Vol.2015-OS-135 No.5
Vol.2015-EMB-39 No.5
2015/11/24

次世代不揮発性メモリを利用するハイパーバイザの試作

広渕 崇宏[1]　高野 了成[1]

概要：次世代不揮発性メモリはリフレッシュ電力が不要である反面、書き込み時の消費電力が DRAM よりも大きいとされる。そこで、STT-MRAM と DRAM を搭載した計算機を想定して、仮想マシンに対して動的にメモリ割り当てを最適化するハイパーバイザを試作した。仮想マシンに対して透過的に、書き込みが頻出するページを DRAM に配置し、消費電力を抑制することを目指す。提案ハイパーバイザは、DRAM および MRAM から仮想マシンのメモリを割り当てる機構、軽量なメモリアクセスのトレース機構、ページマイグレーションを判断するアルゴリズム、DRAM と MRAM 間でページをスワップする機構からなる。Qemu/KVM に対して提案ハイパーバイザを開発している。本稿では Work-in-Progress として、現状のプロトタイプ実装を紹介する。簡易な実験を行った結果、提案機構が書き込みの多いページを DRAM へ動的に再配置できることを確認した。

1. はじめに

次世代の不揮発性メモリに対する関心が高まっている。従来より計算機のメインメモリとして DRAM が広く用いられてきた。しかし DRAM はデータを維持するためにリフレッシュ電力を必要とする。メモリ容量に応じて消費電力が増大するため、計算機に大容量のメモリを搭載すると消費電力が非常に大きくなってしまう。計算機全体の消費電力に占める DRAM の割合が徐々に問題視されつつある。また、微細化の限界に近づきつつあり、その大容量化が鈍化しつつあるという指摘もある。一方、次世代の不揮発性メモリの代表格である STT-MRAM は、その不揮発性ゆえリフレッシュ電力なしでデータを保持できるため、消費電力を低く抑えることができる。将来的に DRAM と同等の読み書き速度を達成しながら、DRAM を上回る容量を安価に提供できるとされる。

しかし、STT-MRAM は将来においても書き込み消費電力が DRAM よりも 2 桁以上大きいとされる [1]。消費電力をできる限り抑制しながら大容量のメモリを実現するためには、DRAM と MRAM の両方を計算機に搭載し、両者を適切に使い分ける方式 (ハイブリッドメモリ) に将来性があると考えている。すでに計算機の DIMM インタフェースに対応する不揮発メモリのメモリモジュールが存在しており、将来 STT-MRAM についても同様の方式で計算機に接続されるものと考えている。不揮発メモリを DRAM 同様に CPU から直接アクセスできるバイトアドレッサブルなメモリとして計算機に接続し、ソフトウェアにおいて DRAM と不揮発性メモリを使い分ける。現状のメモリコントローラをほぼそのまま不揮発メモリに対しても利用できる。

不揮発性メモリによるメモリの大容量化が恩恵をもたらす分野の一つとして、IaaS (Infrastracture as a Service) クラウドのデータセンタが挙げられる。不揮発性メモリを用いて物理計算機に大容量のメモリを搭載できれば、一台の物理計算機に多数の仮想マシンを稼動して集約効率を高めることができる。顧客に対して安価に IaaS サービスを提供することにつながる。しかし、現状のハイパーバイザはハイブリッドメモリに対して十分な機能を備えているとは言い難い。仮想マシンのメモリアクセスに応じて、DRAM と不揮発メモリを使い分ける仕組みが存在しない。ソフトウェアによる DRAM と不揮発メモリの使い分けが必要となるものの、顧客にとっては、自身の仮想マシンに新たな使い分けの仕組みを導入することは大きな負担となる。ハイパーバイザが顧客に不揮発メモリの存在を意識させずに、透過的に DRAM と不揮発メモリを使い分ける仕組みが望ましい。

そこで、我々は次世代不揮発性メモリに対応した仮想マシンハイパーバイザを提案する。ハイパーバイザが仮想マシンの RAM に対する、DRAM と不揮発メモリのマッピングを最適化し、システム全体の消費電力を低減する。仮想マシン上のシステムに新たな仕組みを一切導入する必要が無い。大きなエネルギーを消費してしまう不揮発メモリに対する書き込みを抑制するため、ハイパーバイザは仮想

[1] 国立研究開発法人 産業技術総合研究所 情報技術研究部門
National Institute of Advanced Industrial Science and Technology (AIST)

マシンの RAM において書き込みが頻出するページを判別し、それらのページを DRAM に動的に再配置する。常に変動するアクセス傾向に応じてメモリマッピングを最適化するため、仮想マシンを止めることなくメモリページの実体を DRAM および不揮発メモリ間で変更 (i.e., ページマイグレーション) できる。

さらに、仮想マシンのメモリアクセス傾向を常に把握するため、仮想マシンの性能低下を小さく抑えたメモリアクセストレース機構を新たに開発した。また、メモリアクセス傾向に応じて、メモリマッピングを動的に最適化するアルゴリズムを新たに開発した。将来登場するハイブリッドメモリを想定して提案機構の評価を行うため、ハードウェアが備えるメモリチャネルごとの性能監視機構を利用した新たな性能評価手法も開発した。既存の仮想マシンハイパーバイザ Qemu/KVM を拡張して提案機構を実装している。

本稿では、Work-in-Progress として提案機構の試作及び初期段階の評価について述べる。2 節で想定環境を説明し、3 節で提案機構について述べる。4 節で初期的段階の動作確認として行った実験について記す。5 節で関連研究について述べ、6 節でまとめる。

2. 想定環境

本研究では、将来登場するであろう DRAM と STT-MRAM のハイブリッドメモリシステムを想定する。STT-MRAM は DRAM 同様に DIMM スロットに接続され、CPU は CPU キャッシュを経由してバイトアドレッサブルなアクセスがメモリに対して可能であるとする。STT-MRAM および DRAM それぞれの物理アドレス範囲は、BIOS を通じてハイパーバイザが取得可能であるとする。STT-MRAM は書き込み回数による劣化はほぼ生じ無いためウェアレベリングの必要性はない。STT-MRAM の読み書き速度は DRAM と同等であるとする。読み込みに要するエネルギーも同等であるものの、書き込みに要するエネルギーは STT-MRAM が DRAM よりも大幅に大きい。本稿では、データの不揮発性に着目した仕組みは研究対象としない。

想定する IaaS クラウドでは、従来のように顧客に対して、あらかじめ指定されたメモリサイズの仮想マシンを提供する。顧客の視点ないしゲスト OS からは、仮想マシンのメモリは従来通りすべて DRAM で構成されているように見える。一方で、物理マシンはハイブリッドメモリとして構成されており、提案機構が仮想マシンに対するメモリマッピングを動的に最適化する。

3. 提案機構

将来のハイブリッドメモリシステムを想定した仮想マシンハイパーバイザを提案する。ハイパーバイザが、仮想マシンの RAM に対する、DRAM と STT-MRAM のマッピングを最適化し、システム全体の消費電力を低減する。仮想マシン上のシステムに新たな仕組みを一切導入する必要が無い。

ホスト OS は、その起動時において、そのホスト上で作成する仮想マシンに用いる DRAM および MRAM 物理ページを、あらかじめメモリプールとして確保する。仮想マシンを作成する際には、DRAM および MRAM それぞれのメモリプールから必要な量の物理メモリページを取得し、仮想マシンの RAM として割り当てる。仮想マシンの RAM を 4GB とすれば、例えば、そのうち 4 分の 1 を DRAM から割り当て、残りの 4 分 3 を MRAM から割り当てる。ハイパーバイザは仮想マシンのメモリアクセス傾向を常に観察し、一定期間ごとにページマイグレーションを行う。書き込みが頻出していると判断したページを DRAM に移動すると同時に、DRAM 上にありながらも書き込みが少ないページを MRAM に移動する。このページスワップ動作を一定期間ごとに何度も繰り返すことで、仮想マシン実行中のメモリアクセス傾向の変化に追随しながら、常に書き込みを DRAM 側にできる限り集中できる。

3.1 メモリ割り当て機構

既存のハイパーバイザ Qemu/KVM に対して提案機構の設計を行った。Qemu/KVM の実装を調査した結果、仮想マシンに対するメモリのホットプラグを実現する仕組みや仮想マシンの RAM を Huge Page として割り当てる仕組みのために、`MEMORY_BACKEND` クラスと呼ばれる抽象化が存在することがわかった。`MEMORY_BACKEND` クラスのサブクラスを独自に実装すれば、独自のメモリ割り当て方式を Qemu/KVM に対して実装できる。そこで提案機構においては、新たに `MEMORY_BACKEND_DEVMEM` サブクラスおよび `MEMORY_BACKEND_CONTAINER` サブクラスを追加した。

Qemu/KVM において通常用いられる `MEMORY_BACKEND_RAM` サブクラスは、`malloc()` により仮想マシンの RAM 領域を確保する。一方、新たに実装した `MEMORY_BACKEND_DEVMEM` サブクラスは、`/dev/mem` に対する `mmap()` により、指定した物理アドレス領域を直接仮想マシンの RAM 領域として確保する。仮想マシンを起動する際には、DRAM 用および MRAM 用それぞれ 2 つの `MEMORY_BACKEND_DEVMEM` オブジェクトを作成し、ホスト OS 起動時からあらかじめ確保してあるそれぞれのメモリプールから、空き物理ページを仮想マシンの RAM 領域として割り当てる。

さらに、DRAM および MRAM 用のメモリオブジェクトを統合して、擬似的に 1 つのメモリオブジェクトとして Qemu/KVM の他のコンポーネントに対して見せる仕組みとして、`MEMORY_BACKEND_CONTAINER` サブクラスも新たに実装した。

図 1 に仮想マシンの起動例を示す。仮想マシンの

```
1  qemu-system-x86_64 -enable-kvm -hda image.qcow2 -m size=16G \
2  -object memory-backend-container,id=dmcon0,size=16G \
3  -object memory-backend-devmem,id=dram0,size=8G,phy-offset=0x1100000000,offset=0,parent=dmcon0 \
4  -object memory-backend-devmem,id=mram0,size=8G,phy-offset=0x3080000000,offset=0x200000000,parent=dmcon0 \
5  -numa node,nodeid=0,memdev=dmcon0
```

図 1 提案機構を用いた仮想マシンの実行例

RAM は 16GB であり、そのうち DRAM プールから 8GB、MRAM プールからも 8GB 割り当てる。2 行目において MEMORY_BACKEND_CONTAINER オブジェクトを作成し、3 行目および 4 行目で作成する MEMORY_BACKEND_DEVMEM オブジェクトを統合している。3 行目において、DRAM プールにおける未使用領域の開始物理アドレス 0x1100000000 を指定して 8GB のメモリを確保し、4 行目において、同様に MRAM プールにおける未使用領域の開始アドレス 0x3080000000 を指定して 8GB のメモリを確保している。offset パラメタを指定することにより、DRAM プールから割り当てたメモリの後ろに、MRAM プールから割り当てたメモリを続けている。5 行目において、作成した MEMORY_BACKEND_CONTAINER オブジェクトを、仮想マシンに対して NUMA ノード 0 のメモリとして接続している。

3.2 メモリアクセストレース機構

仮想マシンのメモリアクセス傾向を常に把握するため、仮想マシンの性能低下を小さく抑えたメモリアクセストレース機構を新たに開発した。仮想マシンがメモリに書き込みを行なったゲスト物理ページを一秒毎に把握する。一秒毎に、書き込みが発生したゲスト物理ページのページ番号を、後述するページマイグレーションアルゴリズムに通知する。先行発表 [2] において説明したため、本稿では概要のみを記す。

Intel 製の CPU は仮想マシンの実行を支援する仕組みとして、Extended Page Table (EPT) とよばれるシャドウページ機構を備える。EPT の一つのエントリは、ゲスト物理アドレスがどのホスト物理アドレスに対応するのか、という情報を表す。ハイパーバイザは EPT エントリを作成し、CPU はその EPT を参照しながら仮想マシンを実行する。Intel の Haswell 世代以降の CPU は、EPT エントリに A/D ビットフィールドを有している。CPU が EPT エントリを使用するたびに、すなわち仮想マシンがそのゲスト物理ページにアクセスするたびに、CPU はその A ビットを立てる。同様に CPU がそのゲスト物理ページに書き込みを行なった場合は、その EPT エントリの D ビットを立てる。

提案機構では、ハイパーバイザが作成した EPT を一秒毎に走査して、D ビットが立っている EPT エントリのゲスト物理アドレスを記録し、D ビットをクリアするという動作を繰り返す。ゲスト物理ページへの書き込み時に対応する EPT エントリの D ビットを立てる操作は、ハードウェアレベルで実装されており、観察した範囲では性能低下は発生しない。また提案機構が EPT エントリを毎秒走査する動作も、仮想マシンを動作させたまま、ハイパーバイザ上で仮想マシンとは別のスレッドで実行するため、仮想マシンに対する影響は極めて軽微である。

3.3 ページマイグレーション機構

仮想マシンの RAM を構成する DRAM および MRAM 領域間で、ゲスト物理ページに対応するホスト物理ページを交換する機構を開発した。MRAM 上に存在するにもかかわらず書き込みが多いゲスト物理ページを、DRAM 上に存在するにもかかわらず書き込みが少ないページと交換する操作において用いる。例えば、ゲスト物理ページ 100 がホスト物理ページ 1000 に、ゲスト物理ページ 200 がホスト物理ページ 2000 にマッピングされている際に、両者のマッピングを入れ替えて、ゲスト物理ページ 100 をホスト物理ページ 2000 に、ゲスト物理ページ 200 をホスト物理ページ 1000 にマッピングし直す。この時、仮想マシンは動作中であるため、ページ上のデータも入れ替える必要がある。ホスト物理ページ 1000 の内容を一時バッファに退避し、ホスト物理ページ 2000 の内容をホスト物理ページ 1000 にコピーし、一時バッファに退避したホスト物理ページ 1000 の内容をホスト物理ページ 2000 にコピーする。

先ほど説明した EPT は、ゲスト物理ページがどのホスト物理ページに対応するのか、という情報を表す。そこで、以上の仕組みを実現するため、交換対象の EPT エントリにおいてホスト物理ページを互いに入れ替える仕組みを実装した。仮想マシンを極めて短時間一時的に停止し、その間に 2 つの EPT エントリの内容を書き換える。さらに、ユーザ空間の Qemu およびカーネル空間の KVM において、入れ替え内容を記録するテーブルも実装した。Qemu ないし KVM のコードがホスト物理ページを参照する際には、このテーブルを介した上で本物のホスト物理ページを参照するように、デバイスのエミュレーション等を実装する部分において処理を追加している。同時に、一時バッファを用いてデータの交換も行う処理も作成した。

3.4 ページマイグレーションのアルゴリズム

メモリアクセストレース機構で得られた情報をもとに、どのゲスト物理ページを DRAM 上ないし MRAM 上に配置すべきか判断し、必要なページマイグレーションを決定するアルゴリズムを開発した。

3.4.1 予備調査

最初に予備的な調査として、仮想マシン上でさまざまなワークロードを動作させている最中に、どのようにゲスト物理ページへの書き込みが生じるのか調べた。4GB の RAM を持つ仮想マシンを作成し、先述のメモリアクセストレース機構を用いて、書き込みが生じたゲスト物理ページのページ番号を 1 秒毎に記録した。仮想マシン上では、様々な種類のワークロードを再現するベンチマークプログラムである SPEC CPU 2006 を動かした。

図 2 にアクセス済みおよび書き込み済みページ数の時間変動を示す。ワークロードによって書き込みが生じるゲスト物理ページの数が異なるほか、一つのワークロードの実行期間内においても大きく変動することがわかった。例えば、mcf においては毎秒 40 万ページを超える範囲に書き込みが生じる時もあれば、毎秒 5 万ページ程度のみ書き込まれる時もある。常に同一のゲスト物理ページを DRAM に配置するのではなく、動的に DRAM に配置するゲスト物理ページを変更する必要があると考えられる。

次にゲスト物理ページに対する書き込みの時間的な局所性を調べた。あるゲスト物理ページに対する書き込みが検知されたのち、再び同じページに対して書き込みが検知されるまでの時間を求めた。**図 3** は、X 軸に書き込み間隔をとり、Y 軸にその発生回数を示す。いずれのワークロードにおいても、一度書き込みが検知されたページは、ごく短時間のうちに再び書き込みが検知される傾向があることがわかった。ページマイグレーションのアルゴリズムは、基本的には、最近書き込みが生じたページを優先的に DRAM に配置すれば、DRAM における書き込みの発生確率を高くできると考えられる。

同様に、ゲスト物理ページに対する書き込みの空間的な局所性を調べた。各ワークロード実行期間中において、同一ページに対する書き込み検知回数、言わばページ毎の書き込み頻度を求めた。**図 4** は、X 軸に書き込み発生回数をとり、Y 軸に、その発生回数以上の書き込みが生じたゲスト物理ページ数について、全ゲスト物理ページ数に対する割合を示す。まず、いずれのワークロードにおいても、すべてのゲスト物理ページが一様に書き込まれるのではなく、特定のページに何度も書き込みが発生する。ページマイグレーションのアルゴリズムが、書き込みが生じるページを特定して、特に書き込みが頻出するものから DRAM に配置できれば、DRAM への書き込みヒット率を高めることができるはずである。例えば、gobmk、hmmer、libquantum、h264ref、omnetpp 等においては書き込みが頻出するゲスト物理ページは全体のうち数パーセントであり、この数パーセントのゲスト物理ページを特定して、DRAM に配置すればよい。DRAM が全体のうち 5% (200MB) 以上存在する場合、書き込みが集中するページを一度 DRAM 上に移動すれば、以降は大半の書き込みを DRAM 上に集中できると予想される。gcc においては、全体の 20% (800MB) 以上の DRAM があれば、書き込みが生じるほとんどのページを DRAM に移動できる。DRAM の割当量がそれに満たない場合であっても、書き込みが多いページほど数は少ないため、それらを DRAM に配置できれば、少量の DRAM であっても DRAM への書き込みヒット率を高められる可能性がある。

以上の結果を以下にまとめる。

- ワークロードによっては、書き込みが生じるページ量が大きく変動する。書き込み傾向の変動に追随すべく、ページ配置の最適化は動的に実行すべきである。
- ページ配置の最適化は時間的局所性に基づくべきである。いずれのワークロードも、今書き込みが生じたページは直後においても再び書き込みが発生する可能性が高い。
- ページ配置の最適化は空間的局所性も考慮すべきである。いずれのワークロードも、書き込みが生じるゲスト物理ページは、仮想マシンのゲスト物理ページ全体の一部である。また書き込みが多いページほど数が少ない傾向もある。DRAM への書き込みヒット率を高めるためには、書き込みが発生しうるページを特定し、書き込みが多いページを優先的に DRAM に配置すべきである。

3.4.2 Multi Queue

上記の要求を満たす新たなマイグレーションアルゴリズムを開発した。時間的な局所性 (recency) と空間的な局所性 (frequency) の両方を考慮するキャッシュアルゴリズムとして、我々は MQ (Multi Queue)[3] に着目している。ページマイグレーションのアルゴリズムとして、MQ を提案機構向けに設計し実装した。ページマイグレーションのアルゴリズムは、書き込みを検知したゲスト物理ページ番号を毎秒メモリアクセストレース機構から受け取り、5 秒ごとに DRAM と MRAM 間で交換すべきゲスト物理ページ番号の組を出力する。現状アルゴリズムの細部については改良中であり、本稿では概要のみを記す。

我々の MQ は複数のキューからなり、各要素はゲスト物理ページに相当する。それぞれのキューの末尾に存在するページほど最近書き込みがあったことを示し、また上段のキューほど過去の書き込み検知回数が多いページを保持する傾向になる。初期状態では最下段のキュー Q_0 に DRAM 上にあるゲスト物理ページがつながれている。DRAM 上にあるページに書き込みがあると、そのページをキューの末尾に移動する。この時、これまでにそのページに対し

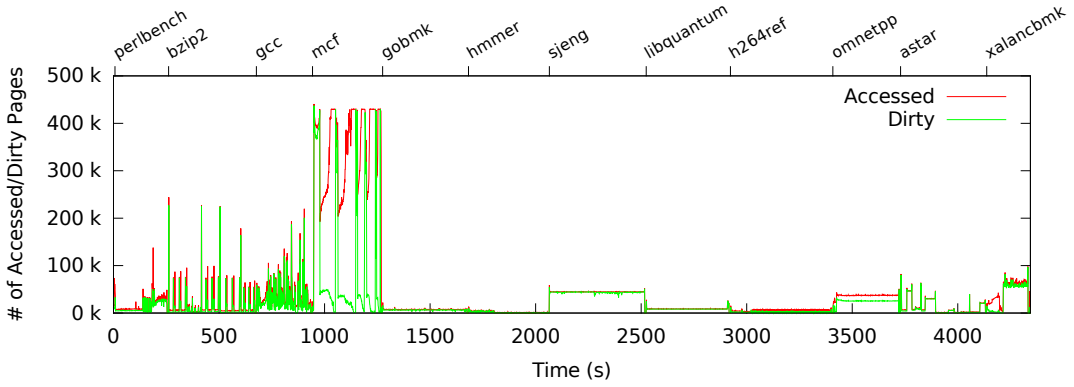

図 2　SPEC CPU 2006 実行中のアクセス済みおよび書き込み済みページ数

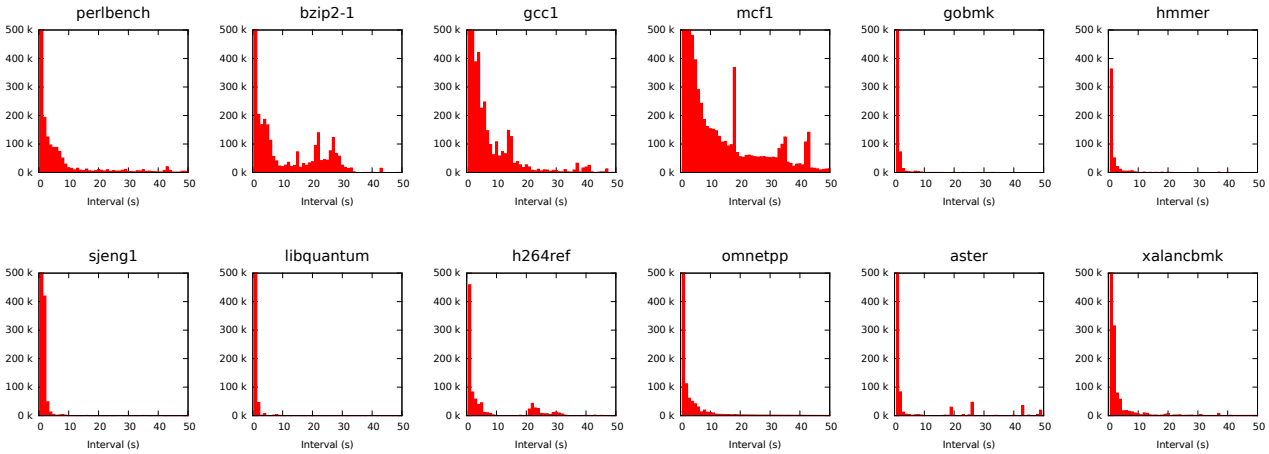

図 3　ゲスト物理ページ書き込みの時間的局所性 (SPEC CPU 2006)
X 軸は同一ゲスト物理ページに対する書き込みを検知した間隔、Y 軸はその発生回数を示す。

て生じた書き込み検知の回数 n に対して、$\lfloor log(n) \rfloor$ 段目のキュー $Q_{\lfloor log(n) \rfloor}$ を選択してその末尾に移動する。つまり、これまでの書き込み検知回数が閾値を超えると、上段のキューに移動する (promote、昇格する)。MRAM 上にあるページに書き込みがあると、すでにいずれかのキューに存在する場合は同様にキュー $Q_{\lfloor log(n) \rfloor}$ の末尾に移動する。いずれかのキューに存在しない場合は Q_0 の末尾に追加する。

一方、MQ は、各キューの先頭にあるページ (つまりそのキューにおいて最も古いページ) を定期的に参照し、その最後の書き込み検知時刻からの経過時間が一定の閾値を超えると、そのページを下段のキューの先頭に移動する (demote、降格する)。最下段のキュー Q_0 の先頭にあるページが降格する場合には、そのページが DRAM 上に存在するならば、MRAM への移動候補を保持する Q_{victim} の末尾につなぐ。MRAM 上に存在するならば、過去の書き込み検知回数をクリアし MQ から取り除く。

5 秒に一度ページマイグレーションを行う際には、上段のキューから順に、末尾から先頭へという順番で MRAM 上に存在するページを見つけ、その MRAM ページを DRAM への移動候補とする。MRAM への移動候補を保持する Q_{victim} に要素があれば、その先頭に存在する DRAM ページを交換対象とする。その DRAM ページは Q_{victim} から取り除く。以上を指定のマイグレーション回数まで、移動対象のページがなくなるまで繰り返し、DRAM と MRAM 間で交換すべきゲスト物理ページ番号の組を出力する。

時間的局所性が存在する対象において有用なアルゴリズムとして LRU (Least Recently Used) が知られている。LRU についても同様に提案機構向けにページマイグレーションのアルゴリズムを実装した。現状のプロトタイプにおいて、SPEC CPU のトレースデータをもとに、MQ および LRU それぞれを用いた場合における DRAM へのヒット率を分析した結果、MQ に一定の優位性があることを確認している。詳細については別途報告する予定である。

4. 動作確認

現状のプロトタイプ実装を用いて提案機構の基本的な動作を確認した。仮想マシンは仮想 CPU を一つ、RAM

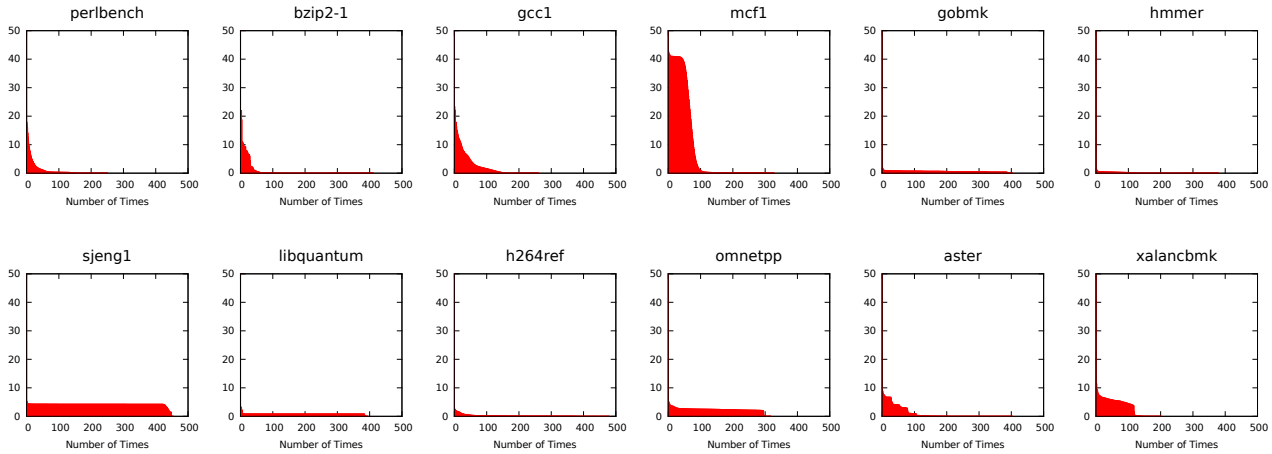

図 4 ゲスト物理ページ書き込みの空間的局所性 (SPEC CPU 2006)
X 軸は同一ゲスト物理ページに対する書き込み発生回数、Y 軸は、その発生回数以上の書き込みが生じたゲスト物理ページ数について、全ゲスト物理ページ数に対する割合を示す。

を 4GB 有する。ゲスト OS からは 4GB の RAM は単一の DRAM として見えるものの、ハイパーバイザ側では実際には 100MB の DRAM と 3.9GB の MRAM からなる。初期状態においては、ゲスト物理アドレスの先頭から MRAM を割り付け、その後に DRAM を割り付けた。実際に MRAM は入手できないため、一部の DRAM 領域を MRAM として見立てて用いている。

4.1 評価手法

DRAM および MRAM へのメモリ書き込み量を評価の指標とする。提案機構によりメモリ書き込みが頻出するページを DRAM に移動し、MRAM へのメモリ書き込み量を減らせることを確認する。一般的に用いられる評価手法として、メモリアクセスのトレースデータをもとにしてシミュレーションを行う手法や、サイクルアキュレート・シミュレータを用いる手法が存在する。しかし、ゲスト OS 全体のように対象とするソフトウェアの規模が大きい場合、しばしば適用が難しくなる。トレースデータが膨大になるほか、シミュレーション時間が極めて長時間に及んでしまう。

そこで、ハードウェアが備えるメモリチャネルごとの性能監視機構を利用した新たな性能評価手法も開発した。Intel Xeon プロセッサが備える性能監視機構 (Performance Monitoring Unit、PMU) を用いて、DRAM および MRAM それぞれに対するメモリ読み書き量を計測する。最近の Xeon プロセッサはメモリチャネルごとのメモリ読み書き量を計測できる。また BIOS でメモリチャネル・インタリービングを無効にすると、各メモリチャネルがそれぞれ異なる物理アドレス領域にマップされる。そこで、我々の性能評価手法では、あらかじめ各メモリチャネルがどの物理アドレス領域にマップされるのか調べておき、その情報もとに仮想マシンに対する DRAM プールおよび MRAM プールをそれぞれ異なるメモリチャネルにマッピングする。それぞれのメモリに対する読み書き量をメモリチャネルごとの PMU を通じて計測可能にした。

表 1 評価実験機におけるメモリチャネルと物理アドレス範囲の対応

Channel	Start Offset	Size	Usage
0	0x0	2GB	Host OS
0	0xd00000000	14GB	Host OS
1	0x100000000	16GB	Host OS
2	0x500000000	16GB	DRAM Pool
3	0x900000000	16GB	MRAM Pool

評価に用いた物理計算機は、CPU として Intel Xeon E5-2618L v3 を 1 個搭載し、4 つのメモリチャネルのそれぞれに 16GB がつながった計 64GB の DRAM を有する。この計算機におけるメモリチャネルと物理アドレス範囲の対応を**表 1** に示す。Usage 欄に示したように、チャネル 2 番のアドレス範囲を DRAM プールとして、チャネル 3 番のアドレス範囲を MRAM プールとして以降の実験で用いた。

4.2 stress

最初に簡易なワークロードを用いて基礎的な実験を行った。負荷生成プログラムである `stress` をゲスト OS 上で動かした。32MB のメモリを確保し、その領域に対して繰り返し読み書きを行う。**図 5** に DRAM および MRAM への書き込み量を、**図 6** にページマイグレーション数を示す。

時刻 16 秒付近で `stress` を開始した。当初 32MB のメモリはすべて MRAM 上に存在していたため、MRAM に対して 2500MB/s 程度の書き込みが発生している。40 秒付近からページマイグレーションを有効にした。5 秒ごとに最大 1000 ページまで再配置を行った。再配置を実行するごとに、MRAM への書き込みが減少し、その分 DRAM への書き込みが増加した。最終的にほぼすべての書き込みが DRAM 上に移動した。仮想マシンに割り当てた 100MB

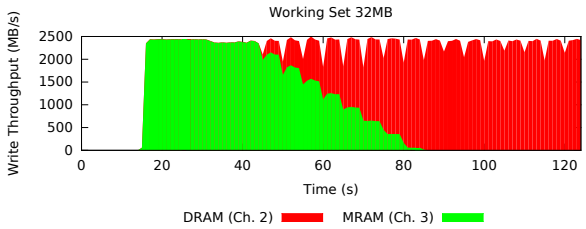

図 5　DRAM および MRAM への書き込み量 (stress、提案機構有効)

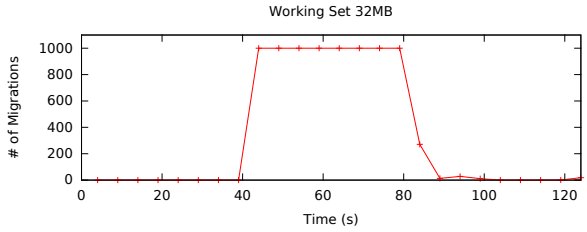

図 6　ページマイグレーション数 (stress、提案機構有効)

の DRAM 上に、書き込みが頻発する 32MB の領域が再配置され収まったことがわかる。提案機構が正しく動作していることが確認できた。

再配置を行う際に一時的に仮想マシンを停止しているため、書き込み量の瞬間的な低下が観察できる。1000 ページを再配置する際には、現状数百 ms の期間仮想マシンを停止している。今後実装を改善することで停止時間を短縮する予定である。

4.3 SPEC CPU

同様の実験を SPEC CPU 2006 を用いて行った。DRAM および MRAM への書き込み量について提案機構を有効にした場合を**図 7** に、無効にした場合を**図 8** に示す。例えば、ワークロード `libquantum` はメモリへの書き込み量が 4GB/s 程度発生している。提案機構を有効にした場合はその大半を DRAM に振り向けることに成功している。また、`gcc` や `mcf` 等は、提案機構有効時においても MRAM への書き込みが依然として顕著である。その原因としては、ワークロードが使用するメモリサイズが実際の DRAM サイズを超えるのが一因であると考えられる。**図 9** は提案機構有効時において実行したページマイグレーション数を示す。ワークロードが使用するメモリが DRAM に収まりきらない場合は、一度に許容するページマイグレーション数である 1000 ページ分の再配置を何度も繰り返している。一方、使用するメモリが DRAM 上に収まった場合には、それ以降ページの再配置はあまり行われていない。

提案機構を有効にした場合と無効にした場合それぞれの実験期間 (約 4200 秒) において、DRAM および MRAM に書き込まれたデータの総量を**表 2** に示す。MRAM への書き込み量は、提案機構を無効にした場合 1.8TB 程度もあったのに対して、提案機構を有効にすることで 293GB 程度まで抑制することに成功した。今回の実験は改良途中のプロトタイプ実装を用いたものであるものの、提案機構によって MRAM への書き込み量が抑制できることが確認できた。今後より詳細な定量的な評価に取り組む。

表 2　DRAM および MRAM へのデータ書き込み総量 (MB)

	Enabled	Disabled
DRAM (Ch. 2)	1,520,757.89	881.58
MRAM (Ch. 3)	293,381.39	1,789,716.84

提案機構を有効にした場合と無効にした場合を比較。

5. 関連研究

文献 [4] では、PCM と DRAM からなるハイブリッド型のメモリシステムを想定して、メモリコントローラによるページマイグレーション機構を提案している。ページマイグレーションのアルゴリズムとして MQ を独自に改良して用いている。我々の提案機構のようにメモリへの書き込みがあったことを 1 秒ごとに検知してマイグレーションを判断するのではなく、メモリアクセスがあるたびにメモリコントローラ内でページのマイグレーションを判断する。我々の研究は特殊なメモリコントローラに依存しない手法としてハイパーバイザでのページマイグレーションを検討しており、研究内容が異なる。ページマイグレーションをハードウェアではなくソフトウェアで実装しても一定の有用性があるという点を今後詳しく検証する予定である。

文献 [5] では、CPU コア上に 3 次元積層されたオンチップメモリと従来のオフチップのメモリの両方が搭載されたシステムを想定して、ハイパーバイザ (Xen) によるページマイグレーション機構を提案している。前者のオンチップメモリは低遅延でアクセスが可能であるものの容量に限りがあり、後者のオフチップメモリはアクセス遅延が大きいものの容量が大きい。そこで頻繁にアクセスされるページをオンチップメモリ側にマイグレーションして性能の向上を図っている。我々の研究でもハイパーバイザレベルでのページマイグレーションに着目しており両者の間に類似性がある。しかし、我々の研究では、ページマイグレーションを通じてハイブリッド型のメモリシステムにおける消費エネルギーの削減を目指している点で研究内容が異なる。

6. まとめ

STT-MRAM と DRAM を搭載した計算機を想定して、仮想マシンに対して動的にメモリ割り当てを最適化するハイパーバイザを試作した。仮想マシンに対して透過的に、書き込みが頻出するページを DRAM に配置し、消費電力を抑制することを目指す。提案ハイパーバイザは、DRAM および MRAM から仮想マシンのメモリを割り当てる機構、軽量なメモリアクセスのトレース機構、ページマイグレーションを判断するアルゴリズム、DRAM と MRAM

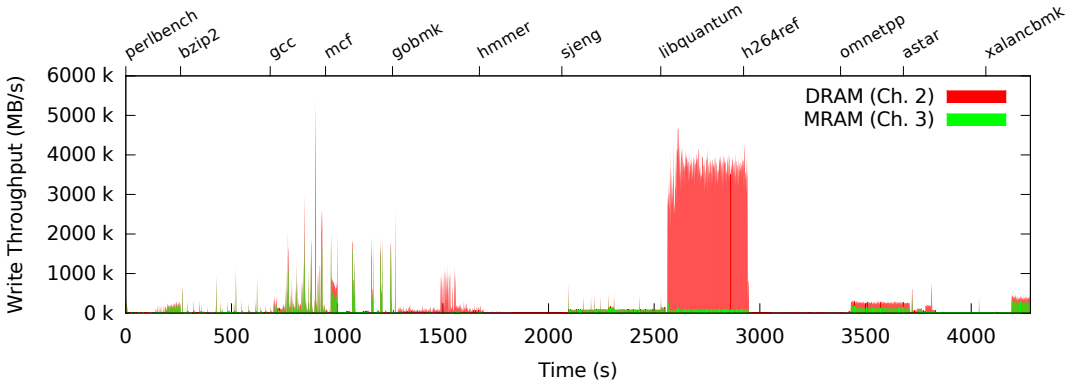

図 7　DRAM および MRAM への書き込み量 (SPEC CPU 2006、提案機構有効)

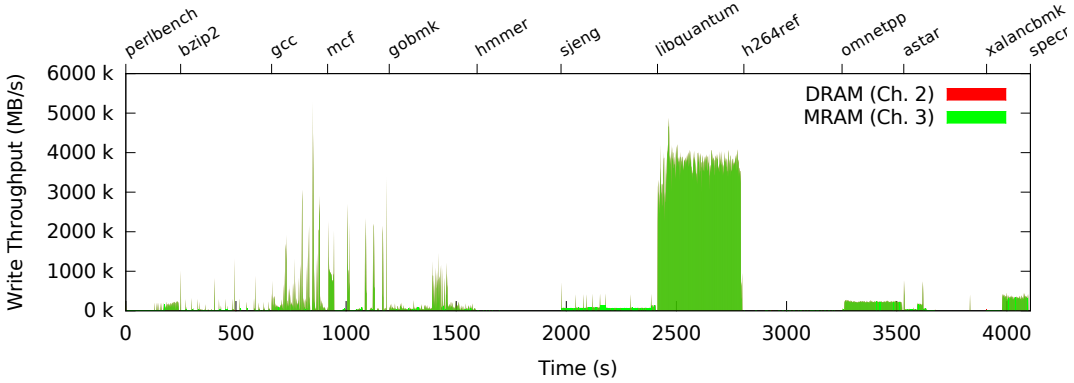

図 8　DRAM および MRAM への書き込み量 (SPEC CPU 2006、提案機構無効)

間でページをスワップする機構からなる。Qemu/KVM に対して提案ハイパーバイザを開発している。本稿では Work-in-Progress として、その概要とプロトタイプの実装について述べた。簡易な実験を行った結果、提案機構が書き込みの多いページを DRAM へ動的に再配置できることを確認した。今後はプロトタイプ実装の完成度を上げて、より詳細な評価実験を行う予定である。

参考文献

[1] The International Technology Roadmap for Semiconductors (ITRS): International Technology Roadmap for Semiconductors 2013 Edition (2013).
[2] 広渕崇宏，高野了成，工藤知宏：オペレーティングシステムに対して透過的なメモリアクセス・トレース機構，情報処理学会研究報告 (2015-OS-133)，情報処理学会，pp. 1–6 (2015).
[3] Zhou, Y., Philbin, J. and Li, K.: The Multi-Queue Replacement Algorithm for Second Level Buffer Caches, *Proceedings of the General Track: 2001 USENIX Annual Technical Conference*, USENIX Association, pp. 91–104 (2001).
[4] Ramos, L., Gorbatov, E. and Bianchini, R.: Page Placement in Hybrid Memory Systems, *Proceedings of the 2011 ACM International Conference on Supercomputing (ICS'11)*, ACM, pp. 85–95 (2011).
[5] Lee, M., Gupta, V. and Schwan, K.: Software-controlled Transparent Management of Heterogeneous Memory Resources in Virtualized Systems, *Proceedings of the ACM SIGPLAN Workshop on Memory Systems Performance and Correctness (MSPC2013)*, ACM, pp. 5:1–5:6 (2013).

Vol.2015-OS-135 No.5
Vol.2015-EMB-39 No.5
2015/11/24

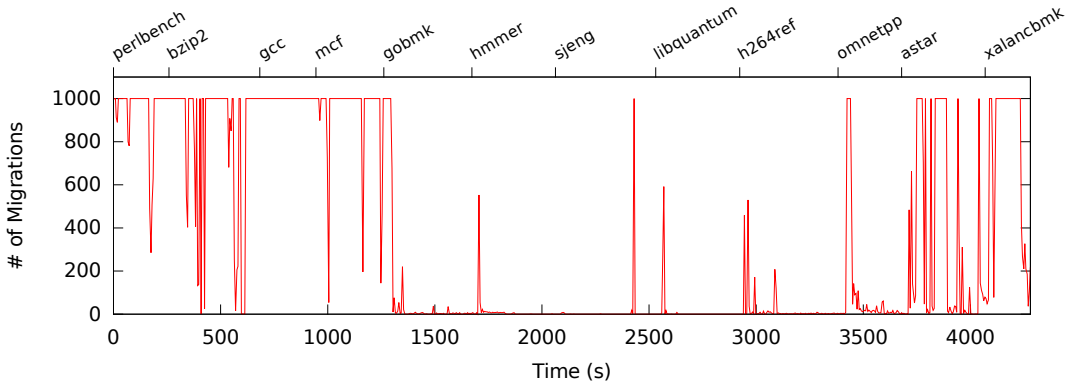

図 9 ページマイグレーション数 (SPEC CPU 2006、提案機構有効)