RC-009

仮想化環境における キャッシュヒット率を考慮した VM メモリ割当

VM Memory Allocation using Cache Hit Ratio

坂本雅哉† 山口実靖†
Masaki Sakamoto Saneyasu Yamaguti

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）を停止させることなくＶＭのメモリ割当量を動的に変更することが可能である．一つの物理計算機上で複数の VM が稼働し，それぞれの VM の負荷が時間により変化する場合は，静的なメモリ割当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 xenballoon がある．しかし xenballoon 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最適化の手法として xenballoon の改変案を提案する．

2. Xenballoon

xenballoon 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変更を動的に行う機能である．VM 上でデーモンとして動作し，ゲスト OS 内のプロセスの推定メモリ使用量(Commited_AS)を監視して，ホスト OS への要求メモリ量を調整する[2].

しかし，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事がある．この場合でも，メモリ量のオーバーコミットによりエラーが発生することは無いが，先にホスト OS へ要求したゲスト OS に優先的にメモリが割り当てられてしまう．そのため，特定のゲスト OS が多くのメモリを占有してしまい，他のゲスト OS のメモリ量が増えないという現象が発生する．

†1 工学院大学工学部情報通信工学科
Department of information and Communications Engineering, Kogakuin University

3. キャッシュヒット率の解析

本稿では，ゲストＯＳにおけるキャッシュヒット率を用いてＶＭメモリ割当量を決定する手法を提案する．本節では，ゲストＯＳにおけるキャッシュヒット率の調査方法，実装について述べる．

本研究では，Linux と Xen を用いる環境を想定し，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 のカーネル内でアクセスフローにおけるページキャッシュの前と後の I/O 量を監視した．

キャッシュ前の I/O 量としては，ページキャッシュへのアクセス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を監視，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改変を行った．

キャッシュ後の I/O 量としては，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を監視，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改変を行った．

そして，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．

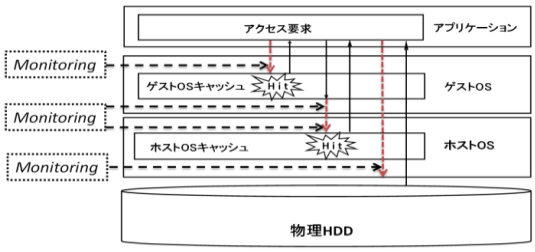

図 1 キャッシュヒット率の測定

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 測定方法 (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（シーケンシャルリード，ランダムリード）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）である．シーケンシャルリードでは，ファイルに与えたファイル番号順にファイルを読み込み，全ファイル読み込み後には先頭のファイルの戻り読み込みを繰り返す．ランダムリードでは，ファイルをランダムに選択し読み込むことを繰り返した．ファイル選択の乱数として，一様分布乱数と平均がファイル数の半分(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性能の変化を調査した．物理計算機，仮想計算機の仕様は表1，2，3の通りである．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

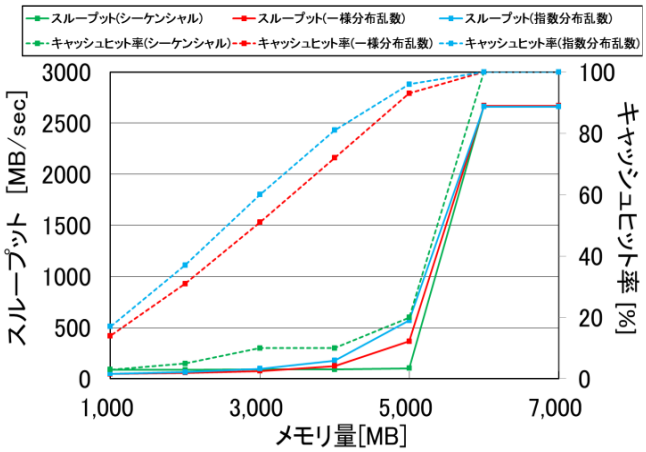

図 2 キャッシュヒット率とスループットの測定

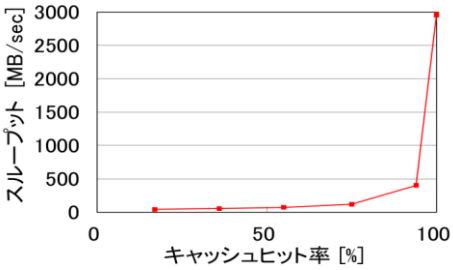

図 3 キャッシュヒット率とスループットの関係

4.2 測定結果 (キャッシュヒット率と I/O 性能)

測定結果を図 2 に示す．まず，VMメモリ割当量とキャッシュヒット率の関係について考察する．図 2 より，シーケンシャルリードの場合は，VMメモリ割当量がベンチマークデータサイズより少ないとキャッシュヒット率が 0%に近いことが分かる．また，ランダムリードの場合は，ベンチマークデータサイズよりVMメモリ割当量が少なくてもメモリ割当量に比例したキャッシュヒットが得られることが確認できる．

次に，キャッシュヒット率と性能の関係について考察する．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 3 に示す．図 3 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．また，図 2 より，一様分布ランダムリード以外の測定においても，キャッシュヒット率が低い状況ではメモリ割当による性能向上の効果が小さく，ヒット率が高い環境では効果が大きいことが分かる．

4.3 測定方法 (ホスト OS とゲスト OS のメモリ割当量)

仮想計算機環境には，ゲストＯＳキャッシュとホストＯＳキャッシュの 2 種類のキャッシュが存在する．ゲスト OS キャッシュにおいてキャッシュヒットしたときの I/O 性能と，ホスト OS キャッシュにて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，4.1 節の実験と同様に VM 内に 1MB のファイルを複数個作成し，ランダム選択したファイルに対して 1MB(ファイル全体)の読み込みアクセスを行った．ファイル数は 100 個～10000 個であり，ファイル選択の乱数としては一様分布乱数を用いた．

ＶＭメモリ割当に関しては，ゲストＯＳに多くのメモリを割り当てたゲストＯＳ 7000MB，ホスト OS1000 ＭＢの環境と，ホストＯＳに多くのメモリを割り当てたゲストＯＳ 1000MB，ホストＯＳ 7000MB の環境を用意した．物理計算機，仮想計算機の仕様は 4.1 節と同じである．

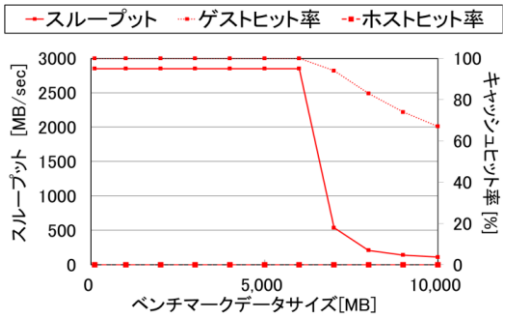

図 4　ゲスト OS メモリ 7000MB，ホスト OS メモリ 1000MB におけるキャッシュヒット率と I/O 性能

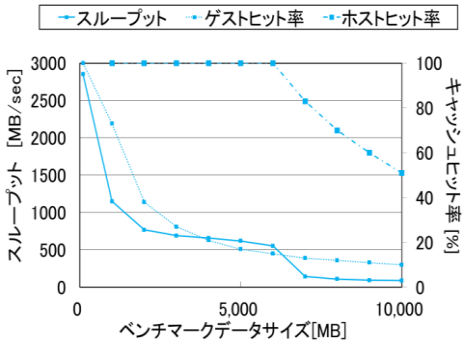

図 5　ゲスト OS メモリ 1000MB，ホスト OS メモリ 7000MB におけるキャッシュヒット率と I/O 性能

4.4 測定結果 (ホスト OS とゲスト OS のメモリ割当量)

測定結果を図 4，5 に示す．両図より，ゲスト OS に多くのメモリを割り当てると，ゲストＯＳキャッシュヒット率が高く，ホストＯＳキャッシュヒット率が低くなっており，ホストＯＳに多く割り当てると逆に結果になっていることが分かる．

また図 4，5 のスループットを比較すると，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．これは上位のキャッシュでヒットした方が少ない処理で I/O 要求が完了できるからだと考えられる．具体的には，I/O 要求をゲスト OS キャッシュからホスト OS キャッシュまで転送する過程の有無が，上位キャッシュヒットと下位キャッシュヒットで異なる．

本実験により，VM における I/O 性能の向上にはゲスト OS に多くのメモリを割り当てることが重要であることが分かる．

5. 提案手法

本章で，ゲストＯＳキャッシュのシュヒット率を監視し，それを元に VM にメモリ割当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，以下の手順により VM メモリ割当量を決める．

1. キャッシュヒット率が閾値(th%)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，VM に再配分する．

A. キャッシュヒット率が閾値(th%)以下の VM に，再配分用メモリを 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閾値(th%)以下の VM に，再配分用メモリを VM のキャッシュミス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．

ただし，上記の α には，下記の補正を施す．

(補正)　キャッシュヒット率が閾値(th%)を超えることが連続した場合，上記 1 の減少率を α から，下記の α'に変更する．これにより過剰なメモリを保持しているＶＭからのメモリの移動の速度が改善されると期待される．

$$\alpha' = (1 + \text{連続閾値超過回数} \times 0.1) \times \alpha$$

メモリ量変更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 章の手法で導出する．

提案手法は，ゲスト OS 上で動作する xenballoon の実行ファイルに変更を加え実装された．追加した機能は，キャッシュヒット率の測定機能，メモリ量とキャッシュヒット率の情報をホスト OS に伝える機能，メモリ割当量の情報をホスト OS から取得する機能である．

これらの動作は，標準状態ではインターバル γ 秒ごとに行われ，キャッシュヒット率が閾値以上(連続回数 3 未満)の場合は 1.5×γ 秒ごとに行われ，閾値以上の判定が 3 回以上連続した場合は 0.5×γ 秒ごとに行われる．キャッシュヒット率が 100%を下回ると性能が急激に低下することが 4 章の実験から確認されているが，前者の調整(1.5×γ)によりヒット率が 100%に近い状態を維持する時間が長くなると期待できる．また，後者の調整(0.5×γ)により余剰なメモリを多く抱えているＶＭからメモリを移動する速度が改善されると期待できる．

またホスト OS 上では，各 VM のメモリ割当量とキャッシュヒット率を取得，提案手法に従った各 VM のメモリ割当量の決定，メモリ割当量の変更が行われる．これらはインターバル δ 秒毎に実行される．

6. 性能評価

提案手法の有効性を確認するために，性能評価を行った．

6.1 評価方法

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．

アクセス対象ファイルの数は，600 秒ごとに変化させ，

各 VM の負荷に時間変化を与えた．アクセス対象ファイル数は 0 個，100 個，1000 個，2000 個，3000 個，4000 個，5000 個，7500 個，10000 個の中からランダムで選択される．存在しているファイルの数は常に 10,000 個であるが，ランダムアクセスでは指定された数のアクセス対象ファイルの中からアクセス対象を選択する．

実験は 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．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 節の実験の提案手法パラメータは，閾値 th%が 99%，α が 10%，β が 10%，γ が 10 秒，δ が 5 秒である．6.3 節の実験においては，α と β は変動させ，他は同一とした．

6.2 既存手法との性能の比較

既存手法(改善前の xenballoon)，各 VM にメモリを均等(2357MB)に静的に割り当てたとき，提案手法 A，B の I/O 性能を図 6，7 に示す．図 6 がスループットの相加平均，相乗平均であり，図 7 が VM に与えられた I/O 負荷量(ベンチマークデータサイズ)と割り当てられたメモリ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 より良くなっており，4 章で示した通り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が有効であることが分かる．相乗平均に注目すると，相加平均に比べ性能差が大きいことが分かる．これは，提案手法では得られる性能が安定して高く，性能が極端に低いＶＭの発生を回避できたからである．静的割り当ての場合は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が，提案手法ではベンチマークデータサイズに合わせてメモリ割当量を調整するため大幅なスループットの低下が回避できたと考えられる．

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メモリ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メモリ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 のメモリ割合が好ましいと言える．

書き込みも含むベンチマークによる評価も付録に記載する．

xenballoon により得られる性能は，静的割り当てにより得られる性能よりも低くなっている．これは，xenballoon がプロセスが使用するメモリ量のみを考慮し，ページキャッシュに使用されるメモリ量を考慮せずにＶＭ割り当てメモリ量を決定するからである．xenballoon ではページキャッシュにメモリが使用されていてもプロセスによるメモリ使用量が少ないとＶＭによる割り当てメモリの自主的な返却が行われ，結果的に低い I/O 性能となる．

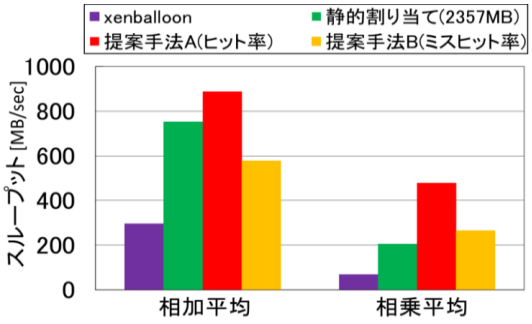

図 6　既存手法と提案手法の比較

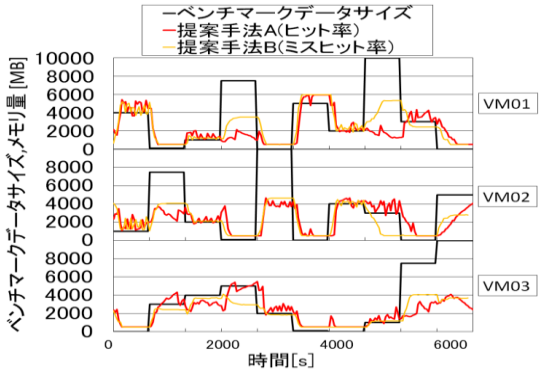

図 7　VM メモリ量変化の推移

6.3 パラメータの影響の評価

提案手法のパラメータ α と β(再配分用メモリ量)値を変化させ，性能の変化を調査した．性能測定は一様分布乱数のランダムリードにより行った．実験結果を図 8，9，10，11，12，13 に示す．

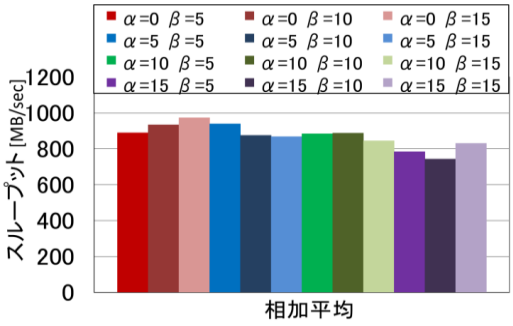

図 8　提案手法の分配量変更による比較（相加平均）

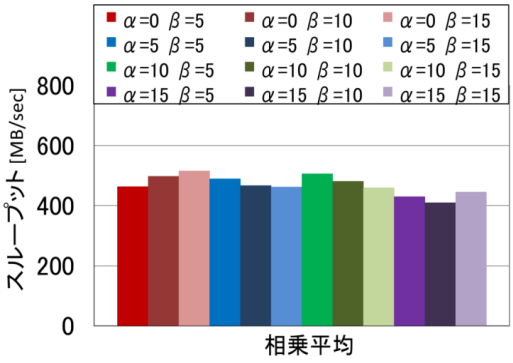

図 9　提案手法の分配量変更による比較（相乗平均）

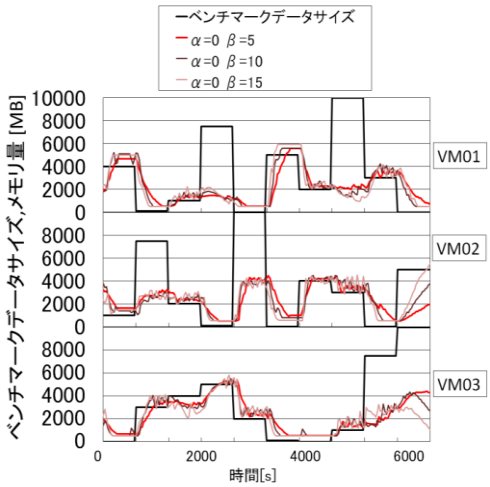

図 10　VM メモリ量変化の推移（分配量変更 α＝0）

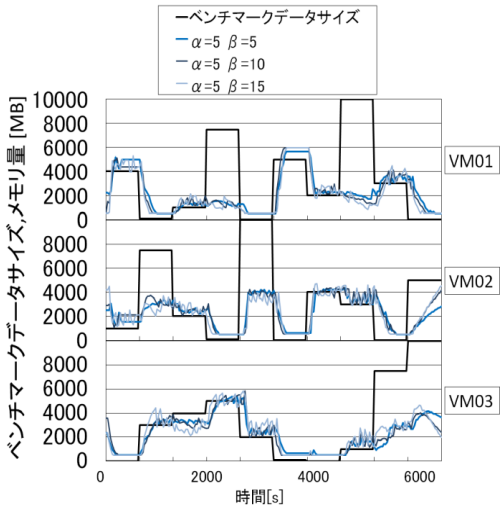

図 11　VM メモリ量変化の推移（分配量変更 α＝10）

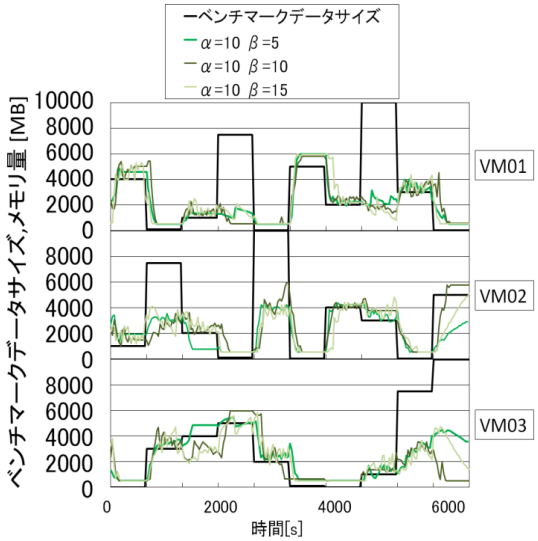

図 12　VM メモリ量変化の推移（分配量変更 α＝15）

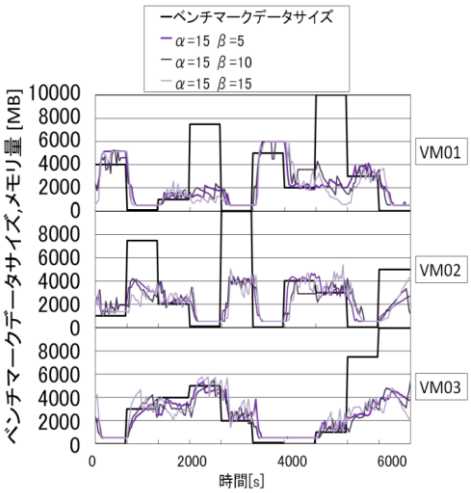

図 13　VM メモリ量変化の推移（分配量変更 α＝20）

図 8，9 より，再配分用メモリの量(α，β)を変更することにより得られる性能が変化するが，その変化が大きくないことが分かる．また，すべての例において得られる性能が既存手法の同等以上となっていることが分かる．よって，パラメータの調整が適切に行われなかったとしても提案手法は有効であると言うことができる．

提案手法の分配量変更で一番性能が向上した値は，α＝0，ß＝15 という結果となった．図 10，11，12，13 より，分配用メモリのための減少率が低い場合は，メモリが必要無い状態となった時に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た．分配用メモリのための減少率が高い場合は，メモリ割当量が 100%をキープする値になったとき，減少するメモリ量が多く，減少したり増加したりの繰り返しになってしまった．よって分配用メモリのための減少率は低すぎず，高すぎない値が適切という結果になった．また，α（キャッシュヒット率が閾値以上の VM のメモリ減少率）が低い方が良い結果となった．今回のベンチマークデータサイズとベンチマーク時間では，このような結果になったが，ベンチマークデータサイズとベンチマーク時間を変更すると，この減少率の最適な値も変化すると考えられる．

7. 関連研究

VM の割り当てメモリ量の変更に関する既存の研究としては，Waldspurger による ballooning 手法の提案[3]がある．カーネル空間にて ballooning ドライバを動作させる手法を提案した先駆的な研究であるが，本稿のように I/O 性能やキャッシュヒット率に着目した研究ではない．

Zhao らは，LRU ヒストグラムを取得してキャッシュヒット率を予測し ballooning を行う手法を提案している[4]．しかし，仮想記憶や PTE に注目した手法であり，本稿の様な I/O 性能に関して寄与のある手法とはなっていない．

仮想化環境に I/O に着目し，VM のメモリ割当について考察した研究して，Jones らの提案[5]がある．当該手法ではバッファキャッシュへの追加と削除から必要メモリ量の推定を行っている．しかし，キャッシュヒット率に

ついてのみ考察されており，最終的な性能については議論されていない．また，キャッシュヒット率の推定は必ずしも正確ではなく，実際のヒット率を求めて使用している本稿の提案手法と比較し正確さにおいて劣っている部分もある．

8. 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 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]http://blog.xen.org/index.php/2008/08/27/xen-33-feature-memory-overcommit/
[3]Carl A. Waldspurger, “Memory resource management in VMware ESX server,” OSDI '02: Proceedings of the 5th symposium on Operating systems design and implementation, 2002.
[4]Weiming Zhao, Zhenlin Wang, “Dynamic memory balancing for virtual machines,” Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, pp. 21-30, 2009.
[5]Stephen T. Jones Andrea C. Arpaci-Dusseau Remzi H. Arpaci-Dusseau, “Geiger: monitoring the buffer cache in a virtual machine environment,” Proceedings of the 12th international conference on Architectural support for programming languages and operating systems, pp. 14-24, 2006.
[6]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
[7]http://blog.xen.org/index.php/2008/08/27/xen-33-feature-memory-overcommit/
[8]Weiming Zhao, Zhenlin Wang, “Dynamic memory balancing for virtual machines,” Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, pp. 21-30, 2009

付録

性能評価 (書き込みを含む)

読み込みと書き込みの両方を含んだランダムアクセスベンチマークを用いて提案手法の性能を評価した．書き込みでは 1MB のファイルを新たに作成し，書き込みの頻度は全体の 10%である．それ以外は 6.2 節の実験と同様である．

静的割り当てと，提案手法の性能を図 14 に，提案手法適用時のベンチマークデータサイズとＶＭメモリ割当量の変化を図 15 に示す．図 14 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 15 より，書き込みが新たに加わった分，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．

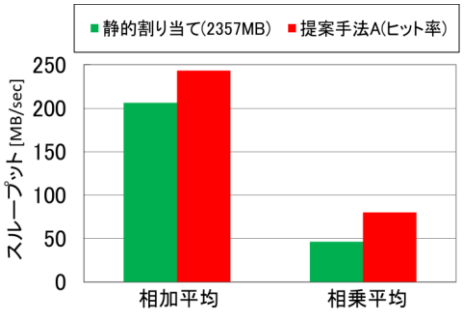

図 14　一様分布乱数と指数乱数分布による比較

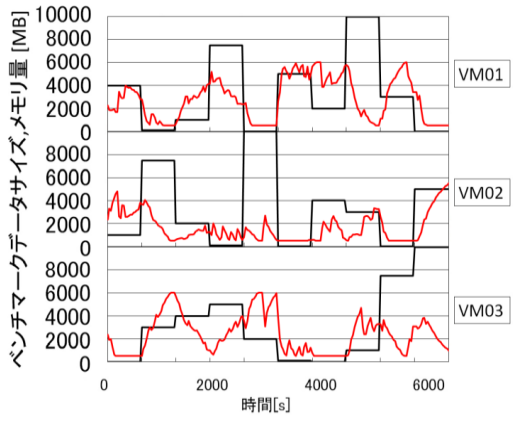

図 15　VM メモリ量変化の推移