平成 29 年度 特別研究報告書

仮想化環境のための
ディスクキャッシュの変化に着目した
動的メモリ割り当て

龍谷大学 理工学部 情報メディア学科

学籍番号　：T130440
氏名　　：石原 吉晃

指導教員　：三好 力 教授，芝 公仁 助教

内容梗概

近年，仮想化環境が普及し，1 台の計算機上で複数のオペレーティングシステム (OS) が動作する環境が増加している．仮想化環境でディスク I/O によるメモリの負荷が動的に変化する場合，静的なメモリ割り当てでは十分にメモリを活用することができない．この問題を解決するためには各仮想計算機 (VM) に割り当てるべき最適なメモリ量を動的に算出し，それぞれに割り当てる必要がある．

本論文では，各 VM が十分にディスクキャッシュを確保できるように各 VM に動的メモリ割り当てを行う手法を提案する．本手法では，VM 上で動作するプロセスが使用するメモリ量と提案機構 Balloon Controller が持つ可変量パラメータ Margin Size を合計した値を各 VM のメモリに割り当てる．Margin Size は Balloon Controller が VM のメモリ使用状況から判断して制御し，必要なディスクキャッシュを確保できるようにする．Balloon Controller を実装して評価実験を行った結果，本手法により静的メモリ割り当てと比較して I/O 性能が向上した．

目 次

1 はじめに

近年，仮想化環境が普及し，1 台の計算機上で複数のオペレーティングシステム (OS) が動作する環境が増加している．仮想化環境でディスク I/O によるメモリの負荷が動的に変化する場合，静的なメモリ割り当てでは十分に物理メモリを活用することができない．この問題を解決するためには，各仮想計算機 (VM) に割り当てるべき最適なメモリ量を動的に算出し，それぞれに割り当てる必要がある．しかし，動的メモリ割り当ての既存手法では，必要なディスクキャッシュの確保はされているが VM 上のプロセスが使用するメモリ量の考慮はされていない．また，適用するためにはゲストとホストの OS を変更する必要がある．本論文では，上記の問題に対処し，各 VM が十分にディスクキャッシュを確保できるよう，各 VM に動的メモリ割り当てを行う手法について述べる．

本論文では，提案手法を Linux OS と オープンソースソフトウェアの仮想マシンモニタ (VMM) である QEMU が動作する環境に実装した．VM の動的メモリ割り当てには，QEMU が提供している機能を使用し，各 VM 上で動作する提案機構 Balloon Controller がメモリを割り当てる．最適なメモリ量は，VM 上で動作するプロセスが使用するメモリ量と Balloon Controller が持つ可変量パラメータ Margin Size を合計した値を使用する．Margin Size は Balloon Controller が VM のメモリ使用状況から判断して制御し，必要なディスクキャッシュを確保する．このようにして，本手法では VM 上のプロセスが使用するメモリ量とディスクキャッシュを考慮して各 VM の最適なメモリ量を算出する．そして，それぞれ割り当てることで， 静的メモリ割り当てよりも物理メモリを活用し，I/O 性能が向上する．また，Margin Size を制御する際に使用する VM のメモリ使用状況は，ゲスト OS が提供している情報を使用するためホストとゲストの OS を変更する必要はない．

以下，本論文では，2 章で本研究の関連研究について述べる．3 章では，提案手法の構成を示し，4 章で提案手法の動作について述べる．5 章では，提案手法の効果を確認する評価実験について述べ，6 章で本論文のまとめを述べる．

2 関連研究

前章で述べた通り，仮想化環境でメモリを活用するためには，VM のメモリ使用状況に対応して，割り当てるメモリ量を動的に変更する必要がある．仮想化環境では，VM が使用できるメモリを動的に変化させる機能を持っている．しかし，この機能は各 VM の最適なメモリ量を算出することはできず，各 VM に割り当てる最適なメモリ量は設定する必要がある．そこで，各 VM の最適なメモリ量を動的に算出するため，いくつかの手法 [1][2][3][4] が提案されている．

仮想化ソフトウェアの Xen に実装されている xenballoon[1] では，VM 上のプロセスが確保しているメモリ量を最適なメモリ量として扱い，動的メモリ割り当てを行っている．結果，xenballoon ではプロセスが確保したメモリ量に対応した割り当てを実現している．しかしこの手法では，ディスクキャッシュが考慮されていないため I/O 性能が劣化する場合がある．また，無条件にメモリの割り当てを行うため，物理メモリ量を超えた割り当てが行われる可能性がある．

文献 [2] では，xenballoon を改善してディスクキャッシュの確保を行っている．各 VM へのメモリ割り当ては，まず各 VM から一定量のメモリを回収し，そのメモリを再配分用のメモリとする．そして，その再配分用のメモリを各 VM のキャッシュヒット率に応じた量を分配する．すなわち，この手法ではディスクキャッシュの確保だけでなく全 VM の優先度付けを行いメモリの分配を行っている．この優先度付けにより，メモリを与えることで他の VM と比較して I/O 性能が向上する VM が優先的にメモリを獲得できる．しかしこの手法では，xenballoon とは異なりホストとゲストの OS を改変する必要がある．そのため，Linux OS にしか適用することができない．また，VM 上のプロセスが使用するメモリ量を考慮せずに VM にメモリ割り当てを行うため，各 VM 上で動作するプロセスのメモリが不足している場合は対応できない．他にもこの手法が抱える問題として，メモリ割り当てのオーバヘッドが非常に大きいことが挙げられる．各 VM が必要なメモリ量を割り当てられた後もそれぞれの VM のメモリが 1 GB 以上の大きな値で変化している．そのため，このような動的メモリ割り当ての処理により，CPU に大きな負荷がかかっていると考えられる．

文献 [3] では，xenballoon と同じく VM 上のプロセスが確保しているメモリ量を使用して，動的メモリ割り当てを行っている．さらに，この手法では VM に割り当てら

れたメモリ量と定数を乗算した値をディスクキャッシュ用のメモリとして確保している．しかし，乗算する定数は VM 上で動作するゲスト OS とプロセスに適した値を事前に調査して設定する必要がある．そのため，VM 上で特定のゲスト OS とプロセスのみが動作するような環境の場合，有効な手法となる．

本論文では，VM 上のプロセスが使用するメモリ量とディスクキャッシュを監視し，各 VM の最適なメモリ量を動的に算出する．また，ディスクキャッシュへの考慮はゲスト OS が提供する情報のみを使用するため OS を変更する必要はない．

3　構成

本章では，提案機構 Balloon Controller の構成について述べる．提案手法では，各 VM にそれぞれ最適なメモリ量を動的に割り当てる．最適なメモリ量は，プロセスのメモリ使用量とディスクキャッシュを考慮して算出する．また，最適なメモリ量の算出にはゲスト OS が提供する情報を使用して算出するため，OS を変更する必要はない．

以下は，1 節で Balloon Controller がどのようして動的にメモリを割り当てるのか簡単に述べ，2 節で本手法が定義する最適なメモリ量について述べる．最後に 3 節で提案手法の全体構成を示す．

3.1　Balloon Controller の基本動作

図 1 に Balloon Controller がどのように動的メモリ割り当てを行うのかを示す．Balloon Controller は始めに各 VM のメモリ使用状況を取得する．次に取得した情報を使用して，各 VM の最適なメモリ量を算出する．最後に算出した各 VM のメモリ量をそれぞれ割り当てる．図 1 の場合は，VM1 はメモリを必要としており，VM2 は未使用のメモリを持っている．そのため，Balloon Controller は VM1 のメモリ量を増加させ，VM2 のメモリ量を減少させる．

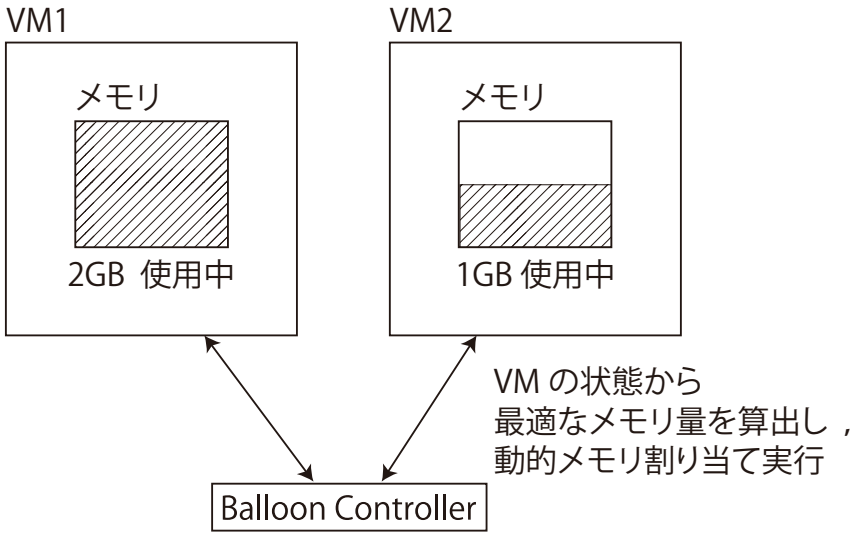

図 1: Balloon Controller の基本動作

3.2 最適なメモリ量

提案手法では，VM 上のプロセスが使用するメモリ量とディスクキャッシュを考慮して各 VM の最適なメモリ量を算出する．最適なメモリ量 W は式 (1) のように VM 上のプロセスが使用しているメモリ量と Balloon Controller が持っている可変量パラメータ Margin Size を合計した値である．

$$W = \text{使用中のメモリ量} + MarginSize \tag{1}$$

使用中のメモリ量は VM 上のプロセスが確保したメモリの合計である．この値には，物理メモリを未割り当てのメモリも含まれている．本手法では，使用中のメモリ量を考慮して各 VM に割り当てるメモリ量を算出することで，VM 上のプロセスが使用するメモリ量を割り当てる．

ディスクキャッシュへの考慮は可変量のパラメータ Margin Size を使用する．この値は各 VM が十分なディスクキャッシュを確保できるように Balloon Controller が制御する．Balloon Controller は各 VM のキャッシュ量と最近使用したキャッシュ量を考慮して Margin Size を算出する．これらの情報はゲスト OS が提供する情報を使用する．

本手法では，上記のようにプロセスが使用するメモリ量とディスクキャッシュを考慮して各 VM に最適な量のメモリを割り当てることで，物理メモリを活用し，静的メモリ割り当てよりも高い I/O 性能を実現する．

3.3 全体構成

提案手法の全体構成を図 2 示す．提案手法では，全 VM 上で Balloon Controller が動作する．本手法では Balloon Controller が，動的メモリ割り当てと前節で述べた Margin Size の制御を行う．

動的メモリ割り当ての実行は，前節で述べたように使用中のメモリ量と Margin Size を合計した値を最適なメモリ量とし，VM にメモリを割り当てる．Balloon Controller はゲスト OS から取得した使用中のメモリ量と自身が保持している Margin Size を合計して，算出した値を VM にメモリ割り当てを行う．

Margin Size は，前節で述べた通り各 VM で十分なディスクキャッシュが確保できるように制御する．Margin Size の制御を行う際，Balloon Controller はまずゲスト OS

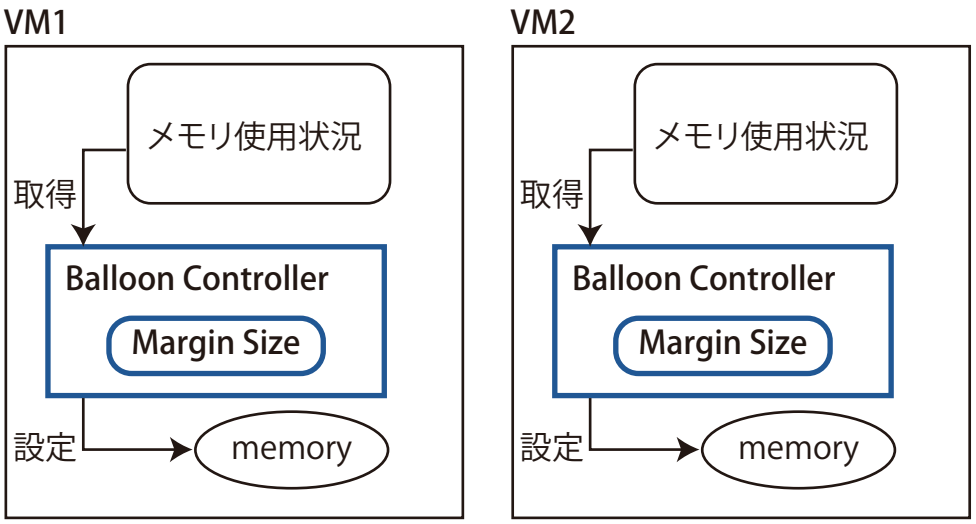

図 2: 全体構成

の情報を取得する．そして，取得した情報を元に Margin Size を算出する．最後に算出した値を自身が保持しているパラメータ Margin Size に設定する．

4 動作

提案機構 Balloon Controller を Linux カーネルと VMM である QEMU を使用した環境に実装した．動的メモリ割り当ては，QEMU に実装されている Balloon の機能を使用して行う．Balloon は，VM のメモリ上で膨張，収縮してメモリの動的割り当てを実現している．Balloon の機能は，実行中の QEMU を制御することができる QEMU Monitor に対して balloon コマンドを発行して使用する．また，ゲスト OS のメモリ使用状況は /proc/meminfo から取得した．前章で述べたプロセスが使用しているメモリ量は Committted_AS，キャッシュ量は Cached，最近使用したキャッシュ量は Active(file) を使用している．

本章では，提案手法の動作について述べる．1 節で，Balloon Controller が動的メモリ割り当てを行う手順を述べ，2 節で，Margin Size を制御する手順を述べる．

4.1 動的メモリ割り当て

Balloon Controller は QEMU が持つ機能を使用して動的メモリ割り当てを行う．図 3 にその手順を示し，以下に処理の流れを述べる．

(1) Committed_AS の取得

(2) QEMU Monitor に対してコマンドを発行

(3) 1 秒間停止

始めに，/proc/meminfo から Committed_AS を取得する．前章で述べた通り取得した Committed_AS と Balloon Controller が保持している Margin Size を合計し，最適なメモリ量とする．次に QEMU Monitor に算出したメモリ量を指定して balloon コマンドを発行し，動的メモリ割り当てを要求する．QEMU が動的メモリ割り当てを要求されると QEMU に実装されている virtio backend driver とゲスト OS に実装されている virtio-balloon が virtio[5] を使用して通信を行い，動的メモリ割り当てを実行する．Balloon Controller は balloon コマンドを発行した後，1 秒間停止する．

以降，上記の処理を繰り返す．このように本手法では，1 秒間隔で balloon コマンドを QEMU Monitor に発行し，動的メモリ割り当てを実行する．

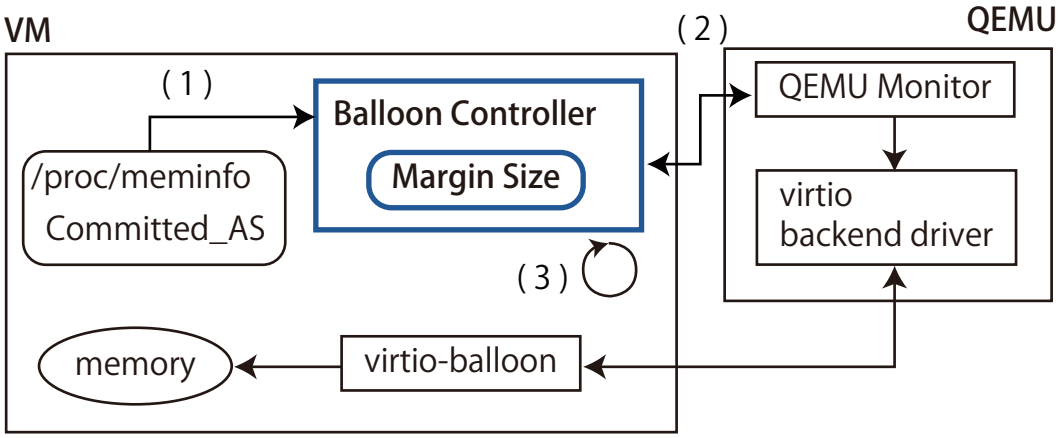

図 3: 動的メモリ割り当ての実行

4.2 Margin Size の制御

本節では，Balloon Controller が Margin Size を制御する手順について述べる．図 4 にその手順を示し，以下に処理の流れを述べる．

(1) /proc/meminfo から必要な情報を取得

(2) 取得した情報から判断して，Margin Size の値を更新

(3) 一定時間停止

まず /proc/meminfo からメモリ使用状況の情報を取得する．次に取得した情報から判断して Margin Size を算出し，更新する．Margin Size の更新後は，5 秒間停止する．以降，上記の処理を繰り返す．

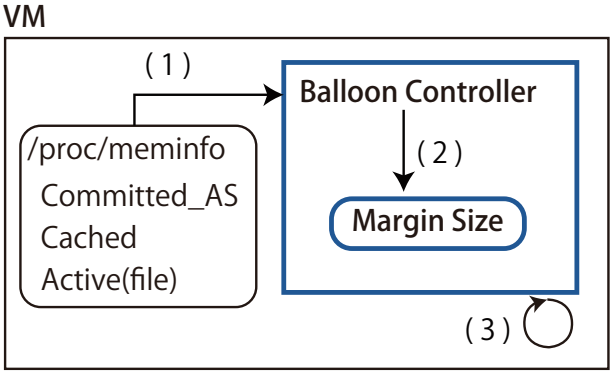

図 4: Margin Size の制御

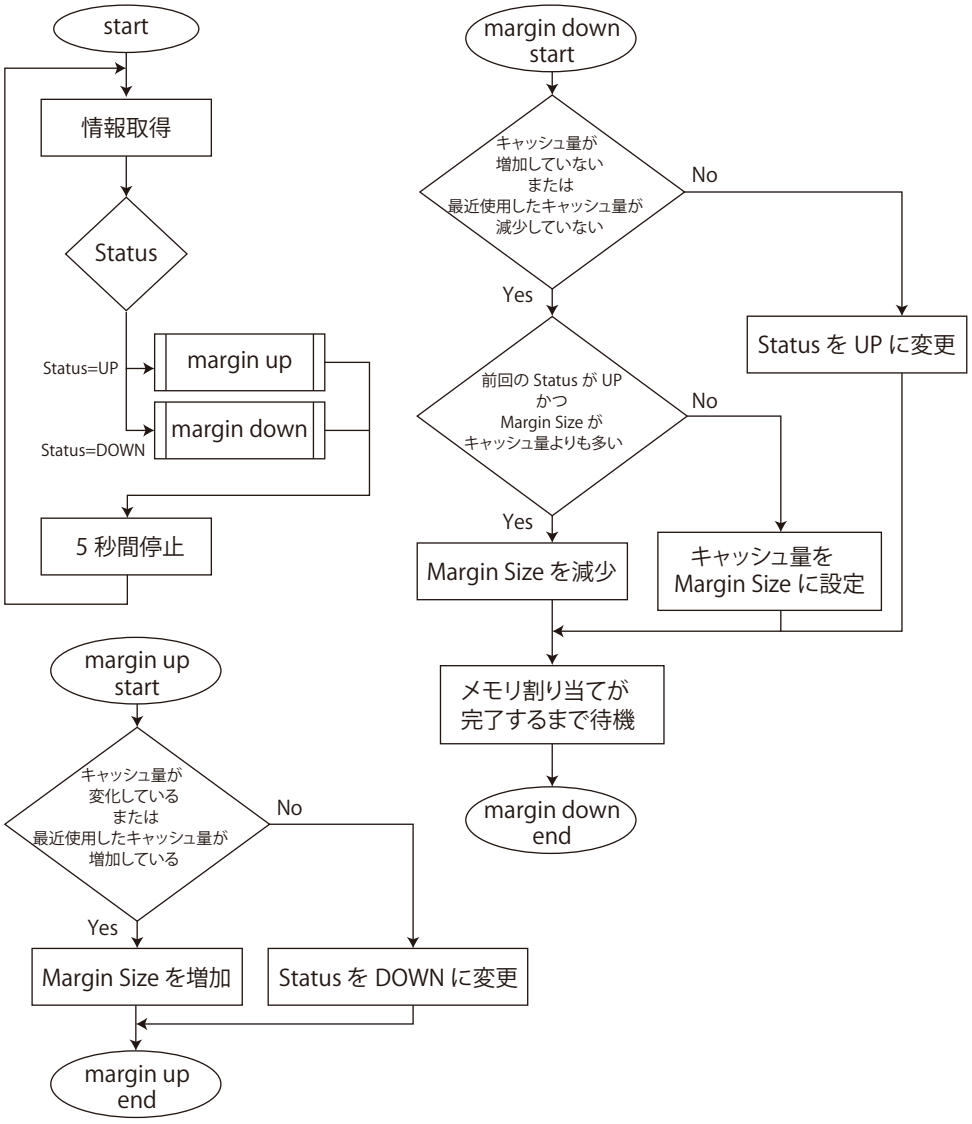

図 5: Margin Size の制御手順

図 5 のように Margin Size の制御では，VM に対して UP か DOWN の状態を付与し，その状態に応じて Margin Size の増減を行う．

状態が UP の場合は，キャッシュ量が変動している，または最近使用されたキャッシュが増加している場合に Margin Size を増加させる．そうでない場合は，状態を DOWN に変更する．

VM 上で一定の I/O 量が発生している場合，その I/O 量と同じ量のキャッシュが確保されると，キャッシュ量は変化しなくなる．本手法では，キャッシュが変化しなくなるまで Margin Size を増加させることで VM 上の I/O 量に適したメモリの割り当て

を行っている．キャッシュ量が変化しなくなった場合，現在のキャッシュ量が最適な量である場合と現在のキャッシュ量よりも VM 上で発生している I/O 量の方が低い場合が考えられる．そのため，VM の状態が UP でかつキャッシュ量が変化しなくなった場合，VM の状態を DOWN に変更して最適なキャッシュ量を確認するため一度キャッシュ量を減少させる．

また，Margin Size は式 (2) のように UP 状態が続くにつれて増加する．上昇量の上限値は 200MB とする．

$$\text{上昇量} = 25 \times UP \text{の連続回数}\,[MB] \tag{2}$$

DOWN の場合，キャッシュ量が増加していない，または最近使用したキャッシュ量が減少していない場合に Margin Size を減少させる．メモリ量を減少する際は物理メモリを回収する必要があり，処理が完了するまで時間がかかる．そのため，Margin Size を減少させる際は，動的メモリ割り当てが完了するまで待機する．また，Margin Size の下限値は 100 MB とする．上記の Margin Size を減少させる条件に当てはまらない場合は，状態を UP に変更する．

Margin Size を減少させ動的メモリ割り当てが行われると，キャッシュ用のメモリが減少する．その時，キャッシュ用のメモリが不足した場合は，キャッシュの上書きによって本来有効に活用できるはずのキャッシュが削除されてしまう．そのため，Margin Size の減少により，有効なキャッシュが減少した場合はキャッシュ用のメモリが不足していると判断し，状態を UP に変更する．また，キャッシュ量が増加した場合も新たな I/O が発生したとして状態を UP に変更する．

Margin Size は，式 (3) のように DOWN 状態が続くにつれて減少する．減少量の上限値は UP の場合と同じく 200MB とする．

$$\text{減少量} = 50 \times DOWN \text{の連続回数}\,[MB] \tag{3}$$

また，Margin Size を減少する場合は，前回の状態が UP であり，Margin Size がキャッシュ量よりも多いならばキャッシュ量を Margin Size に設定する．

以上の処理は状態変化の有無に関わらず 5 秒間隔で行う．上記のように Margin Size を制御し，各 VM にメモリを割り当てることで各 VM の I/O 量に適したディスクキャッシュが確保される．

5 評価

本章では，I/O 量に応じて Balloon Controller が適切に Margin Size を制御し，静的メモリ割り当てと比較して I/O 性能が向上しているか確認する．1 節で，評価実験について述べ，2 節でその実験結果を示す．そして，最後に 3 節で考察を述べる．

5.1 評価方法

実験では，1 台の物理計算機上で VM を 2 台同時に稼働させ，各 VM 上でファイル読み込みを行う計測プログラムを動作させる．計測プログラムは 30 分間 ファイルの読み出しを行う．読み出すファイルは 1 万個のファイルから一定数のファイルを選択する．ファイルは一様分布乱数を使用して選択する．各ファイルの容量は 1MB で，読み出し対象となるファイルの数は 10 分ごとに変化させる．読み出すファイルの数は各 VM で 500，0，500 と 0，500，0 のように一定数のファイル読み出しを交互に行う．また，読み出すファイル数は 500 から 7000 個まで 500 刻みで変化させる．以上の実験を提案手法を適用した環境と均等にメモリを分配した静的メモリ割り当ての環境で行い，結果を比較する．

表 1，2 に実験で使用する環境を示す．実験環境では，物理メモリ量が 4096 MB で，提案手法を適用した環境の各 VM は 3072 MB 以下のメモリが動的に割り当てられる．均等にメモリを分配した静的メモリ割り当ての環境は各 VM に 1536 MB のメモリを割り当てる．

表 1: 物理計算機の構成

OS	Debian 9.2
Kernel	Linux-4.11.2
CPU	Core i5-6600 3.30GHz
CPU Core	4
Memory	4GB
HDD	2TB

表 2: 仮想計算機の構成

OS	Ubuntu 17.10
Kernel	Linux-4.13.0
Virtual CPU Core	1
Virtual Memory	計測によって変動

5.2 実験結果

図 6，7 に 2500 MB のファイルを 2 つの VM で交互に読み出した時の各 VM のメモリ量の変化を示す．縦軸は VM が読み出すファイルサイズと VM に割り当てられたメモリ量，横軸は計測プログラム開始からの経過時間である．図中の opti_size は VM 上で動作しているプロセスが読み込むファイルの容量で，MemTotal は VM に割り当てられたメモリ量である．図 6 から，読み出すファイル量が変化する約 600 秒で，遅れてはいるが不要なメモリが回収されて，VM のメモリ量が減少している．また読み出すファイルが増加して必要なメモリ量が増加する約 1200 秒でも，そのファイル量に対応して VM に割り当てられたメモリ量が増加している．図 7 も図 6 と同じように読み出すファイル量が変化する約 600 秒と約 1200 秒で，メモリ量が読み出すファイル量に対応して増減している．以上のことから，I/O 量の増減に対応して Balloon Controller が適切に Margin Size を制御していることが確認できた．

図 8 に，読み出しの速度を示す．縦軸は 1 秒間で読み出したファイルサイズ，横軸は VM 上で読み出すファイルサイズである．提案手法を適用した環境で，読み出すファイル量が 1500 MB 以上の場合，静的メモリ割り当てと比較して読み出し速度が最大約 9.5 倍に向上していた．しかし，約 1500 MB 以下の場合，読み出し速度が約 14 % 劣化していた．また，読み出すファイル量が 3000 MB 以上になると，提案手法を適用した環境でも読み出し性能が劣化していった．そして，読み出すファイル量が 6000 MB 以上では提案手法を適用した環境も静的メモリ割り当てと変わらない性能となった．

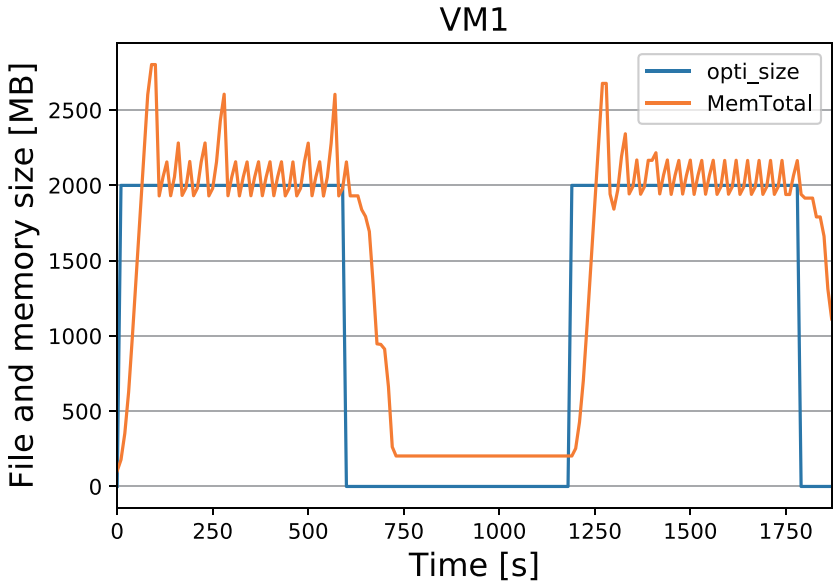

図 6: 読み出すファイル量が 10 分ごとに 2000，0，2000 と変化する VM のメモリ量変化

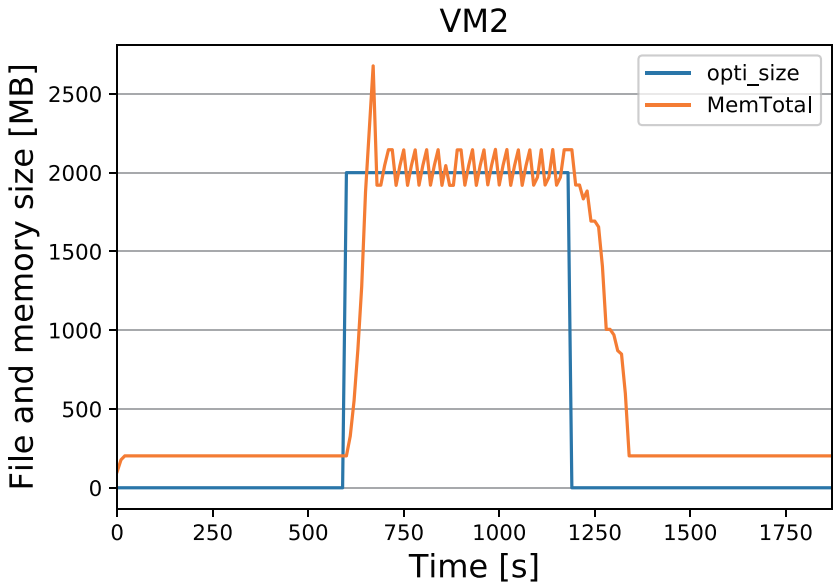

図 7: 読み出すファイル量が 10 分ごとに 0，2000，0 と変化する VM のメモリ量変化

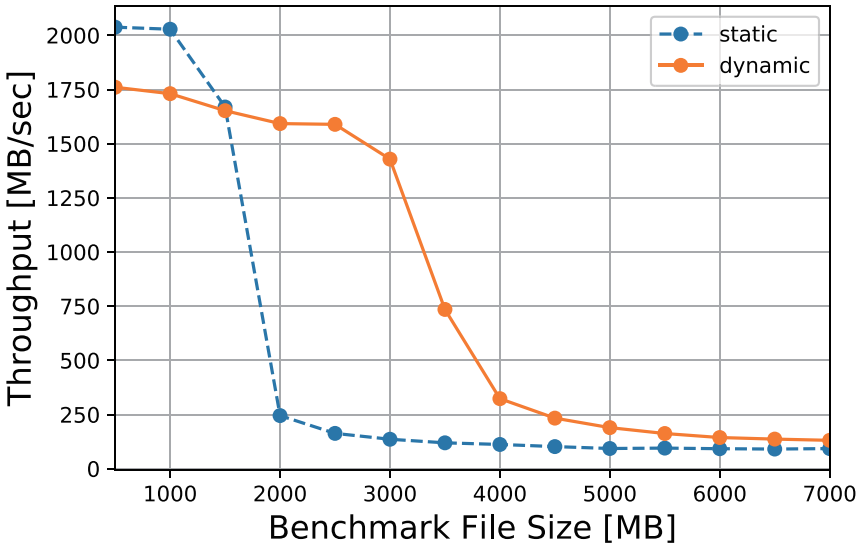

図 8: 提案手法と静的メモリ割り当てのファイル読み出し速度

5.3 考察

図 6，7 から，Margin Size がうまく制御され，ファイル読み出しに対応して，VM のメモリ量が割り当てられていることを確認した．

図 6 で約 600 秒から約 1200 秒の間，メモリ量が変化しなかったのは，Margin Size の下限値が 100 MB に設定されているからであると考えられる．Balloon Controller が Margin Size を制御する際，VM の状態が DOWN で 100 MB 以下の値が Margin Size として算出された場合，Margin Size を 100 MB に設定する．また，DOWN から UP の状態に遷移するためには，キャッシュ量が増加するか最近使用したキャッシュ量が大きく減少する必要がある．ファイル読み出しが発生しない場合はこの条件には当てはまらない．そのため，ファイル読み出しが停止した VM は，100 MB まで Margin Size が減少し，UP に状態が変更されることもなく DOWN の状態が続き，VM のメモリ量が変化していなかったと考えられる．

図 8 から，読み出すファイルのサイズが一定量を超えると提案手法を適用した環境の方が静的メモリ割り当てよりも読み出し速度が向上した．これは，静的メモリ割り当てよりも，提案手法を適用した環境の方が使用できるメモリ量が多いからであると考えられる．静的メモリ割り当ての場合，均等にメモリを分配し，1536 MB のメモリ

しか使用できない．一方，提案手法を適用した環境では，最大 3072 MB のメモリを使用することができる．そのため，確保できるキャッシュ量に差ができ，静的メモリ割り当てと比較してファイル読み出し速度が向上した．そして，読み出すファイルサイズが 3000 MB を超えると確保できるキャッシュ量以上の読み出しが発生し，提案手法を適用した環境もファイルの読み出し速度が劣化していった．

また，1500 MB までは，静的メモリ割り当ての方が提案手法を適用した環境よりもファイルの読み出し速度が高かった．これは，Balloon Controller が常にメモリ量を変化させているため，最適なメモリ量を長時間維持できないことが原因として考えられる．

6　おわりに

本論文では，プロセスが使用するメモリ量とディスクキャッシュを考慮した VM への動的メモリ割り当ての手法について述べた．本手法では，各 VM に対して十分なディスクキャッシュを確保できるように動的メモリ割り当てを行う．本手法を実装して実験を行った結果，読み出すファイルサイズが一定量を超えると，静的メモリ割り当てよりも本手法の方が読み出し速度が向上した．

本手法では，物理メモリ量への考慮をしていないため，物理メモリ量以上の I/O が発生した場合，物理メモリ量を超えたメモリ割り当てが発生してしまう．そのため，上記問題を考慮した動的メモリ割り当ての検討が今後の課題である．

謝辞

本研究を行うにあたり，終始熱心なご指導とご鞭撻を頂いた三好力教授，芝公仁助教授に心から感謝致します．また本研究を進めるにあたり，有意義な助言を頂いた芝研究室の院生の方に深く感謝を致します．最後に日頃参考となる貴重な意見やご協力を頂いた芝研究室及び三好研究室の皆様に感謝致します．

参考文献

[1] Stephen Spector. Memoryovercommit. https://blog.xenproject.org/2008/08/27/xen-33-feature-memory-overcommit/, 2008.

[2] 日名川幸矢, 竹内洸祐, 山口実靖. 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. 研究報告マルチメディア通信と分散処理（DPS） 2013-DPS-154(17), 03 2013.

[3] Anna Melekhova, Larisa Markeeva. Estimating working set size by guest os performance counters means. *The Sixth International Conference on Cloud Computing, GRIDs, and Virtualization*, 2015.

[4] Automatic ballooning. https://www.linux-kvm.org/page/Projects/auto-ballooning, 2013.

[5] Rusty Russell. virtio: Towards a de-facto standard for virtual i/o devices. 2008.