準仮想化ページフォルトを用いたポストコピー型ライブマイグレーションの性能向上手法

広渕 崇宏[†] 山幡 為佐久[††] 伊藤 智[†]

仮想マシン（VM）のライブマイグレーションの新たな手法としてポストコピー型に注目が集まっている。今日普及しているプレコピー型と比べ、マイグレーション完了時間が短くまた確定的であるという利点がある。しかし、ポストコピー型は実行ホスト切り替え後に性能低下が発生しうるため、その抑制が実装上の課題とされる。そこで、本稿では、ポストコピー型ライブマイグレーションにおける VM のページフォルトを準仮想化することで、実行ホスト切り替え後の性能低下を緩和する手法を提案する。ゲスト OS が未転送のメモリページにアクセスした場合にゲスト OS を一時的に停止するのではなく、そのメモリページにアクセスした、ゲスト OS 上のプロセスのみを一時停止する。そして、ゲスト OS 上の他のプロセスの実行はそのまま継続する。複数のプロセスもしくはネイティブスレッドからなるアプリケーションにおいて、実行ホスト切り替え後の性能低下が抑制できることが期待される。提案手法を我々が開発するポストコピー型ライブマイグレーション実装（Yabusame）において実装し、性能評価を行った。ウェブサーバに対するベンチマークにおいて、提案手法は実行ホスト切り替え後の性能低下を緩和することが確認できた。

Improving Performance of Postcopy Live Migration with Para-virtualized Page Fault

Takahiro Hirofuchi ,[†] Isaku Yamahata [††] and Satoshi Itoh[†]

Postcopy live migration is a promising alternative of virtual machine (VM) migration, which transfers memory pages after switching the execution host of a VM. It allows a shorter and more deterministic migration time than precopy migration. There is, however, a possibility that postcopy migration would degrade VM performance just after switching the execution host. In this paper, we propose a performance improvement technique of postcopy migration, extending the para-virtualized page fault mechanism of a virtual machine monitor. When the guest operating system accesses a not-yet-transferred memory page, our proposed mechanism allows the guest kernel to defer the execution of the current process until the page data is transferred. In parallel with the page transfer, the guest kernel can yield VCPU to other active processes. We implemented the proposed technique in our postcopy migration mechanism for Qemu/KVM. Through experiments, we confirmed that our technique successfully alleviated performance degradation of postcopy migration for a web server benchmark.

1. はじめに

データセンタの運用効率を向上させる技術として、仮想マシン（VM）の再配置機能（ライブマイグレーション）に注目が集まっている。VM を停止することなく他の物理マシンに移動する技術であり、サーバ機器のメンテナンスやロードバランスを容易にする。

今日ではプレコピー型とよばれるライブマイグレーション手法[1),2)] が広く用いられているが、異なる特徴を持つポストコピー型[3)~5)] とよばれる手法にも注目が集まりつつある。プレコピー型では、VM のメモリを宛先ホストに全て転送した後に実行ホストを切り替える。メモリ転送中も VM は移動元ホストで動作し続けるため、更新されたメモリページを再帰的に宛先に転送する必要がある。一方、ポストコピー型は、先に VM の実行ホストを宛先に切り替え、その後メモリページを転送する。実行ホストの切り替えにともなうデータ転送量は数百 KB であり、一般的な環境では瞬時に実行ホストを切り替えられる。再帰的なメモリ転送が不要なためマイグレーション全体が完了する時間も短い。

我々は先行研究[5)] で仮想マシンモニタ Qemu/KVM[6)]

† 産業技術総合研究所 / National Institute of Advanced Industrial Science and Technology (AIST)
†† ヴィーエー・リナックス・システムズ・ジャパン株式会社 / VA Linux Systems Japan K.K.

に対するポストコピー型ライブマイグレーションを試作し、その有用性を検証した。資源消費量に応じて動的に VM の配置を調整するシステムに適用すると、プレコピー型よりも高いレベルの性能保証や消費電力削減効果が得られることを確認した[7),8)]。現在、Qemu/KVM のメインラインに統合できる実用化レベルの品質を持ったポストコピー型ライブマイグレーション実装（Yabusame）をオープンソースで開発している[9)~11)]。

ポストコピー型ライブマイグレーションを開発する上での課題として、実行ホスト切り替え直後に発生する VM の性能低下を抑制することが挙げられる。ポストコピー型は、基本的には、VM が新たなメモリページアクセスするたび、移動元からメモリページの内容を転送する。この際、わずかな時間ではあるものの、VM を一時的に停止する必要があるため、VM の性能が低下してしまう。我々の先行研究においては、オンデマンドなメモリ転送と並行して、バックグラウンドで重要なメモリページをあらかじめ宛先に極力転送することで一時的な停止時間の低減を図っている。この手法により一定の改善効果が得られる。しかし、いまだ改善の余地が残っており、性能低下のさらなる緩和手法が求められている。

そこで本稿では、ポストコピー型ライブマイグレーションにおける VM のページフォルトを準仮想化することで、実行ホスト切り替え後の性能低下を緩和する手法を提案する。ゲスト OS が未転送のメモリページにアクセスした場合にゲスト OS を一時的に停止するのではなく、そのメモリページにアクセスした、ゲスト OS 上のプロセスのみを一時停止する。そして、ゲスト OS 上の他のプロセスの実行はそのまま継続する。複数のプロセスもしくはネイティブスレッドからなるアプリケーションにおいて、実行ホスト切り替え後の性能低下が抑制できることが期待される。ゲスト OS が未転送のメモリページにアクセスしページフォルトが発生すると VM Exit が発生する。通常はメモリページの準備ができるまでゲスト OS は停止するため、ゲスト OS はページフォルトを関知することはない。しかし、提案手法では、未転送のメモリページにアクセスしたことを通知する特殊な割り込みを VM に投入し即座に VM Entry することで、ゲスト OS の実行を継続可能にしている。ゲスト OS において特別な対応が必要になるものの、KVM の Asynchronous Page Fault（APF）機能と実装を共通化したことで、APF 機能を標準的に備える Linux-2.6.38 以降であれば、改変を施すことなくゲスト OS として提案機構に対応する。

2 節で、Qemu/KVM およびポストコピー型ライブマイグレーションの実装を説明し、ポストコピー型ライブマイグレーションの課題を述べる。3 節で提案機構の詳細を述べ、4 節で評価実験を行う。5 節で関連研究を述べ、6 節でまとめる。

2. ポストコピー型ライブマイグレーションの課題

図 1 に Qemu/KVM および我々が開発しているポストコピー型ライブマイグレーション（Yabusame）の概要を示す。本稿での提案機構は図中に含まれない。ポストコピー型ライブマイグレーションにおいて発生しうる実行ホスト切り替え後の性能低下要因について説明する。なお、Yabusame の実装は基本的な設計は先行研究[5)] で発表したプロトタイプと同様であるものの、より Qemu と親和性の高い設計となるように注意深く再設計されている。KVM ドライバによる仮想化支援を利用しない Qemu の実行モード（Qemu TCG モード[12)]）においても動作し、またプレコピー型マイグレーションのコードと処理が共通化されている等の違いがある。

Qemu/KVM において、VM はホスト OS 上で一つの Qemu プロセスとして存在し、そのプロセスは複数のスレッドから構成されている。VCPU スレッドは VM に割り当てた仮想 CPU の個数に応じて存在し、各仮想 CPU においてゲスト OS を駆動する役割を持つ。メインスレッドは、主に VM のネットワーク I/O 等、各種デバイスのエミュレーションを提供する。また、Qemu はディスク I/O を処理するため Posix の非同期 I/O と同等の機構を実装しているため、必要に応じて複数のディスク I/O スレッドが起動する。ホスト OS カーネル中には、シャドウページテーブルの作成等を担う KVM ドライバが存在する。

ポストコピー型ライブマイグレーション実行時には、移動先ホストにおいて我々が開発した UMEM ドライバを有効にする。そして、Qemu プロセスは、UMEM ドライバが提供する匿名メモリページを VM の RAM としてマッピングする。また、Qemu プロセスはメモリページの転送を担う UMEM デーモンをその子プロセスとして起動する。

ポストコピー型ライブマイグレーションを開始すると、最初に移動元ホストで VM を停止する（図 1 中 a)。次に VM の CPU レジスタやデバイスの状態を移動先ホストに転送し、即座に移動先ホストで VM を開始する（図中 b および c)。その後、VM が新たな

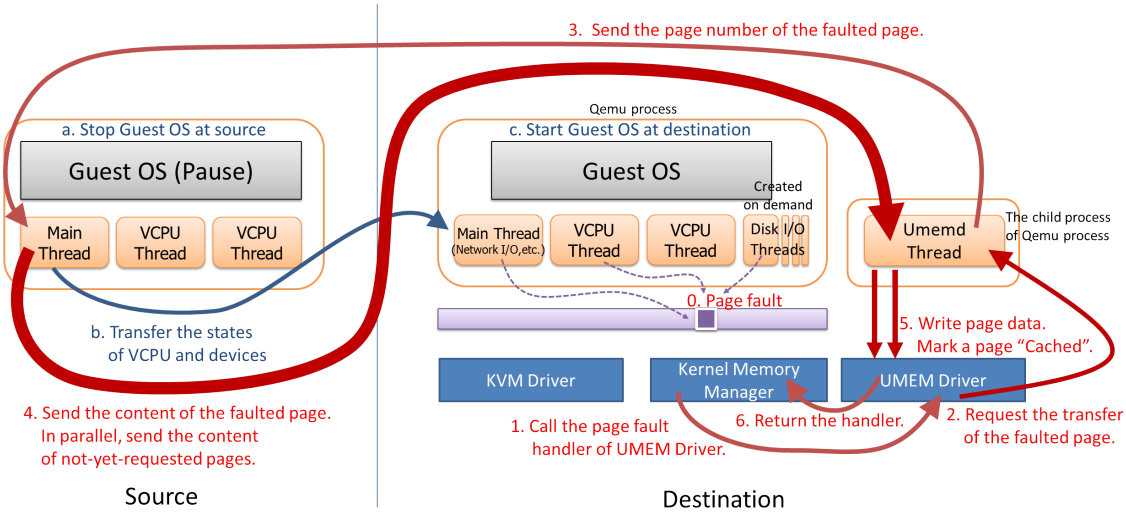

図 1 Qemu/KVM およびポストコピー型ライブマイグレーションの概要

メモリページにアクセスするたび、オンデマンドに移動元ホストからメモリページを転送する。また、オンデマンドな転送と並行してバックグラウンドで残りのメモリページを移動先に転送する。全てのメモリページが転送できたら、移動元ホストに残しておいたメモリページを破棄する。

図 1 中の 0 から 6 においてメモリ転送の仕組みを説明している。移動先ホストで VM を起動した直後は、いずれのメモリページも Qemu プロセスに割り付けられていない。VM を構成するいずれかのスレッドが、各メモリページに最初にアクセスすると、移動先ホストでページフォルトが発生する。このページフォルトが VCPU スレッドにおいてゲスト OS を実行中に発生した場合、VCPU スレッドが担う仮想 CPU はゲスト OS の実行を抜ける（VM Exit）。いずれのスレッドで発生した場合も、ホスト OS でページフォルトの原因となったスレッドの実行がブロックし、ホスト OS のメモリ管理機構は UMEM ドライバのページフォルトハンドラを呼び出す。

対象メモリページの内容がバックグラウンドでの転送により既に転送済みであれば、UMEM ドライバは即座にページフォルトハンドラを完了する。ホスト OS カーネルはスレッドの実行を再開する。ページフォルトが VCPU スレッドにおいてゲスト OS 実行中に発生していたのであれば、その仮想 CPU においてゲスト OS の実行を再開する（VM Entry）。この場合極めて短時間当該スレッドを停止するのみである。本稿ではこれをマイナーフォルトと呼ぶ。

一方、対象メモリページの内容が転送済みでない場合は、UMEM ドライバはユーザランドの UMEM デーモンにメモリページの取得を依頼する。UMEM デーモンは移動元へ対象メモリページのページ番号を通知し、移動元の Qemu プロセスはそのメモリページの内容を UMEM デーモンに送信する。UMEM デーモンは、ページ内容を UMEM ドライバに伝達し、そのページを転送済みとしてマークすることを依頼する。UMEM ドライバはページフォルトハンドラを完了し、当該スレッドの実行が再開される。VCPU スレッドが停止していた場合、マイナーフォルトと同様に、その仮想 CPU においてゲスト OS の実行を再開する。本稿ではこれをメジャーフォルトと呼ぶ。

メジャーフォルトはネットワーク越しのデータ転送をともなうため、マイナーフォルトよりもはるかに大きな処理時間を要する。例えば、本稿の実験機材においては、マイナーフォルトは数マイクロ秒であるのに対し、メジャーフォルトはギガビットイーサネットを用いた場合で数百マイクロ秒にも及ぶ。そのためメジャーフォルトが頻出するとゲスト OS で実行中のアプリケーションの性能が低下する要因となってしまう。

3. 提案手法

そこで、ポストコピー型ライブマイグレーションにおける VM のページフォルトを準仮想化することで、実行ホスト切り替え後の性能低下を緩和する手法を提案する。提案機構は KVM が備える Asynchrnous Page Fault（APF）機能を拡張して実装した。APF は、ゲスト OS がホスト OS 上でスワップアウトされているメモリページにアクセスした場合に、そのメモ

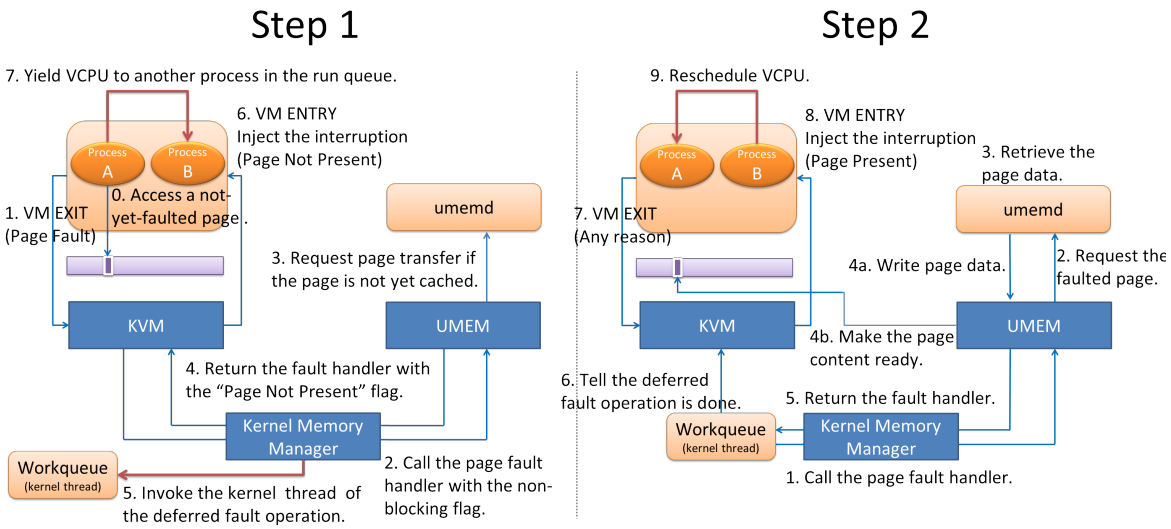

図 2 提案機構の動作概要

リページのスワップイン処理を行いながら、同時にゲスト OS の実行も継続する機能である。提案機構においては、UMEM ドライバにおいて APF に対応した特殊なページフォルト処理を実装した。ページフォルトが起きたメモリページの内容を移動元から取得しながら、同時にゲスト OS の実行も継続できる。注意深くコードを精査した結果、すでに APF に対応した KVM ドライバやゲスト OS カーネルであれば、改変することなくポストコピー型ライブマイグレーションに対応できることがわかった。

提案機構の動作概要を**図 2** に示す。はじめに、メジャーフォルトが発生すると、メモリページが未だに移動元から転送されていないことをゲスト OS に対して即座に通知する（図 2 中 Step 1）。

(0) ゲスト OS があるメモリページにはじめてアクセスする。
(1) ページフォルトが発生し VM の実行から抜ける。
(2) KVM ドライバはフォルトしたページを Qemu プロセス空間へ割り付けようとする。ただしノンブロッキングで試みる。ホスト OS のメモリ管理機構が UMEM ドライバのページフォルトハンドラをノンブロッキングフラグ付きで呼び出す。
(3) UMEM ドライバは転送済みのメモリページを記録しているビットマップを走査し、メモリページが未転送であればメジャーフォルトが確定する。UMEM デーモンにページ転送を依頼する。
(4) UMEM ドライバはページ不在フラグ付きでページフォルトハンドラを完了する。
(5) ホスト OS のメモリ管理機構は遅延ページフォルト処理用のカーネルスレッド（ワークキュー）を起動する。
(6) KVM ドライバは VM に割り込みを投入しページ不在を通知する。
(7) ゲスト OS カーネルの割り込みハンドラがページの不在を把握し、ページフォルトを引き起こしたプロセスの実行を停止する。他に CPU 待ちのプロセスが存在すれば、そのプロセスを実行する。

以上の動作においてブロックする処理は存在せず、後述する実験環境等においては数マイクロ秒で完了する。

次に、メモリページの転送を待ち、転送が完了したことをゲスト OS に対して通知する（図 2 中 Step 2）。

(1) ワークキューは KVM ドライバ同様にフォルトしたページを Qemu プロセス空間へ割り付けようとする。メモリ管理機構が UMEM ドライバのページフォルトハンドラを再度呼び出す。ただしノンブロッキングフラグを付けないため、通常のフォルト処理となる。
(2) UMEM ドライバは転送済みのメモリページを記録しているビットマップを再度走査し、メモリページが未転送であれば UMEM デーモンにページ転送を依頼する。
(3) UMEM デーモンは移動元からメモリページを受信する。
(4) UMEM デーモンはページ内容を VM メモリに書き込み、UMEM ドライバにおいて当該ページを転送済みと記録する。

(5) UMEM ドライバはページフォルトハンドラを完了する。
(6) ワークキューにおいてマッピング処理が完了する。KVM ドライバに対してページフォルト処理が完了したことを通知する。
(7) 何らかの理由で VM Exit が発生する。あるいは、過去に HLT 命令を実行したままの VCPU であればすでに VM Exit 済みであり、この場合 VCPU スレッドの待機状態を解除する。
(8) VM Entry 時に VM に割り込みを投入し、ページの準備ができたことを通知する。
(9) ゲスト OS の割り込みハンドラがページの準備完了を把握し、ページフォルトを引き起こしたプロセスを再び実行待ちキューに入れる。

この図 2 中第二段階におけるページフォルトハンドラ呼び出しは、ネットワークを通じてメモリページ内容が転送されることを待つ可能性があるため、ブロッキングな処理である。しかし、ワークキューという VCPU スレッドとは別のカーネルスレッドが担うため、VM の実行を直接妨げることはない。

提案機構は、VCPU スレッドがゲスト OS を直接実行している場合、つまり VM Entry した状態において機能する。VCPU スレッドが VM Exit した状態において、あるメモリページにはじめてアクセスした場合は機能しない。例えば、VCPU スレッドは APIC 等の処理のためゲスト OS のページテーブルを参照する場合がある。このとき、ページテーブルを保存しているメモリページに対する VCPU スレッドのアクセスは、提案機構が元にする APF の対象外である。また、メインスレッド等の VCPU スレッド以外のスレッドによるページアクセスも同様に対象外である。このような場合、メモリページの内容が移動元から転送されるまで、そのスレッドの実行はブロックされる。

VM は仮想 CPU の個数に応じて複数の VCPU スレッドを持つ。そのような場合にも各 VCPU スレッドごとに提案機構は動作する。また、現在ワークキューで取得待ち中のメモリページに対して、再度同じ VCPU スレッドからページフォルトが発生した場合は、その VCPU スレッドの実行を停止する。このような場合をダブルフォルトと呼び、そのメモリページの取得を完了した後に VCPU スレッドの実行を再開する。つまり通常のページフォルト処理にフォールバックする。

以上の提案機構を Qemu/KVM および Linux カーネルに対して実装した。2012 年 6 月において最新の Qemu/KVM および Linux カーネルに対して動作を確認している。我々は提案機構の実装を全てオープンソースで公開しており、ユーザからのフィードバックを元に改良を進めている。

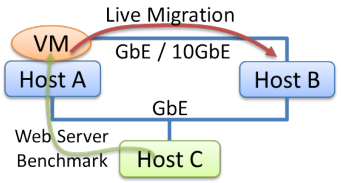

図 3 実験環境

4. 評　　価

提案手法の有効性を検証するため評価実験を行った。はじめに簡易なマルチスレッドプログラムを用いて、提案機構の動作を確認した。次に、ウェブサーバを用いて性能評価を行った。実験環境を**図 3** に示す。ホスト A および B は VM の移動元および移動先計算機であり、マイグレーションにともなうデータ転送用に GbE あるいは 10GbE ネットワークを用いる。ホスト C では仮想ディスクサーバおよびウェブサーバのベンチマークプログラムを起動する。物理マシンはいずれも Intel Xeon E5620 を 2 個（計 8CPU コア）および 24GBbytes のメモリを搭載している。VM には仮想 CPU を 1 コアおよびメモリを 1GBytes 割り当てた。本環境はアドレス変換を支援するハードウェア機能 Intel EPT（Extended Page Table）を備える。

以降の実験では、特に言及しない限り 10GbE ネットワークを用いた結果について説明する。ping により計測したホスト間の伝送遅延は 150us 程度である。また、オンデマンドな転送のみを有効にし、メジャーフォルトが発生したページおよびその前後それぞれ 8 ページを一度に転送する設定とした ☆。現状の実装では、バックグラウンド転送がオンデマンド転送を妨げて VM の性能を劣化させる可能性を避けるため、オンデマンド転送のスループットが十分小さくなった後に、バックグラウンド転送を有効にし残りのページを全て転送することを想定している。本実験ではマイグレーション直後の性能低下に注目するため、計測期間中にバックグラウンド転送は起動しない。

4.1 マイクロベンチマーク

最初に、ゲスト OS 上で簡易なマルチスレッドプログラムを用いて、提案機構の基本的動作を確認した。

☆ Yabusame においては、メジャーフォルト解決時に一度に転送するページ数に対して、任意の値を設定できる。ワークロードの特性等によって最適な値は異なるが、一般的に性能低下を十分小さくできる値として前後 8 ページという値を利用している。

```
198: [vm exit]                              => (Step 1: 1)
203: [umem enter]  gfn=0x180d               => (Step 1: 2)
209: [umem return] gfn=0x180d need_retry => (Step 1: 4)
226: [umem enter]  gfn=0x180d               => (Step 2: 1)
242: [vm entry]                             => (Step 1: 6)
     (skip events in 163usec)
405: [umem return] gfn=0x180d done          => (Step 2: 5)
     (skip events in 99usec)
504: [vm entry]                             => (Step 2: 8)
```

図 4 ページ番号 0x180d に関するホスト OS 側提案機構の動作（各行先頭は usec 単位の時刻。gfn はページ番号。各行末尾は図 2 中の動作段階に対応。）

```
634: [apf wait] tid=4884 gfn=0x180d  => (Step 1: 6)
674: [schedule] tid=4884 to tid=4885 => (Step 1: 7)
900: [apf wake]          gfn=0x180d  => (Step 2: 8)
948: [schedule] tid=4885 to tid=4884 => (Step 2: 9)
```

図 5 ページ番号 0x180d に関するゲスト OS 側提案機構の動作（各行先頭は usec 単位の時刻。ただし時刻はホスト OS 側と厳密に同期していないことに注意。gfn はページ番号、tid はカーネルスレッド ID。各行末尾は図 2 中の動作段階に対応。）

はじめに 4 つのスレッドがそれぞれ 200MBytes のメモリを確保する。その後、マイグレーションの開始と同時に、各スレッドは各自のメモリ領域を先頭から 1Byte ずつアクセスし、メモリ領域の最後に到達するまでの時間を各スレッドそれぞれで計測した。提案機構が有効な時には到達時間はそれぞれ約 9.8 秒であり、提案機構が無効な時では約 17.6 秒であった。提案機構は到達時間を約 44%削減することに成功している。

提案機構の動作を詳細に調べるため、カーネル内のイベントを取得する機構である SystemTap[13] を用いながら、再度同じ実験を行った。ホスト OS 上で、VM Entry および Exit、UMEM ドライバのページフォルトハンドラの呼び出しおよび完了を記録した。またゲスト OS 上で、APF の割り込みハンドラの呼び出し、プロセス切り替えを記録した。**図 4** および**図 5** に、ページ番号 0x180d のページフォルトに関連する提案機構の動作を示す。各行末尾は図 2 中の動作段階に対応する。図 4 においてホスト OS 時刻 198usec にページフォルトを原因とする VM Exit が発生し、ホスト OS カーネルが UMEM ドライバのページフォルトハンドラを呼び出した。しかし、ホスト OS 時刻 209 秒においてページ内容が未だ転送されていないことが判明している。ゲスト OS は、図 5 においてゲスト OS 時刻 634usec にページが未転送であることを把握し、ゲスト OS 時刻 674usec にページフォルトを引き起こしたスレッド（tid=4884）から別のスレッド（tid=4885）へ実行を切り替えている。その後、ホスト OS 上でホスト OS 時刻 405usec にページの取得が完了している。ゲスト OS はゲスト OS 時刻 900usec にページの取得を把握し、その後ページの取得待ちであったスレッドを再開している。

以上により、提案機構がページフォルト取得処理を行いながら、並行してゲスト OS を実行できることが確認できた。SystemTap の出力を解析したところ、ベンチマークプログラム実行中における単位実時間あたりのゲスト OS 実行時間（VMX Non Root モードの走行時間）の割合は、提案機構有効時は 0.55、無効時は 0.38 程度であった。提案機構に対して理想的なベンチマークにおいて、提案機構によりゲスト OS の実行効率が向上したことが確認できた。

4.2 ウェブサーバ

次に、実際のアプリケーションを用いて評価した。複数のプロセスを起動するウェブサーバ型のアプリケーションは、提案機構によってマイグレーションにともなう性能低下を抑制できると予想される。そこで、VM 上にウェブサーバプログラムを設定し、ライブマイグレーション前後の処理性能を計測した。あらかじめウェブコンテンツとしてランダムなバイト列からなる 100KBytes のファイルを 5000 個生成したうえで、Apache ウェブサーバをプレフォークモードで起動した。1 サーバプロセスで 1 クライアント接続を処理する動作モードである。図 3 中ホスト C から、ウェブサーバ用のベンチマークプログラムである Siege[14] を用いて、ウェブサーバの応答速度の時間変化を計測する。Siege は、HTTP クライアントを 128 スレッド起動し、各スレッドがランダムな順番でウェブコンテンツファイルを取得するよう設定した。Apache ウェブサーバについても最大 128 サーバプロセス起動するよう設定してある。

VM のメモリサイズが 1GBytes であるため、合計 500MBytes のウェブコンテンツ・ファイルは全てゲスト OS のページキャッシュに載せることができる。ベンチマークを開始後ウェブサーバのスループットが安定した状態になってから、ポストコピー型ライブマイグレーションを実行した。マイグレーション実行時にはすべてのコンテンツがキャッシュされていると推測される。

図 6 および**図 7** は、それぞれ提案機構を有効にした場合および無効にした場合について、各 HTTP リクエストの応答時間の時系列変化を示す。各点は各リクエストの到着時刻（横軸値）および応答時間（縦軸値）を表し、曲線は 100msec ごとに集計した平均応答時間の時間変化も示す。了解性を高めるため応答時間が 0.5 秒および 1.0 秒である位置に補助線も描画した。それぞれの場合において、時刻 10 秒付近でマイ

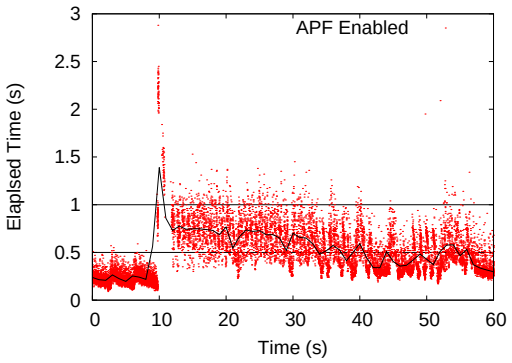

図 6 リクエスト応答時間の分布（提案機構有効）。各点は各リクエストの到着時刻（横軸値）および応答時間（縦軸値）を表す。曲線は平均応答時間の時間変化。

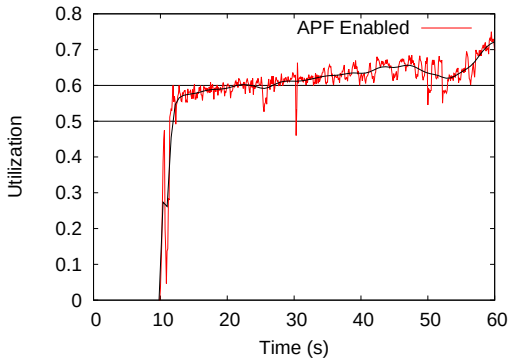

図 8 単位実時間に占めるゲスト OS 実行時間の割合（提案機構有効）。横軸は時刻、縦軸は割合。傾向把握を容易にするためベジェ曲線による補助線も記載。

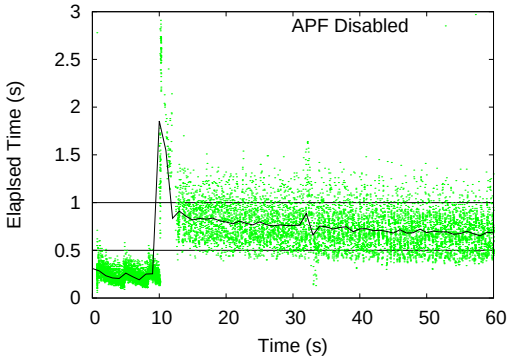

図 7 リクエスト応答時間の分布（提案機構無効）。各点は各リクエストの到着時刻（横軸値）および応答時間（縦軸値）を表す。曲線は平均応答時間の時間変化。

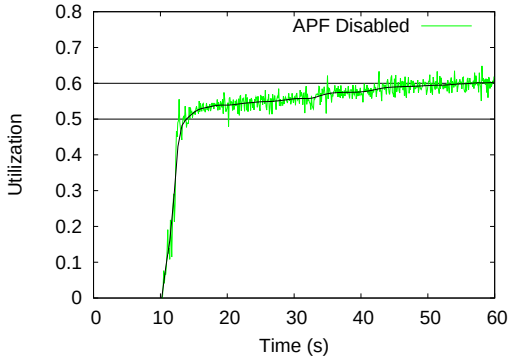

図 9 単位実時間に占めるゲスト OS 実行時間の割合（提案機構無効）。横軸は時刻、縦軸は割合。傾向把握を容易にするためベジェ曲線による補助線も記載。

グレーションを開始したため、リクエスト応答時間が上昇している。このとき提案機構を無効にした場合は平均応答時間が 1.8 秒程度に上昇するのに対し、有効にした場合には 1.3 秒程度に留まっている。提案機構を有効にした場合は、その後の応答時間も短縮できている。無効の場合は時刻 35 秒付近で 0.75 秒程度であるのに対して、有効時は 0.5 秒を切る程度にまで小さい。マイグレーション開始以前の応答時間は 0.25 秒程度であることを考えれば、提案機構はこの時刻において応答時間の増加量を半減することに成功している。

図 8 および**図 9** は、それぞれ提案機構を有効にした場合および無効にした場合において、単位実時間に占めるゲスト OS 実行時間の割合を表す。SystemTap により VMX Non Root モードの走行時間を計測し、100msec ごとに集計を行った。了解性を高めるため、縦軸値が 0.5 および 0.6 を示す補助線を描いている。またベジェ曲線による補助線も記載した。提案機構を有効にした場合の方が、マイグレーション開始直後において、ゲスト OS 実行時間の増加が急激である。有効時は、開始後 2 秒程度でゲスト OS 実行時間を 0.55 程度まで高められている。一方、無効時は同時刻は 0.5 秒弱に留まる。その後、時刻 35 秒付近で有効時は 0.65 程度まで上昇するのに対し、無効時は 0.55 程度に留まっている。提案機構は、ゲスト OS 実行時間をより多く確保することを可能にし、その結果としてリクエスト応答時間を抑制できたと説明できる。

図 10 は、提案機構を有効にした場合において、ページフォルト処理と並行してゲスト OS を実行できた時間を表している。ページフォルト処理待ちのページが存在するにもかかわらず、提案機構により VMX Non Root モードを走行できた時間を集計した。開始直後は実時間に占める割合が 0.15 程度まで上昇し、その後 0.05 付近を徐々に減少している。このグラフの値は、提案機構の有無によるゲスト OS 実行時間の差分と概して同等であり、提案機構によるゲスト OS 実行時間の増加量が概ね説明できる。

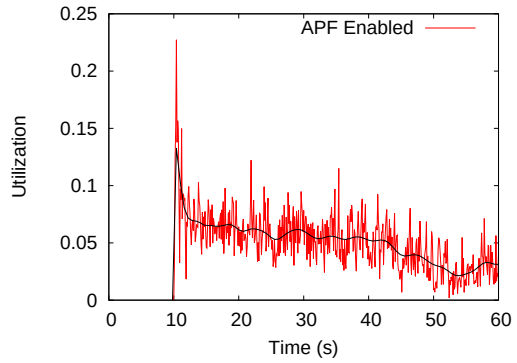

図 10 提案機構によりページフォルト処理に並行してゲスト OS を実行した時間。横軸は時刻、縦軸は割合。傾向把握を容易にするためベジェ曲線による補助線も記載。

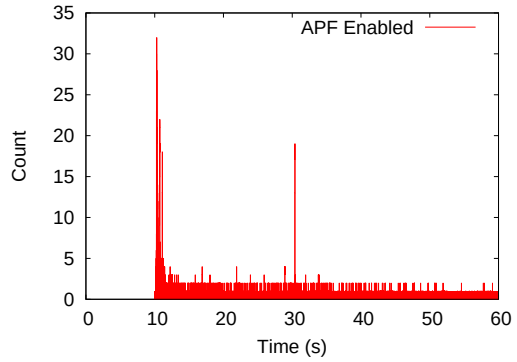

図 11 提案機構において各 VM Entry 時にページフォルト完了待ちであったページ数。横軸は時刻、縦軸は個数。

図 11 は、各 VM Entry 時にページフォルト完了待ちであったページ数を表す。マイグレーション開始直後には 5 個から 32 個程度のページがページ取得待ちキューにしばしば存在し、その後も数個のページ取得が連続して発生している。提案機構が有効に機能する状況が発生し続けていたことが確認できる。

図 12 はダブルフォルトの発生回数を表す。マイグレーション開始直後には、100msec あたり 10 回および 25 回程度のスパイクが 2 度発生しているものの、その後は散発的に数回程度発生するのみである。開始直後は、プログラムのテキスト領域などの複数のプロセスが共有するページに対するフォルトが発生するため、転送中のメモリページに対するフォルトが再度別のプロセスによって引き起こされる可能性が高い。その後は、ページキャッシュ中のウェブコンテンツデータへの参照が主たるページフォルト要因であり、本実験において Siege の各クライアントスレッドはランダムな順番でファイルを取得するため、HTTP サーバプロセスがほぼ同時に同じページキャッシュを参照すること

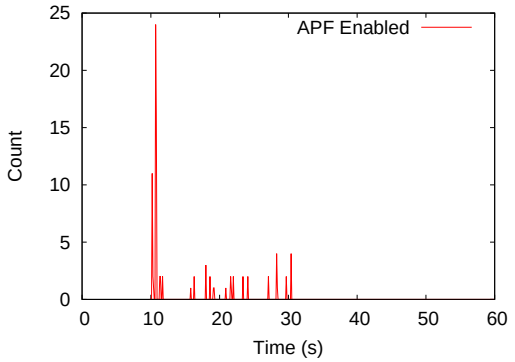

図 12 提案機構におけるダブルフォルトの 100msec あたりの発生回数。横軸は時刻、縦軸は個数。

は起こりにくくなっている。

図 13 は提案機構を有効にした場合において、マイナーフォルトおよびメジャーフォルトの発生回数を表す。VCPU スレッドがゲスト OS を実行中に発生したメジャーフォルトとそれ以外のメジャーフォルトを分けて描いている。メインスレッドに起因するフォルトや、VCPU スレッドの VMX Root モードにおけるゲスト OS ページの参照に起因するフォルトは、提案機構は対象とできない。しかし、それらはマイグレーション開始直後にごく少量発生するのみであり、性能への影響は小さい。また本実験ではメジャーフォルトを発生させたページについて前後の 8 ページ分も同時に転送していたため、マイナーフォルト発生回数を増加させて、メジャーフォルト発生回数を抑制できていた。なお、メジャーフォルト発生ページのみを転送する設定とした場合は、提案機構が作用するメジャーフォルトの発生回数が増大するため、提案機構による性能改善効果が大きくなった。しかし、マイグレーション直後のウェブサーバ応答時間がいずれも悪化してしまうため、メジャーフォルト発生前後のページも同時に転送することが望ましいといえる。

以上により、提案機構がページフォルト処理と並行してゲスト OS の実行を可能としたことで、ポストコピー型ライブマイグレーション中のゲスト OS 実行時間を増加させることができた。これによりウェブサーバのベンチマークにおいて応答時間を低減できた。複数のプロセスやスレッドからなる同様のアプリケーションにおいて性能向上が期待できる。なお、マイグレーション用のネットワークとして GbE を使用した場合も、若干ウェブサーバの応答時間およびゲスト OS 実行時間の割合の回復が遅くなるものの、ほぼ同等の結果が得られた。

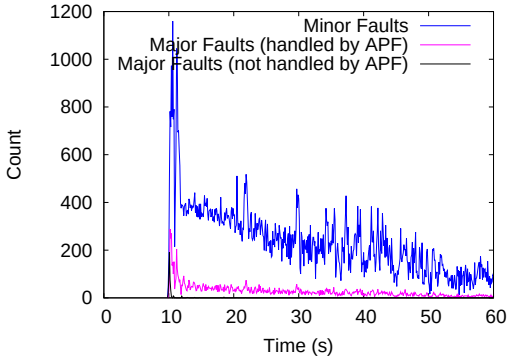

図 13 提案機構におけるページフォルトについて種類別の 100msec あたりの発生回数。横軸は時刻、縦軸は個数。提案機構が対象としないメジャーフォルト Major Faults (not handled by APF) は時刻 10 秒付近にごく少量発生している。

5. 関 連 研 究

VM のメモリがホスト OS 上でスワップアウトされるとページフォルト処理遅延の増加が問題となりえる。これはダブルページング問題[15] とよばれてきた。IBM の z/VM においては、対象メモリページがスワップアウトされていることを伝える擬似的なページフォルト割り込み（Pseudo-Page-Fault Interruption）を VM に投入し、ゲスト OS の実行を継続する仕組みが備わっている。KVM においても APF として同様の機構が実装されている。提案機構はこの仕組みをポストコピー型ライブマイグレーションに応用した。これらページフォルトを非同期化する仕組みは、これまでダブルページング問題の解決のために用いられてきたのみであり、ライブマイグレーションの性能向上に用いられた事例は我々の知る限り存在しない。

ポストコピー型ライブマイグレーションの性能向上について関連研究を挙げる。Snowflock[3] はポストコピー型ライブマイグレーションと同様の仕組みを用いて VM を他の物理ホスト上に高速に複製する機構である。メモリ管理機構は、アプリケーションが複製先ホストで新たに割り当てたページを区別して管理し、アプリケーションがそれらのページを参照しても移動元から転送しない。その後継の研究[16] では VM が作成したページテーブル等を解析し、関連性が高いメモリ領域をまとめて転送しメジャーフォルトの頻度を軽減する。文献 4) では，ゲスト OS に専用ドライバを組み込んで特殊なスワップデバイスを作ることで，ポストコピー型のライブマイグレーションを実装している．最も新しいメジャーフォルトが発生したページを中心にして前後のページを近いものから順に先送りすることで、メジャーフォルトの頻度を軽減している。

プレコピー型ライブマイグレーションを開始したあと、最終段階でポストコピー型ライブマイグレーションに移行する手法が研究されている[17),18]。VM のメモリサイズが大きくまたメモリの更新速度が速い場合、プレコピー型における再帰的なメモリコピーが収束しない可能性がある。そこで、マイグレーション開始後一定時間が経過したら再帰的なメモリコピーを中断して、ポストコピー型ライブマイグレーションに移行し残りのメモリページをオンデマンドに取得する。またデータ転送を RDMA で処理することで、メジャーフォルトの遅延を軽減している[19]。

我々が開発するポストコピー型ライブマイグレーションでは、メジャーフォルトが発生したページを中心として前後それぞれ指定の量だけ先送りする機能を備えている。今後プレコピー型との連携や RDMA との連携を考えている。また我々の先行研究[20] では、一連のライブマイグレーションの高速化を図るため、VM が以前存在したホストに戻った際にメモリを再利用する機構を開発し、ポストコピー型ライブマイグレーションの完了時間を短縮している。

データの取得に関する遅延をできる限り隠蔽するため、計算処理とデータ取得処理を同時並列的に実行する手法は、システムソフトウェアの様々な側面において利用されてきた。ポストコピー型ライブマイグレーションに比較的近い事例として、例えば、分散共有メモリの処理をマルチスレッド化することで、通信遅延の影響を軽減した研究[21] 等を挙げることができる。

6. ま と め

本稿では、ポストコピー型ライブマイグレーションにおける VM のページフォルトを準仮想化することで、実行ホスト切り替え後の性能低下を緩和する手法を提案した。ゲスト OS が未転送のメモリページにアクセスした場合にゲスト OS を一時的に停止するのではなく、そのメモリページにアクセスした、ゲスト OS 上のプロセスのみを一時停止する。そして、ゲスト OS 上の他のプロセスの実行はそのまま継続する。複数のプロセスもしくはネイティブスレッドからなるアプリケーションにおいて、実行ホスト切り替え後の性能低下が抑制できることが期待される。提案手法を我々が開発するポストコピー型ライブマイグレーション実装（Yabusame）において実装し、性能評価を行った。ウェブサーバに対するベンチマークにおいて、提案手法は実行ホスト切り替え後の性能低下を緩和することが確認できた。

今後の課題として、プレコピー型とポストコピー型のハイブリッド動作を Yabusame において実装することを検討している。プログラムのテキスト領域を保存するメモリ領域は VM の動作に不可欠でありながら、他の領域に比べ書き換え頻度は大きくないと予想される。これらのページをプレコピー型で先に転送した上で、ポストコピー型に移行すれば性能低下を更に抑制できると考えられる。関連研究で述べたように、再帰的なページ転送が収束しにくいというプレコピー型の欠点を補完できるという点で、オープンソースコミュニティからの要望も大きい。

謝辞 本研究は、科研費（23700048）およびCREST（情報システムの超低消費電力化を目指した技術革新と統合化技術）の助成を受けた。Yabusame の開発は、経済産業省およびエヌ・ティ・ティ・コミュニケーションズ株式会社の支援を受けた。Yabusame の現行バージョンは産業技術総合研究所の依頼にもとづき VA Linux Systems Japan 株式会社が中心となって開発している。

参 考 文 献

1) Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live migration of virtual machines. In *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation*, pp. 273–286. USENIX Association, 2005.
2) Michael Nelson, Beng-Hong Lim, and Greg Hutchins. Fast transparent migration for virtual machines. In *Proceedings of USENIX Annual Technical Conference*, pp. 25–25. USENIX Association, 2005.
3) Horacio Andrés Lagar-Cavilla, Joseph Andrew Whitney, Adin Scannell, Philip Patchin, Stephen M. Rumble, Eyal de Lara, Michael Brudno, and Mahadev Satyanarayanan. SnowFlock: Rapid Virtual Machine Cloning for Cloud Computing. In *Proceedings of the fourth ACM european conference on Computer systems*, pp. 1–12. ACM Press, 2009.
4) Michael R. Hines and Kartik Gopalan. Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning. In *Proceedings of the 5th International Conference on Virtual Execution Environments*, pp. 51–60. ACM Press, 2009.
5) 広渕崇宏, 中田秀基, 伊藤智, 関口智嗣. 既存VMM への適用が容易でゲスト透過なポストコピー型仮想マシン再配置機構. 情報処理学会論文誌:コンピューティングシステム, Vol. ACS31, pp. 248–262, Sep 2010.
6) Avi Kivity, Yaniv Kamay, Dor Laor, and Anthony Liguori. kvm: the Linux virtual machine monitor. In *Proceedings of the Linux Symposium*, pp. 225–230. The Linux Symposium, 2007.
7) Takahiro Hirofuchi, Hidemoto Nakada, Satoshi Itoh, and Satoshi Sekiguchi. Reactive cloud: Consolidating virtual machines with postcopy live migration. *IPSJ Transactions on Advanced Computing Systems*, Vol. ACS37, pp. 86–98, Mar 2012.
8) Takahiro Hirofuchi, Hidemoto Nakada, Satoshi Itoh, and Satoshi Sekiguchi. Making vm consolidation more energy-efficient by postcopy live migration. In *The Second International Conference on Cloud Computing, GRIDs, and Virtualization*, pp. 195–204. IARIA, Sep 2011.
9) Postcopy Live Migration for Qemu/KVM (Yabusame). `http://grivon.apgrid.org/quick-kvm-migration`.
10) Takahiro Hirofuchi and Isaku Yamahata. Yabusame: Postcopy live migration for qemu/kvm. KVM Forum 2011, Aug 2011.
11) Isaku Yamahata and Takahiro Hirofuchi. Postcopy live migration for kvm/qemu. LinuxCon Japan 2012, Jun 2012.
12) Fabrice Bellard. Qemu, a fast and portable dynamic translator. In *Proceedings of USENIX Annual Technical Conference*, pp. 41–41. USENIX Association, 2005.
13) Vara Prasad, William Cohen, Frank Ch. Eigler, Martin Hunt, Jim Keniston, and Brad Chen. Locating system problems using dynamic instrumentation. In *Proceedings of Linux Symposium 2005*, pp. 49–64. IEEE Computer Society, 2005.
14) Jeffrey Fulmer. `http://www.joedog.org/siege-home`.
15) Robert P. Goldberg and Robert Hassinger. The double paging anomaly. In *Proceedings of the national computer conference and exposition*, pp. 195–199. ACM Press, 1974.
16) Roy Bryant, Alexey Tumanov, Olga Irzak, Adin Scannell, Kaustubh Joshi, Matti Hiltunen, H. Andres Lagar-Cavilla, and Eyal de Lara. Kaleidoscope: cloud micro-elasticity via vm state coloring. In *Proceedings of the sixth conference on computer systems*, pp. 273–286. ACM Press, 2011.
17) William J. Armstrong, Richard L. Arndt, Timothy R. Marchini, Naresh Nayar, and Wolfram M. Sauer. IBM POWER6 partition mobil-

ity: moving virtual servers seamlessly between physical systems. *IBM Journal of Research and Development*, Vol. 51, No. 6, pp. 757–762, Nov 2007.

18) Benoit Hudzia. Enhancing live migration process for cpu and/or memory intensive vms running enterprise applications. KVM Forum 2011, Aug 2011.

19) Canturk Isci, Jiuxing Liu, Bulent Abali, Jeffrey O. Kephart, and Jack Kouloheris. Improving server utilization using fast virtual machine migration. *IBM Journal of Research and Development*, Vol. 55, No. 6, Nov 2011.

20) 穐山空道, 広渕崇宏, 高野了成, 本位田真一. 都鳥:メモリ再利用による連続するライブマイグレーションの最適化. 情報処理学会論文誌:コンピューティングシステム, Vol. ACS37, pp. 86–98, Mar 2012.

21) Kritchalach Thitikamol and Pete Keleher. Multi-threading and remote latency in software DSMs. In *Proceedings of the 17th International Conference on Distributed Computing Systems*, pp. 296–304. IEEE Computer Society, May 1997.