情報処理学会論文誌　コンピューティングシステム　Vol.8　No.1　22–33 (Mar. 2015)

クラウド環境におけるキャッシュ利用効率向上を目的としたメモリスケジューラの提案

田島 幸恵[1,a)]　竹内 理[1]　野中 裕介[1]

受付日 2014年7月22日, 採録日 2015年1月1日

概要：普及の進むクラウドコンピューティング環境においては，リソースが複数 VM 間で共有されるため，リソースの最適配置を行うリソーススケジューラが求められる．たとえば IO インテンシブな VM を複数稼動させる環境では，アクセス局所性が低いが IO 負荷の高い VM がキャッシュ領域を大量に消費し，アクセス局所性が高いが IO 負荷の低い VM のキャッシュヒット率を下げるディスクキャッシュポリューション問題が発生する．本論文では，サーバのキャッシュ領域を有効活用しディスク IO を削減することで上記課題を解決するキャッシュアウェアメモリスケジューラ（CAMS）を新規に提案し，その実装，評価を行った．CAMS では，ハイパバイザが，ゲスト OS のキャッシュ領域全体に対するキャッシュヒットした領域の比率であるキャッシュ利用効率を監視し，各ゲスト OS のキャッシュ利用効率を均一にするようにメモリスケジューリングを行うことでディスクキャッシュポリューション問題の解決を図る．CAMS によるディスク IO 削減効果を定量的に評価した結果，最大で，比較対象としたスケジューラの 5.8 倍の IO 削減が実現できることを確認した．また，この効果はアクセス局所性の低い VM のディスク IO 負荷が高いほど高まることを確認した．

キーワード：クラウド，ハイパバイザ，メモリスケジューラ，キャッシュアウェア，ディスク IO 削減

Proposal of Memory Scheduler for Improving Cache Usage in Cloud Computing

Sachie Tajima[1,a)]　Tadashi Takeuchi[1]　Yusuke Nonaka[1]

Received: July 22, 2014, Accepted: January 1, 2015

Abstract: The demand of appropriate resource scheduler in cloud computing is increasing, because resource are shared in multiple VMs. When there are multiple IO intensive VMs, a disk cache pollution problem occurs. The problem is that the cache hit rate of one VM, whose access locality is high but load is light, decreases, when other VM, whose access locality is low but load is heavy, consumes a lot of cache pages. In order to solve this problem, we have proposed, implemented and evaluated cache aware memory scheduler (CAMS). CAMS runs in hypervisor layer, and monitors cache usage of each guest OS. It calculates cache utilization ratio, which is the ratio of cache hit pages to the whole cache pages, and adjusts allocated memory sizes of each VM so that each cache utilization ration becomes equal to avoids a disk cache pollution. We evaluated the IO reduction effect of CAMS quantitatively. We confirmed that CAMS can increase disk IO reduction amounts by 5.8 times at maximum in comparison with another scheduler. We also confirmed that the heavier disk IO load becomes, the bigger disk IO reductions amount becomes.

Keywords: cloud, hypervisor, memory scheduler, cache aware, disk IO reduction

1. はじめに

近年のクラウドの普及にともない，仮想化技術を利用し，1 台の物理サーバ上に複数の VM を搭載して動作させるこ

[1] 株式会社日立製作所
Hitachi Ltd., Yokohama, Kanagawa 244–0817, Japan
[a)] sachie.tajima.ts@hitachi.com

とが増えてきている．クラウド環境では，CPU やメモリと比較してデータ転送速度の劣るディスクにおいてボトルネックが発生しやすいと報告されている [1]．クラウド環境によっては，ディスクボトルネック発生時の性能劣化を低減するために，ディスク IO 帯域の範囲を一定に保つサービスを提供している [2]．

IO 帯域に上限が設定された場合に VM の最大 IOPS を向上させる方法の 1 つに，ディスクへのアクセスを減らすキャッシュの活用があげられる．VM 環境下でキャッシュの有効活用を実現する手法の 1 つにハイパバイザ層のメモリスケジューリングがあり，そのスケジューリング方式には以下の 3 方式が考えられる．

(a) VM 間または VM とハイパバイザの間のメモリ重複を排除し，VM に割り当てることのできるメモリ量を増やす，メモリ重複排除方式 [3], [4], [5], [6]
(b) ハイパバイザがキャッシュを集中的に管理する一括キャッシュ管理方式 [7], [8]
(c) 高キャッシュヒット率が期待できる VM に優先的にメモリを割り当てるメモリ割当て量調整方式

著者らの調べた限りキャッシュの有効活用を実現する既存メモリスケジューラは方式 (a) および (b) に基づいており，方式 (c) に基づくメモリスケジューラは提案されていない．

IO インテンシブな Web サーバや NFS サーバなどの，異なるキャッシュヒット率を示す複数の VM が動作する環境下では，ディスクアクセスの局所性が低くキャッシュヒット率が低い VM が他の VM と同様にキャッシュ領域を消費するため，ディスクアクセスの局所性が高くキャッシュヒット率が高い VM のキャッシュ領域が圧迫されることがある．本問題をディスクキャッシュポリューション問題と呼ぶ．このような各 Web サーバや NFS サーバはそれぞれ別々のディスク領域にアクセスするため，方式 (a) によるキャッシュ利用効率の向上は期待できない．一方，方式 (b)，(c) では各 VM のキャッシュアクセス状況に応じたメモリスケジューリングにより，ディスクキャッシュポリューション問題発生時にキャッシュ利用効率を向上させることが期待できる．

また，クラウド環境では多種のゲスト OS が稼働する [9] が，各ゲスト OS においては信頼性と呼ばれる正しい処理を担保する性質が求められる [10]．近年の汎用 OS は，十分な信頼性を持つといわれている [11] が，メモリスケジューラを適用するためにゲスト OS を改変することで信頼性が損なわれる可能性がある．本問題を信頼性保証問題と呼ぶ．方式 (b) は適用にともなうゲスト OS 改変の影響範囲が大きいため信頼性の保証には困難がともなう．

そこで本論文では，ディスクキャッシュポリューション問題および信頼性保証問題に対応する新規のハイパバイザ層のメモリスケジューラとしてキャッシュアウェアメモリスケジューラ（CAMS）を提案する．さらに，CAMS を実装し，その効果の定量的な評価を行う．

本論文では，以降 2 章で，本研究で提案する CAMS で解決しようとしているディスクキャッシュポリューション問題および信頼性保証問題の概要と解決方式について説明する．3 章では，CAMS によるディスクキャッシュポリューション問題の解決方針と，その実現構成を提案する．4 章で CAMS の実装方法について示す．5 章で CAMS によるキャッシュ利用効率改善についての定量的な評価実験について述べる．6 章で本研究の関連研究を紹介し，7 章で本論文のまとめを述べる．

2. 既存メモリスケジューラの問題点と解決方式

本章では，1 章で既存メモリスケジューラにおける問題としてあげたディスクキャッシュポリューション問題および信頼性保証問題について述べる．また，他方式と比較することで，方式 (c) を用いると信頼性保証の難化を抑えつつ，ディスクキャッシュポリューション問題を解決できることを述べる．

2.1 ディスクキャッシュポリューション問題

本節では, CAMS で解決しようとしているディスクキャッシュポリューション問題 [12] の概要を説明する．これは，過去にアクセスがあったが最近はアクセスのないデータがキャッシュに存在することで，キャッシュヒット率を低下させる現象のことである．一般に，アプリケーション間に発生する現象として知られているが，本問題は VM 間でも発生しうる．シンプルなメモリスケジューリングを行う比較対象メモリスケジューラについて述べた後，当該メモリスケジューラで発生する問題点について説明する．

2.1.1 比較対象メモリスケジューラ

比較対象メモリスケジューラとして，改変した KVM のメモリスケジューラとバルーニング [13] を組み合わせたメモリスケジューラ（シンプルメモリスケジューラ：SMS）を想定する．以下，SMS の動作の概要を示す．

KVM は各ゲスト OS をプロセスとして管理し，各ゲスト OS に割り当てる物理メモリ領域（キャッシュメモリ領域を含む）を LRU 方式で管理する．オーバコミット環境で物理メモリ不足になった場合，KVM のメモリスケジューラは，通常スワップ動作を行う．スワップが発生すると，メモリアクセスごとにディスクアクセスが発生するため性能が劣化する．そこで SMS では，KVM のメモリスケジューラを改変し，物理メモリ不足を契機にスワップアウトすべき物理ページを割り当てている VM のバルーンサイズを拡大し，物理メモリを回収することを想定する．VM 間のメモリ融通により，ディスクアクセスを発生させることなく動作の実行が可能となる．このとき，スワップアウトす

べき物理ページ分だけ当該 VM 上で動作するゲスト OS のキャッシュ領域サイズが削減され，当該領域が他のゲスト OS の物理メモリとして割り当てられる．

2.1.2　問題点

同一物理サーバ上で IO 発行頻度は高いがディスクアクセスの局所性が低い高負荷 VM と，IO 発行頻度は低いがディスクアクセスの局所性が高い低負荷 VM が稼働する構成を想定する．このとき，高負荷 VM のキャッシュヒット率は低く，低負荷 VM のキャッシュヒット率は高くなることが期待される．また，メモリはオーバコミットされていると想定する．

高負荷 VM は IO 発行のたびに，IO データをキャッシュ領域に追加し，当該キャッシュ領域へのアクセスを行う．その際，当該領域に物理ページが割り当てられていないと，他 VM から物理ページを回収しなければならない．SMS は，LRU 方式に従い，最も長い間アクセスされていない物理メモリの回収を試みる．

IO 頻度の違いから，この回収対象となる物理メモリは低負荷 VM に割り当てられている物理メモリとなる可能性が高く，低負荷 VM のバルーンサイズが拡大する．結果として，低負荷 VM のキャッシュ領域は削減される．

この結果，システムの大部分のメモリが高負荷 VM に割り当てられる．一方で低負荷 VM に割り当てられるメモリ量は小さくなる．そして，キャッシュを活用しない高負荷 VM は多くのキャッシュ領域を利用可能となり，キャッシュを活用する低負荷 VM は少ないキャッシュ領域しか利用できない．この結果，低負荷 VM のキャッシュヒット率が低下し，システム全体のキャッシュ利用効率が低下する．この問題をディスクキャッシュポリューション問題と呼ぶ．

2.2　信頼性保証問題

クラウド環境では，多くの種別，多くのバージョンのゲスト OS の稼働に対応することが求められる．クラウド環境上のゲスト OS の要件は，非クラウド環境上の OS の要件と変わらず，クラウド環境上のゲスト OS にも信頼性保証が必要と考えられる．

ゲスト OS に無改変の汎用 OS を用いる場合，信頼性について新たに考慮する必要はない．しかしゲスト OS の用いる機能が汎用 OS では未提供の場合，改変した汎用 OS をゲスト OS として用いる必要がある．この場合，改変後も信頼性を保証する必要があるため，改変箇所の影響範囲が大きいほど信頼性の保証は困難となる．この問題を信頼性保証問題と呼ぶ．

汎用 OS の改変にあたっては，改変することで得られる新機能提供などのメリットと，クラウド環境で稼働する多種のゲスト OS すべてにおいて，改変後の信頼性を保証する必要があることのデメリットを比較する必要がある．

2.3　解決方式

前章で，ディスクキャッシュポリューション問題の解決が期待できる方式として (b) と (c) をあげた．本節では，各方式における信頼性保証の難易度を比較し，信頼性保証の難化を抑えつつ，ディスクキャッシュポリューション問題を解決する方式を検討する．

方式 (b) では，キャッシュアクセスが発生すると，ゲスト OS はハイパバイザに当該キャッシュ領域の範囲を通知する必要がある．この処理はゲスト OS の既存のキャッシュ管理機構とは異なるため，既存の処理およびデータ構造を無効化したうえでハイパバイザへの通知処理を追加する必要がある．改変箇所の影響範囲は大きく，信頼性保証の難易度は高いといえる．

方式 (c) では，ゲスト OS がキャッシュヒット率をハイパバイザに通知し，ハイパバイザが各ゲスト OS からのキャッシュヒット率を集約して各 VM へのメモリ割当て量を決定し，ゲスト OS 上のバルーンドライバを介してメモリ割当て量を変更する．これはゲスト OS の既存のキャッシュ管理機構に監視機構を追加することで実現が可能であり，既存の処理およびデータ構造を変更する必要はない．改変箇所の影響範囲は小さく，信頼性保証の難易度は低いといえる．

以上より，ディスクキャッシュポリューション問題を解決するメモリスケジューラとしては，方式 (b) よりも方式 (c) を用いた方が，信頼性保証は容易と考えられる．

本論文では，ゲスト OS の改変にともなって信頼性保証が難化するデメリットがあったとしても，ディスクキャッシュポリューション問題が解決できるメリットによる価値があることを確認するために，方式 (c) を用いたキャッシュアウェアメモリスケジューラ（CAMS）を提案し，効果を定量的に評価する．

3.　キャッシュアウェアメモリスケジューラの提案

本章では前章で述べたディスクキャッシュポリューション問題の CAMS における解決方針および解決方式について述べる．

3.1　解決方針

CAMS では，各 VM のキャッシュ利用効率に応じてメモリ割当て量を調整することで，ディスクキャッシュポリューション問題の解決を図る．

調整によって，キャッシュヒット率の高い VM へのメモリ割当て量を増やし，キャッシュヒット率の低い VM へのメモリ割当て量を減らす．ゲスト OS として Linux [*1] が使用されていると想定すると，各ゲスト OS は，未使用メモ

[*1] Linux は，Linus Torvalds 氏の日本およびその他の国における登録商標または商標です．

リ領域の大半をキャッシュ領域として使用している．そのため，上記メモリ割当て量の調整により，キャッシュを活用しない高負荷 VM が消費するキャッシュ領域が制限され，キャッシュを活用する低負荷 VM は十分なキャッシュ領域を利用可能になる．

3.2　メモリ割当て量調整方式

CAMS では，VM 上で稼働するゲスト OS に改変を加え，キャッシュ利用効率の監視を可能にする．監視結果をハイパバイザに集約して，各 VM のキャッシュ利用効率に応じたメモリのスケジューリングを行う．

監視対象であるキャッシュ利用効率を以下のように定義する．

$$\text{キャッシュ利用効率} = \frac{\text{ヒットページ数}}{\text{キャッシュページ数}}$$

キャッシュ利用効率はリード IO に関する情報のみ用いて算出する．ライト IO はキャッシュのヒット/ミスによらず，キャッシュがダーティになり，一定期間内にディスクに書き出されるため，利用効率算出の際の情報としては用いない．上記式のヒットページ数は，単位時間ごとにリード IO 時に 1 回以上ヒットしたキャッシュページの総数を表し，キャッシュページ数とは，当該 VM 上で動作するゲスト OS が確保しているキャッシュ領域のページ数を表している．

CAMS では，同一物理サーバ上で稼働する VM のキャッシュ利用効率を比較し，キャッシュ利用効率の均一化によりシステム全体のキャッシュ利用効率の向上を図る．具体的には，以下の処理を行う．

(1) CAMS は単位時間ごとに VM ID により識別される各 VM からキャッシュ利用効率を取得する．
(2) キャッシュ利用効率が最も低い VM から N ページ（現状の実装では 64 ページと想定）のメモリを回収する．
(3) キャッシュ利用効率が最も高い VM に N ページ（現状の実装では 64 ページと想定）のメモリを割り当てる．
(4) キャッシュ利用効率が均一になるまで，(1)，(2)，(3) の処理を繰り返す．

本アルゴリズムを用いることで，誤ったメモリ割当て処理を行った場合でも次の処理で修正が可能であり，キャッシュ利用効率の均一化を図ることができる．

3.3　提案構成

CAMS の提案構成を**図 1** に示す．CAMS は以下の要素から構成される．

CAMS においては，active page monitor が IO を監視してキャッシュ利用効率を算出し，memory scheduler に算出結果を通知する．memory scheduler がメモリ回収/割当てを行う VM を決定し，対象 VM の balloon driver にメモリ割当て量の変更を通知する．balloon driver がバルーニング機構を用いて VM のメモリ割当て量を変更する．

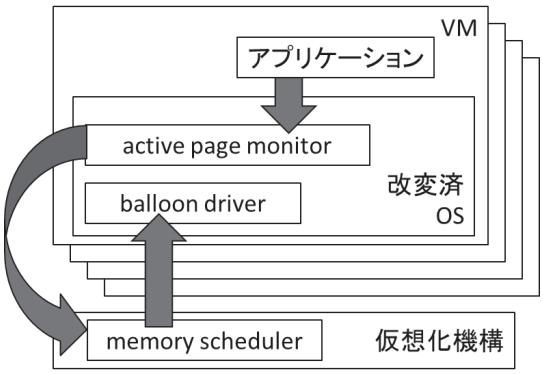

図 1　提案構成
Fig. 1　Proposed architecture.

各要素の詳細を以下に述べる．

- active page monitor
  - リードヒットしたキャッシュページのページアドレスを取得し，単位時間ごとにヒットページ情報を集計する．
  - ゲスト OS が所有するキャッシュページ数を取得する．キャッシュページ数は OS が提供する情報から取得可能と想定する．
  - 単位時間あたりのヒットページ数とゲスト OS が所有するキャッシュページ数を用いて算出したキャッシュ利用効率をゲスト OS とハイパバイザ間の通信用 API を用いてハイパバイザに通知する．
- memory scheduler
  - 各 VM から通知されたキャッシュ利用効率から，VM のメモリ割当て量を決定し，メモリ割当てを実現するバルーンサイズを balloon driver に通知する．
- balloon driver
  - バルーニングを用いてバルーンサイズで指定された量のメモリを VM から回収することで，VM のメモリ割当て量を変更する．

4.　キャッシュアウェアメモリスケジューラの実装

本章では，実装した CAMS の構成について説明する．モジュール構成図を**図 2** に示す．

- ハイパバイザ
  - ハイパバイザに memory scheduler と balloon device emulator を追加実装した．
  - memory scheduler は，TCP/IP 通信ベースのキャッシュ利用効率 API（**表 1** 参照）を利用して active page monitor からキャッシュ利用効率情報を取得する．バルーンサイズ変更要求 API（**表 2** 参照）の balloon_get を用いて，前回指定したバルーンサイズ

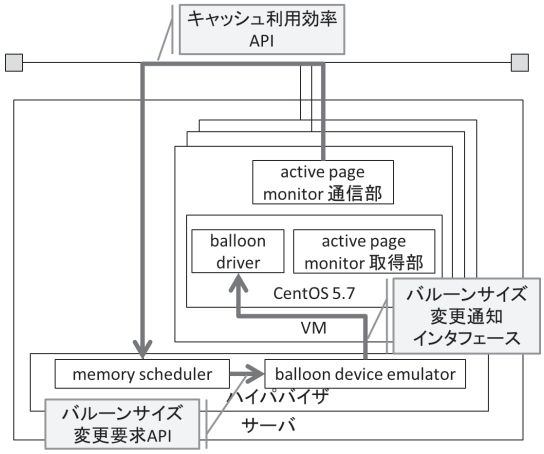

図 2　実装構成
Fig. 2　Implemented architecture.

表 1　キャッシュ利用効率 API
Table 1　Cache usage API.

パラメータ	説明
VM ID	あらかじめ作成した active page monitor 通信部が読み込み可能な構成情報から取得
ヒットページ数	active page monitor 取得部で取得
キャッシュページ数	active page monitor 通信部で取得

表 2　バルーンサイズ変更要求 API
Table 2　Balloon resize requirement API.

パラメータ	説明
balloon_set	
VM ID	memory scheduler で取得
バルーンサイズ	memory scheduler で決定
balloon_get	
VM ID	memory scheduler で取得
バルーンサイズ	balloon device emulator で取得

表 3　バルーンサイズ変更通知インタフェース
Table 3　Balloon resize notify interface.

パラメータ	説明
num_pages	変更後のバルーンサイズ balloon device emulator で取得
actual	現状のバルーンサイズ PCI 構成空間から取得

変更が完了していることを確認する．さらに，3.3 節に示した動作を行い，バルーンサイズ変更要求 API の balloon_set を用いて，バルーンサイズ変更を要求する．

– balloon device emulator はバルーンサイズ要求 API 経由での要求に基づきバルーンサイズ変更通知インタフェース（**表 3** 参照）である balloon device の PCI 構成空間のレジスタの値を参照/更新する．このインタフェースは virtIO [14] の仕様に基づいているため，ゲスト OS の既存の balloon driver をほぼ無改変で流用できる．

- 改変済み OS
  – ゲスト OS は linux-2.6.18-274.el5（CentOS 5.7 [*2]）の Linux に active page monitor，balloon driver を追加した．
  – active page monitor は，取得部（特権モードで動作）と送信部（ユーザモードで動作）からなり，3.3 節に示した動作を行って，キャッシュ利用効率の取得，送信を行う．
  – balloon driver は既存ドライバを流用した．ただし，性能向上のため，ページの回収/割当て単位は 4 K ページ単位ではなく 2 M ページ単位で行うように変更した．

提案構成では active page monitor から memory scheduler へのキャッシュ利用効率の通知には，ゲスト OS とハイパバイザ間の通信用 API を用いると記載した．実装構成では，ゲスト OS やハイパバイザの改変量を抑えるため，キャッシュ利用効率 API に TCP/IP 通信を用いた．TCP/IP 通信には約 0.2 ミリ秒の時間を要する一方で，バルーンサイズ変更処理はキャッシュ上のデータのディスクへの書き込み処理をともない，約 90 ミリ秒の時間を要する．所要時間のオーダが異なるため，TCP/IP 通信の遅延は問題としない．

5. 評価実験

本章では前章で述べた CAMS を用いて，VM で発生するファイル IO のアクセス範囲およびアクセス頻度が異なる場合の CAMS の効果を確認する評価実験および実験結果と考察について述べる．

5.1 評価指標

CAMS はキャッシュの有効活用を目的としている．キャッシュは活用されるほど，ディスク IO を削減する効果がある．そこで，以下の式により得られる値をディスク IO 削減率として，CAMS の効果を評価する値とする．

$$\text{ディスク IO 削減率} = 1 - \frac{\text{ディスクに要求される IO バイト数}}{\text{アプリケーションから発行された IO バイト数}}$$

ディスク IO 削減率は VM ごとに取得可能な数値であり，値が大きいほど，CAMS は効果的であるといえる．

5.2 実験環境

表 4 に示す構成の物理サーバ上に実験環境を構築する．

[*2] CentOS の名称およびそのロゴは，CentOS ltd の商標または登録商標です．

表 4　実験環境
Table 4　Server configuration.

CPU	Intel Xeon *3 E31275 @ 3.40 GHz　2 コア
Memory	8 GB
NIC	Intel Gigabit
Kernel	linux-2.6.18-274.el5（CentOS 5.7）

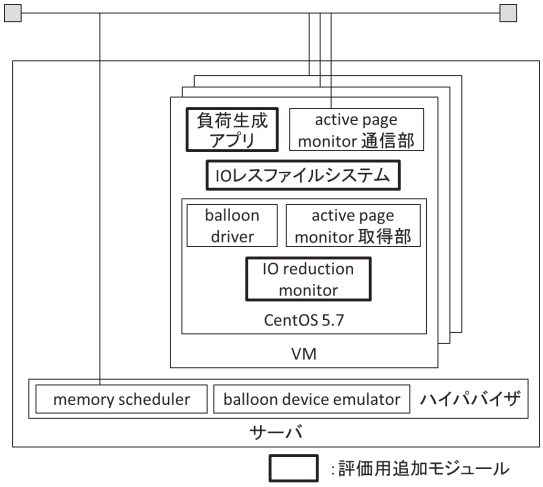

図 3　実験環境 1
Fig. 3　Evaluated architecture 1.

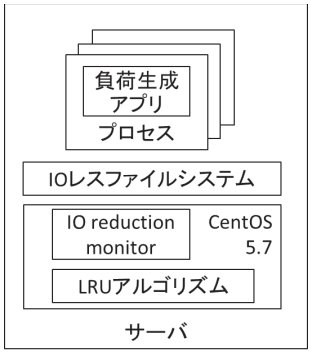

図 4　実験環境 2
Fig. 4　Evaluated architecture 2.

評価実験では，CAMS と SMS を比較する．CAMS の評価のために**図 3** に示す実験環境 1 を，SMS の評価のために**図 4** に示す実験環境 2 を構築する．

実験環境 1 では，CAMS の評価のために，4 章で示したモジュールに加えて，以下のモジュールを追加で実装した．

- 負荷生成アプリケーション
  - 複数コンテンツ（ファイル）へのアクセス負荷をエミュレートするアプリケーション．
  - コンテンツ数，コンテンツサイズ，アクセス頻度，ホットタイトル比率，ホットアクセス比率を指定できる．
  - コンテンツへのアクセスはべき分布に従うと仮定した．ホットタイトル比率で指定された一部のコンテンツに，ホットアクセス比率で指定された比率のアクセスが集中する状態を再現し，アクセス局所性を変動できるようにした．
  - 指定したアクセス頻度の間隔でファイル IO 負荷を発生させることで，ファイル IO 負荷の変動も可能にした．
- IO レスファイルシステム
  - FUSE 上に実装した独自ファイルシステムであり，POSIX API を提供する．
  - 名前空間を管理して，アプリケーションからの IO を受け取るが，ディスクへの IO は発行しない．OS 標準のファイルシステムと同様に VFS 層においてキャッシュを使用する．本ファイルシステムにより，ディスクに IO 負荷をいっさいかけずに，当該ゲスト OS のキャッシュヒット/ミスの挙動を正確にエミュレートできる．
- IO reduction monitor
  - 前節で述べた評価指標であるディスク IO 削減率を求める．
  - VFS 層に要求されるリード IO のバイト数を取得する，これを前節のアプリケーションから発行された IO バイト数とする．
  - VFS 層から発行されるリード IO のバイト数を取得する，これを前節のディスクに要求される IO バイト数とする．
  - 各 IO バイト数を用いて算出したディスク IO 削減率は proc ファイルシステム [15] 経由でユーザ空間に通知する．

実験環境 2 では，物理環境上に CentOS 5.7 を稼働させ，CentOS 5.7 上で複数のプロセスを生成し，負荷生成アプリケーションを稼働させる．このような構成をとることで，2.1.1 項に示した SMS の動作をエミュレートした．

なお，評価指標であるディスク IO 削減率を取得するために CentOS 5.7 を改変し IO reduction monitor を実装している．

5.3　評価方法

CAMS は，キャッシュ利用効率に応じたメモリ割当て量の調整機能を提供する．

本機能の効果を確認するために，キャッシュ利用効率の異なる 2VM を同時稼働させ，各 VM のディスク IO 削減率の推移を測定する実験を行う．2VM は 2.1.2 項で示したキャッシュヒット率は高いが低負荷の VM（VM1）とキャッシュヒット率は低いが高負荷の VM（VM2）からなるとする．ファイルサーバや Web サーバはディスクアクセスの局所性が高い場合が多い [16] ことから，VM1 は Web サーバとして使用されている VM を想定する．VM2

*3　Xeon は，Intel Corporation またはその子会社の米国およびその他の国における商標または登録商標です．

にはディスクアクセスの局所性が最も低いケースであるランダムアクセスを実施する VM を想定する．なお，オンライントランザクションやクラウドストレージに局所性の低いディスクアクセスを行う傾向があるといわれている [17]．

各 VM で稼働する負荷生成アプリケーションの各パラメータには以下に示す値を設定する．

VM1

ディスクアクセスの局所性の高いファイル IO を実施する．

一般に，Web サーバのコンテンツごとのアクセス数分布はべき分布に従う [18] といわれている．そこで，アクセス範囲の 20%の領域に 80%のアクセスが集中する（べき分布）と仮定したファイル IO を実施する．

- コンテンツ数　4,096，40,960，81,920，163,840，245,760，327,680
- コンテンツサイズ　256 KB
- アクセス頻度　10 ミリ秒ごと，20 ミリ秒ごと，40 ミリ秒ごと，60 ミリ秒ごと，80 ミリ秒ごと
- ホットタイトル比率　20%
- ホットアクセス比率　80%

VM2

ディスクアクセスの局所性の低いファイル IO を実施する．

- コンテンツ数　4,194,304
- コンテンツサイズ　256 KB
- アクセス頻度　10 ミリ秒ごと
- ホットタイトル比率　20%
- ホットアクセス比率　20%

VM1 と VM2 が混在する環境を実験環境 1 と実験環境 2 のそれぞれで再現し，それぞれでのディスク IO 削減率を比較する．VM1 のコンテンツ数およびアクセス頻度を変動させることで，CAMS が SMS と比較して効果のあるアクセス範囲およびアクセス頻度の評価を試みる．

5.4　実験結果

前節で示した実験の測定結果を以下に示す．なお，**図 5**，**図 6**，**図 7**，**図 8** の Y 軸はディスク IO 削減率を示す．アプリケーションから発行された IO バイト数よりもディスクに要求される IO バイト数が大きい場合ディスク IO 削減率は負数となる．キャッシュミスが発生すると，ディスク IO を実施する際に，先読みと呼ばれる要求されている範囲よりも先の部分を含めた IO 要求が実施される．これにより，ディスク IO 削減率は負数となることがある．また，VM1 のアクセス範囲はコンテンツ数およびコンテンツサイズから算出される．たとえば，コンテンツ数が 327,680 でコンテンツサイズが 256 KB のとき，アクセス範囲は 80 GB となる．

図 5，図 6 から，CAMS 適用時に VM2/VM1 のアクセス頻度比が大きくなっても，VM1 および VM2 のディスク IO 削減率は変動しないことが分かる．たとえばアクセス範囲が 80 GB の場合，VM2/VM1 のアクセス頻度比が大きくなっても，CAMS 適用時の VM1 のディスク IO 削減率は最大で 4.6%しか低減しない．

また，VM1 のアクセス範囲が広くなるにつれ，VM1 のディスク IO 削減率は低減していることが分かる．たとえば VM2/VM1 のアクセス頻度比が 8 の場合，アクセス範囲

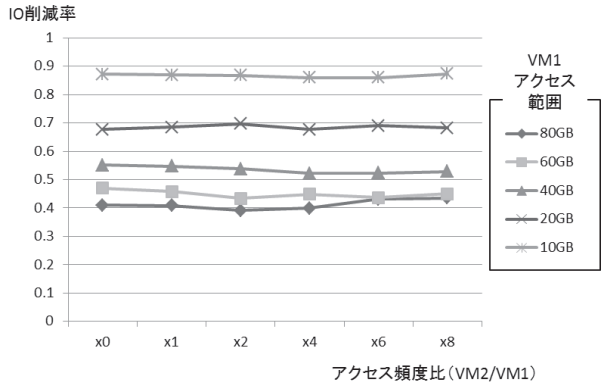

図 5　VM1 のディスク IO 削減率（CAMS 適用時）
Fig. 5　IO reduction of VM1 (CAMS).

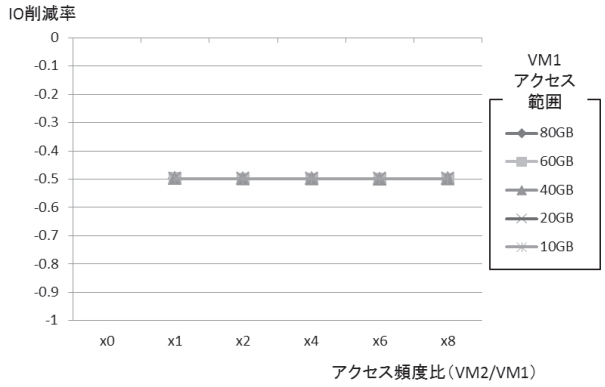

図 6　VM2 のディスク IO 削減率（CAMS 適用時）
Fig. 6　IO reduction of VM2 (CAMS).

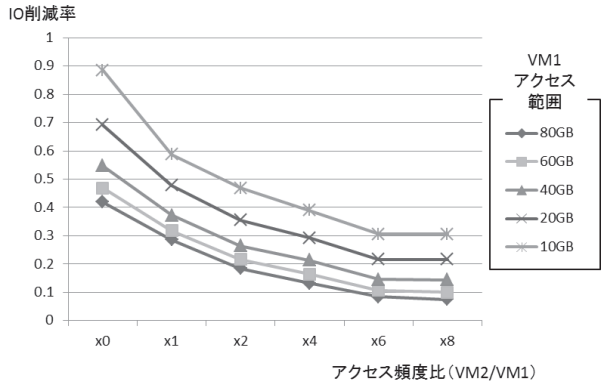

図 7　VM1 のディスク IO 削減率（SMS 適用時）
Fig. 7　IO reduction of VM1 (SMS).

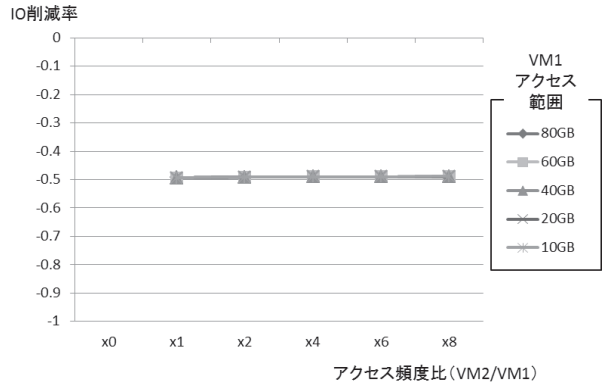

図 8　VM2 のディスク IO 削減率（SMS 適用時）
Fig. 8　IO reduction of VM2 (SMS).

が 10 GB から 80 GB に向上すると，CAMS 適用時の VM1 のディスク IO 削減率は 0.9 から 0.4 に低減する．

図 7，図 8 から，SMS 適用時に VM2/VM1 のアクセス頻度比が大きくなると，VM2 のディスク IO 削減率は変動しないが，VM1 のディスク IO 削減率は減少することが分かる．たとえばアクセス範囲が 80 GB の場合，VM2/VM1 のアクセス頻度比が大きくなると SMS 適用時の VM1 のディスク IO 削減率は最大で 81.6%低減する．

また，VM1 のアクセス範囲が広くなるにつれ，VM1 のディスク IO 削減率は低減していることが分かる．たとえば VM2/VM1 のアクセス頻度比が 8 の場合，アクセス範囲が 10 GB から 80 GB に向上すると，SMS 適用時の VM1 のディスク IO 削減率も 0.3 から 0.1 に低減する．

SMS に対する CAMS の効果を明らかにするため，IO 削減率向上率の変動を以下の式で算出した．

$$\text{IO 削減率向上率} = \frac{\text{CAMS 適用時のディスク IO 削減率}}{\text{SMS 適用時のディスク IO 削減率}}$$

たとえば，VM2/VM1 のアクセス頻度比が 1 のとき，CAMS 適用時の VM1 のディスク IO 削減率は 86.9%であり，SMS 適用時の VM1 のディスク IO 削減率は 58.8%である．このときの VM1 の IO 削減率向上率を 86.9/58.8 = 1.5 と表す．**図 9** に VM1 の IO 削減率向上率の推移を示す．

図 9 から以下の 2 点が分かる．

(1) VM2/VM1 のアクセス頻度比が大きくなるほど，IO 削減率向上率が向上し，CAMS の効果が強く現れる．たとえばアクセス範囲が 80 GB の場合，VM2/VM1 のアクセス頻度比が 1 から 8 に増加すると，IO 削減率向上率は 1.0 倍から 5.8 倍に向上する．

(2) VM1 のアクセス範囲が広くなるほど，IO 削減率向上率が増加し，CAMS の効果が強く現れる．これは，IO 削減率低減の度合いが CAMS 適用時の方が小さいためである．たとえば VM2/VM1 のアクセス頻度比が 8 の場合，アクセス範囲が 10 GB から 80 GB に向上すると，IO 削減率向上率は 2.9 倍から 5.8 倍に増加する．

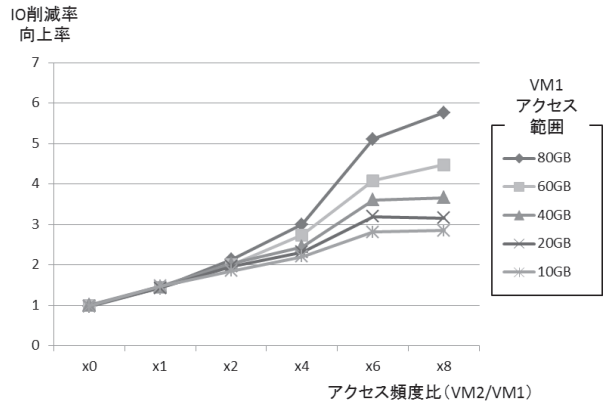

図 9　IO 削減率向上率の推移
Fig. 9　Progress rate of IO reduction.

表 5　Sysbench 実行結果
Table 5　Result of Sysbench.

	Sysbench スコア
CAMS 非適用時	10.007
CAMS 適用時	10.017

5.5　オーバヘッド

CAMS のオーバヘッドを確認する．

CAMS 適用時と非適用時において Sysbench [19] の CPU 性能測定を行うことで，CPU 高負荷時の CAMS の影響を確認する．

実験では，以下のファイル IO を実施する VM 上で Sysbench を用いた．

- コンテンツ数　4,096
- コンテンツサイズ　256 KB
- アクセス頻度　10 ミリ秒ごと
- ホットタイトル比率　20%
- ホットアクセス比率　80%

実行結果を**表 5** に示す．なお，Sysbench スコアは，指定数以下の素数の数え上げに要した時間を表している．値が低いほど CPU の処理能力は高いと判断できる．

実験の結果，CAMS の適用により Sysbench スコアは 0.1%程度悪化することが分かった．CAMS のオーバヘッドは微細といえる．

5.6　考察

5.4 節で述べた (1)，(2) の測定結果が得られた理由の考察を行う．

5.4 節の (1) の測定結果が得られた理由は以下のとおりであったと考えられる．

VM2 はディスクアクセスの局所性が低いため，ほとんどのアクセスはキャッシュミスし，キャッシュ利用効率はつねにほぼ 0 である．一方，VM1 はアクセスの局所性が高いため，アクセスはキャッシュヒットしやすく，キャッ

シュ利用効率は 0 よりも大きい値をとる．

以上より，CAMS では VM2 から回収したメモリを VM1 に割り当てる．VM1 のメモリ割当て量が増え，VM2 のメモリ割当て量が減っても，各 VM のキャッシュ利用効率の関係は上記のまま変わらない．そのため，VM1 にはシステムの大部分のメモリが割り当てられる．この動作は VM2 のアクセス頻度の高低の影響は受けない．そのため，CAMS 適用時には，VM2 のアクセス頻度の高低は VM1 のディスク IO 削減率に大きな影響をおよぼさないという結果となった．

それに対し，SMS を使用した場合は，各 VM が使用するキャッシュ領域の量はほぼ各 VM のアクセス頻度の比率になると考えられるため，VM1 に割り当てられるメモリはおおよそ以下の容量になると考えられる．

$$\text{VM1 のメモリ容量} = \frac{\text{VM に割当て可能なメモリ量}}{\text{VM1 のアクセス頻度} + \text{VM2 のアクセス頻度}} \times \text{VM1 のアクセス頻度}$$

すなわち，VM2 のアクセス頻度が高いほど，VM1 のメモリ容量は減り，ディスク IO 削減率は低下する．そのため，SMS 適用時には，VM2 のアクセス頻度が高いほど，VM1 のディスク IO 削減率は減少するという結果となった．

以上より，VM2 のアクセス頻度が高くなるほど IO 削減率向上率は高くなる．

5.4 節の (2) の測定結果が得られた理由は以下であったと考えられる．

VM1 のアクセス範囲が広がると，VM1 のアクセス局所性が低くなるため，アクセス範囲が広がるほど VM1 のディスク IO 削減率は低減する．

本実験において，VM のキャッシュ容量は VM1 のアクセス範囲の変動の影響を受けない．

アクセス範囲が広がるとタイトル数が増えるため，各タイトルへのアクセス数は減少する．高頻度でアクセスを受けていたタイトルほど，アクセス範囲が広がったことによるアクセス数減の度合いは大きい．キャッシュ容量が大きいと，アクセス数がさほど多くないタイトルもキャッシュに格納しているため，キャッシュ容量が小さいときと比較してアクセス範囲が広がったことによるアクセス数減の影響を受けにくい．

そのため，キャッシュ容量が小さい場合（SMS 適用時）は，アクセス範囲が広くなるにつれ，上記アクセス比率が急激に低減していくのに対して，キャッシュ容量が大きい場合（CAMS 適用時）はこの低減率が小さくなる．この結果，VM1 のアクセス範囲が大きくなるほど，IO 削減向上率は高くなった．

クラウド環境においては，VM 構築時に使用可能 IO 帯域幅を制限する場合がある．CAMS は，IO 帯域制限時の IOPS 向上効果が期待できる．

たとえば，VM1 でアクセス範囲が 80 GB のべき分布アクセスを行い，ランダムアクセスを行う VM2 が 8 台稼働している場合の IOPS について検討する．なお，VM2/VM1 のアクセス頻度比 8 は，同一サーバ上に VM1 と同じ IO 負荷の VM2 が 8 台稼働している状況と等しい．

各 VM のディスク IO 削減率を以下に示す．

- VM1（CAMS 適用時）　0.44
- VM1（SMS 適用時）　0.08
- VM2　−0.49

IO 帯域上限時の IOPS を 100 とした場合の各 VM の IOPS を以下に示す．

- VM1（CAMS 適用時）　179
- VM1（SMS 適用時）　109
- VM2　67

以上より，システム全体の IOPS を以下に示す．

- CAMS 適用時　715
- SMS 適用時　645

例にあげた構成においては，CAMS 適用時は SMS 適用時と比較して，システム全体の IOPS が約 1.1 倍となることが分かる．

本結果は，VM1 と VM2 のディスクアクセスの局所性の差が大きくメモリスケジューリングによるディスク IO 削減効果の出やすい環境を前提としている．VM1 と VM2 のディスクアクセスの局所性の差が縮まるにつれ，ディスク IO 削減効果は出にくくなることに注意する必要がある．ただし，CAMS のオーバヘッドは微細なため，CAMS が性能に大きく悪影響を与えることはないと考えられる．

6. 関連研究

本章では関連研究として，キャッシュの利用状況に応じたメモリ割当てによりディスクキャッシュポリューション問題の直接的解決が期待できるメモリスケジューラと，システム全体のキャッシュヒット率を向上させることでディスクキャッシュポリューション問題の間接的解決が期待できるメモリスケジューラついて述べる．

6.1 ディスクキャッシュポリューション問題の直接的解決が期待できるメモリスケジューラ

キャッシュの利用状況に応じたメモリ割当て方式として Meng ら [20] の提案するメモリスケジューラが知られている．

Meng らの提案メモリスケジューラは，アプリケーションごとにキャッシュ領域を分割し，アプリケーションのディスク IO パタンに応じて割当てキャッシュ量の調整を行う．その際，各領域に割り当てるキャッシュ量の増減にともない変動するキャッシュミス量を MU という値で表し，割当て量調整の尺度として用いる．これにより，キャッシュミ

ス量を最小化するアプリケーションへのキャッシュ割当てを実施する．

Meng らの提案スケジューラをハイパバイザで用いると本研究で提案するメモリスケジューラと類似した挙動を示し，ディスクキャッシュポリューション問題は解決可能と考えられる．ただし，本研究で提案するメモリスケジューラが，影響範囲の小さい箇所を改変したゲスト OS を用いる一方で，Meng らの提案スケジューラは，既存の IO 処理を変更する独自のドライバが必要である．改変箇所の影響範囲は大きく，信頼性保証の難易度は高いと考えられる．

以上より，本研究で提案するメモリスケジューラも Meng らの方式も，適用することでディスクキャッシュポリューション問題の解決が期待できる．ただし Meng らの方式と比較して提案するメモリスケジューラには信頼性保証が容易であるという利点があるといえる．

6.2 ディスクキャッシュポリューション問題の間接的解決が期待できるメモリスケジューラ

仮想化環境においてシステム全体のキャッシュヒット率の向上を図るメモリスケジューラは多数提案されている．これらのメモリスケジューラを，（A）重複領域を排除することで使用可能なキャッシュ領域を拡張するタイプと，（B）VM から取得した情報に基づいてキャッシュヒット率の向上が期待できる VM にメモリを割り当てるタイプに分類する．また，タイプ（B）のスケジューラを，（B-1）ゲスト OS を改変して情報を取得するタイプと，（B-2）ゲスト OS を改変せずに情報を取得するタイプに分類する．

タイプ（A）のメモリスケジューラとしては Arcangeli ら [3]，Milos ら [4]，Jones [5] によるものが知られている．

Arcangeli らの提案メモリスケジューラは，メモリ内容を定期的に検索し，同じ内容が格納されていた場合には，物理メモリを共有することで，消費する物理メモリ量を削減する．この物理メモリにはキャッシュ領域も含まれるので，消費するキャッシュ領域サイズの削減も実現できる．なお，Arcangeli らの提案メモリスケジューラには検索時間が長大であるという問題がある．本問題は Miller ら [6] の提案する手法により，検索時に用いるヒント量を制限することで，低減が可能といわれている．

Milos らの提案メモリスケジューラは，ゲスト OS が発行するディスク IO を監視して，ページ番号と当該ページに格納されているデータの転送元となるディスクブロック番号の対応関係を追跡する．そして，同一データを保持している可能性のあるページ番号の組を検出し，内容が一致するのであれば重複排除を実施する．

Jones の提案メモリスケジューラは，ハイパバイザがゲスト OS によるディスク IO を監視することで，ゲスト OS 上のキャッシュからのデータ削除を検出する．ゲスト OS において削除されたデータのみハイパバイザの所有する 2 次キャッシュに格納することで，ゲスト OS とハイパバイザ間のキャッシュ重複を排除している．2 次キャッシュでは VM のキャッシュ情報が共有されるため，複数 VM が同一領域にアクセスする場合，当該領域の重複を排除することができる．

タイプ（A）のメモリスケジューラを用いることで，重複領域が排除され，VM の使用可能なキャッシュ領域が拡張すため，システム全体のキャッシュ利用効率の向上が期待できる．しかし，タイプ（A）ではキャッシュ利用状況に応じたメモリスケジューリングは行わないため，ディスクキャッシュポリューション問題は解決できない．ただし，CAMS と組み合わせてさらなる VM 性能の改善を図ることは可能である．

タイプ（B-1）のメモリスケジューラとしては，Magenheimer ら [7]，Hwang ら [8] によるものが知られている．これらのスケジューラは，ハイパバイザが各 VM に追加のキャッシュ領域を提供することでキャッシュ利用効率の向上を図る．

Magenheimer らの提案メモリスケジューラは，ハイパバイザは VM に未割当てのメモリ領域をプールとし，VM 上のキャッシュから削除されたデータの格納先としてプールを使用する．VM はキャッシュ上のデータを削除する際に，ハイパバイザに当該データ領域の範囲を通知し，ハイパバイザは当該データをプールに格納する．

Hwang らの提案メモリスケジューラは，VM に未割当てのメモリ領域を追加キャッシュとして使用する．提案スケジューラは VM 上で発行された IO をユーザ空間やカーネル空間で取得し，ハイパバイザに通知する．ハイパバイザはキャッシュ上にデータがあるときはキャッシュからデータを取得する．ハイパバイザでキャッシュミスした場合は，ゲスト OS に戻り従来どおりの IO を実施する．なお，この際，取得したデータをハイパバイザのキャッシュに格納する．

タイプ（B-1）のメモリスケジューラを用いることで，未使用のメモリ領域を VM のキャッシュとして利用することが可能となり，システム全体のキャッシュ利用効率の向上が期待できる．また，方式 [7], [8] では複数 VM 間のキャッシュ利用状況を比較していないためディスクキャッシュポリューション問題を解決することはできないが，キャッシュ利用状況を監視し比較する機構を追加することで，問題への対応は可能となると考えられる．ただし，前述のとおりこれらの手法を用いた場合，OS 改変による影響範囲は大きくなり，信頼性保証の難易度は高くなる．

タイプ（B-2）のメモリスケジューラとしては，Waldspurger [13]，Zhao ら [21]，Chen ら [22] によるものが知られている．

Waldspurger の提案メモリスケジューラは，意図的に割当て状態の物理ページの一部を非割当て状態にし，ページ

例外発生頻度を測定することで，ゲスト OS のメモリアクセス量を推測する．そして，その量に応じて各ゲスト OS に割り当てるメモリ量を動的に変更する．

Zhao らの提案メモリスケジューラは，ゲスト OS への割当て物理ページの特権レベルを意図的に変更し，ページごとのアクセス保護例外頻度を測定する．そして，アクセス頻度の高いページに優先して物理メモリを割り当てている．

Chen らの提案スケジューラは，スワップの発生を抑制することで，1 章で述べたディスクボトルネックの軽減を図るメモリスケジューラである．これによりキャッシュ領域を持たない VM の発生を回避できる．

Chen らの提案メモリスケジューラは，ゲスト OS でのスワップの発生を検知することで，VM へのメモリ割当て量不足を検出し，当該 VM への物理ページ割当てを優先して行う．なお，Amit ら [23] の提案する VSWAPPER は，VM のメモリ空間をハイパバイザ上のファイルにマッピングすることで，ページのラベリングを非 anonymous 化したり，マッピングの解消でページアウトを模擬したりする機構であり，スワップの発生を抑制したメモリ割当てが実現できる．VSWAPPER を用いることで，より効率的にスワップの発生を抑制できる．

タイプ（B-2）のメモリスケジューラを用いることで，未使用または使用頻度の低いメモリ領域を使用頻度の高いメモリ領域として再割当て可能となり，メモリ利用効率が向上し，システム全体のキャッシュ利用効率の向上が期待できる．しかし，タイプ（B-2）では，ゲスト OS として無改変の汎用 OS を用いるため，汎用 OS が提供しない情報であるキャッシュ利用効率に応じたメモリスケジューリングを行うことは困難である．CAMS と組み合わせてさらなる VM 性能の改善を図ることは可能だが，現状では単独でディスクキャッシュポリューション問題を解決する手法は提案されていない．

7. おわりに

本論文では，クラウド環境で発生するディスクキャッシュポリューション問題を解決するキャッシュアウェアメモリスケジューラ（CAMS）の提案および実装，評価を行った．

CAMS は，VM 上で稼働するゲスト OS に改変を加えることで各ゲスト OS のキャッシュ利用効率を監視する．そして，キャッシュヒットが発生したキャッシュ領域の全領域に対する比率を示すキャッシュ利用効率が均一となるよう，各 VM へのメモリ割当て量を調節する．これにより，キャッシュを活用しない高負荷 VM に割り当てられるメモリ量を制限し，キャッシュを活用する低負荷 VM に十分なメモリを割り当て，ディスクキャッシュポリューション問題を解決できる．

CAMS により SMS と比べてシステム全体のディスク IO 量がどの程度削減できるかを評価した．アクセス局所性の高い IO インテンシブな VM（VM1）とアクセス局所性の低い IO インテンシブな VM（VM2）とが稼働している環境を用いた．評価の結果，VM2/VM1 のアクセス頻度比が 1 から 8 まで高くなるにつれ IO 削減率向上率を 1.5 倍から 5.7 倍に高められることが分かった．また，VM1 のアクセス範囲が 10 GB から 80 GB に広がるにつれ，IO 削減率向上率を 2.9 倍から 5.8 倍に高められることが分かった．

また，IO 帯域に上限がある場合 CAMS により IOPS 向上効果が期待できることが分かった．VM1 と VM2 のディスクアクセスの局所性の差が大きいほど，効果は大きいと考えられる．

参考文献

[1] 上原　宏：これから始める「次世代ストレージ」，日経コンピュータ，pp.116–119, 日経 BP 社 (2013).

[2] 矢口竜太郎：動き出した基幹系クラウド, 日経SYSTEMS, pp.32–53, 日経 BP 社 (2013).

[3] Arcangeli, A. et al.: Increasing memory density by using ksm, *Linux Symposium* (2009).

[4] Milos, G. et al.: Satori: Enlightened page sharing, *Proc. 2009 USENIX Annual Technical Conference* (2009).

[5] Jones, S.T.: Geiger: monitoring the buffer cache in a virtual machine environment, *ASPLOS XII Proceedings of the 12th international conference on Architectural support for programming languages and operating systems* (2006).

[6] Miller, K. et al.: Xlh: more effective memory deduplication scanners through cross-layer hints, *USENIX ATC* (2013).

[7] Magenheimer, D. et al.: Transcendent memory and linux, *Linux Symposium* (2009).

[8] Hwang, J. et al.: Mortar: filling the gaps in data center memory, *Proc. 10th international conference on Virtual execution environments* (2014).

[9] 千葉一彦：第 7 回クラウドランキング，日経コンピュータ，pp.56–69, 日経 BP 社 (2013).

[10] 南波幸雄：インフラ知識大全，IT アーキテクト，第 10 巻，pp.30–37, IT アーキテクト編集部 (2007).

[11] 森側真一：エンタープライズ Linux 最前線，日経 Linux, p.65, 日経 BP 社 (2008).

[12] 佐々木盛朗：連続メディアデータ対応のウェブキャッシングアルゴリズム，分散システム/インターネット運用技術シンポジウム 2003 論文集 (2003).

[13] Waldspurger, C.A.: Memory resource management in vmware esx server, *Proc. 5th Symposium on Operating Systems Design and Implementation* (2002).

[14] Russell, R.: virtio: towards a de-facto standard for virtual i/o devices, *ACM SIGOPS Operating Systems Review — Research and developments in the Linux kernel* (2008).

[15] 野原　透：触って学ぶ Linux カーネル, 日経 Linux, pp.73–79, 日経 BP 社 (2008).

[16] 松本俊子：文書処理装置，及びファイルサーバ管理支援方法，並びにファイルサーバ管理支援プログラム (2012). Japan Patent, WO2012114808 A1.

[17] 中村俊介：読み出し性能と書き込み性能を両立させるクラウドストレージ，先進的計算基盤システムシンポジウム論文集 (2011).

[18] 長健二朗：インターネット計測とデータ解析 第5回，IIJ技術研究所，入手先 ⟨http://www.iijlab.net/~kjc/classes/sfc2013s-measurement/sfc2013s-measurement-5.pdf⟩（参照 2014-06-19）．
[19] Kopytov, A.: Sysbench, 2010, sysbench-0.4.10-1.src.rpm.
[20] Meng, X. et al.: A flexible two-layer buffer caching scheme for shared storage cache, *11th IEEE International Conference on High Performance Computing and Communications* (2009).
[21] Zhao, W. et al.: Dynamic memory balancing for virtual machines, *VEE '09 Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments* (2009).
[22] Chen, X. et al.: 2013. USPatent 8359451 B2.
[23] Amit, N. et al.: Vswapper: A memory swapper for virtualized environments, *Proc. 19th international conference on Architectural support for programming languages and operating systems* (2014).

田島 幸恵

1980年生．2003年東京工業大学工学部情報工学科卒業．2005年同大学大学院知能システム科学専攻修士課程修了．同年より現在まで（株）日立製作所横浜研究所（旧システム開発研究所）勤務．情報ストレージシステムの研究開発に従事．

竹内 理　（正会員）

1969年生．1992年東京大学理学部情報科学科卒業．1994年同大学大学院理学系研究科情報科学専攻修士課程修了．同年より現在まで（株）日立製作所横浜研究所（旧システム開発研究所）勤務．2009年東京大学大学院学際情報学府博士課程修了．学際情報学博士．連続メディア処理向きマイクロカーネル，仮想計算機モニタ，ストレージ制御ソフトウェアの研究，特に，QoS保証型資源スケジューリング方式，異種OS共存方式，リアルタイム通信方式，OSデバッグ方式，高信頼システムソフトウェア構築方式の研究に従事．2006年情報処理学会論文賞受賞．

野中 裕介　（正会員）

1975年生．2002年九州大学大学院システム情報科学府博士後期課程修了．同年より現在まで（株）日立製作所横浜研究所（旧システム開発研究所）勤務．主として情報ストレージシステムの研究開発に従事．博士（情報科学）．