 14　VM メモリ量変化の推移（分配量変更 α＝20）

図 9，10 より，提案手法の分配量変更で一番性能が向上した値は，α＝0，β＝15 ということが分かる．図 11，12，13，14 より，分配用メモリのための減少率が低い場合は，メモリが必要ない状態となったときのメモリ割当量の減少の速度が遅く，メモリを必要としている VM へのメモリ割当に時間が掛かってしまっていることが分かる．分配用メモリの減少率が高い場合は，メモリ割当量が 100%を維持する値になったとき，減少するメモリ量が多く減少と増加を繰り返す結果となってしまった．よって，分配用メモリのための減少率 α と β を適切に調整することにより更なる性能向上が可能であることが分かる．ただし，図 7 と図 9，10 の比較より，これらのパラメータが適切でなくても，既存手法の性能を大きく上回ることが分かる．

6.4　I/O 性能測定(書き込みが発生する場合)

読み込みと書き込みの両方が発生するアプリケーションに対する提案手法の有効性を確認するため，書き込みを含んだランダムアクセスベンチマークにてスループットを測定した．読み込みは前項の実験と同様の方法で行い，書き込みでは 1MB のファイルを新規に作成する．今回の実験では，書き込みの頻度を全体の 10%とした．実験結果を図 15，16 に示す．

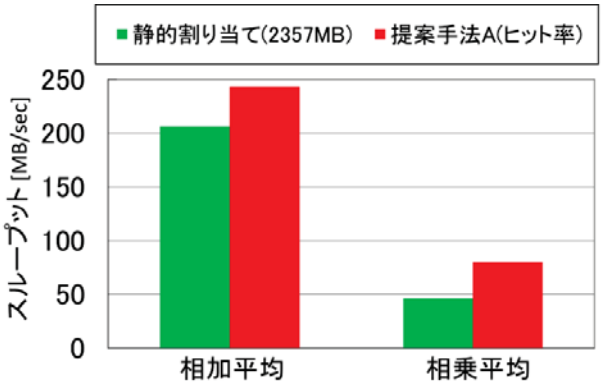

図 15　一様分布乱数と指数乱数分布による比較(相加平均)

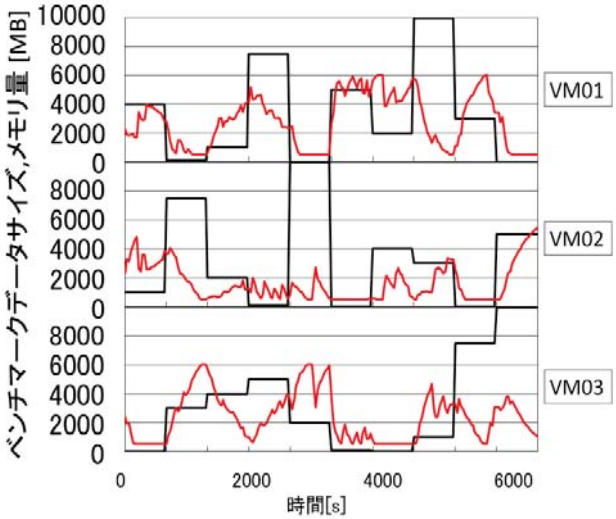

図 16　VM メモリ量変化の推移

図 15 より，書き込みが読み込みと書き込みの両方が発生する場合でも，提案手法の方が既存手法より性能が高く，提案手法が有効であることが分かる．図 16 より，書き込みが新加わった分，VM のキャッシュヒット率が 100%近い値になるために必要なメモリ量が増加しているのが分かる．

7. おわりに

本研究では，仮想計算機への動的メモリ割り当てに着目し，キャッシュヒット率に基づく手法を提案した．評価実験の結果，提案手法の性能が既存手法より優れていることが分かり，提案手法の有効性が確認された． 今後は，データベースなど応用を用いての評価を行っていく予定である．

謝辞 本研究は JSPS 科研費 22700039, 24300034 の助成を受けたものである

参考文献

1) Masaya Yamada, Saneyasu Yamaguchi, "Filesystem Layout Reorganization in Virtualized Environment," The 9th IEEE International Conference on Autonomic and Trusted Computing (IEEE ATC 2012), ATC4-2, 2012
2) Xenballooning について
http://blog.xen.org/index.php/2008/08/27/xen-33-feature-memory-overcommit/