瞬間的な実行ホスト切り替えを可能とする 仮想マシンの高速ライブマイグレーション機構

広渕 崇宏　　中田 秀基　　伊藤 智　　関口 智嗣

産業技術総合研究所 情報技術研究部門

我々は，データセンタの運用効率を向上させるため，資源消費量に応じた動的な仮想マシン（VM）再配置を目指している．しかし，既存のライブマイグレーション機構は，実行ホストの切り替えに時間がかかり，すばやく負荷をバランスさせることが難しい．そこで，我々は，迅速な実行ホストの切り替えを可能とする，新たなライブマイグレーション機構を提案する．VM メモリページの転送を実行ホスト切り替え後に行うことで，実行ホストの切り替え時間を大幅に短縮し 1 秒以内に可能にする．競合する機構と比較して，仮想計算機モニタへの変更が少なくゲスト OS への改変が不要である点に優位性がある．マイグレーション動作を検証するため，SPECweb2005 を用いて評価実験を行った．負荷の高いウェブサーバを実行する VM であっても，1 秒以内に実行ホストを切り替えることができた．実行ホスト切り替え後には性能低下が見られるものの，先行キャッシュ機構により改善できた．

Relocating a Virtual Machine Instantaneously by an On-demand Memory Transfer Mechanism

Takahiro Hirofuchi　Hidemoto Nakada　Satoshi Itoh　Satoshi Sekiguchi

National Institute of Advanced Industrial Science and Technogy (AIST)

We are developing an efficient resource management system with aggressive virtual machine (VM) relocation among physical nodes in a datacenter. Existing live migration technologies, however, require long time to change the execution host of a VM; it is difficult to optimize VM packing on physical nodes dynamically, corresponding to ever-changing resource usages. In this paper, we propose an advanced live migration mechanism enabling instantaneous relocation of VMs. To minimize the time of switching the execution host, memory pages are transferred after a VM restarts at a destination host. A special character device driver allows transparent memory page retrieval from a source host for the running VM at the destination. In comparison with related work, the proposed mechanism supports guest operating systems without any modifications to them. The major parts of the mechanism are implemented independently of a virtual machine monitor. Experiments were conducted by using the SPECweb2005 benchmark. A running VM with heavily-loaded web servers was successfully relocated to a destination within one second. Temporal performance degradation after relocation was alleviated by pre-caching memory pages.

1　はじめに

計算機センタやデータセンタの資源運用の効率性を高めるために, 仮想計算機（VM）とそのライブマイグレーション技術が注目されている．仮想計算機技術によって計算機資源を抽象化して論理的に分割・共有できる．さらに仮想マシンモニタ（VMM）が備えるライブマイグレーション機能によって, VM を一切停止することなく異なる物理ノード上に再配置可能になる．

我々が進めている省エネデータセンタプロジェクトでは，ライブマイグレーションによって計算機クラスタ上の VM 配置を積極的に変更することで，データセンタの稼動エネルギーコストを削減す

ることを目指している．今日，Amazon EC2 [1] 等の仮想ホスティング・サービスプロバイダにおいては，VM に対して割り当てた性能値（性能の目安となる値）を元に，一つの物理ノードに対して配置する VM 数を固定的に決めている．たとえ，VM の CPU 消費量が少なくて，物理ノードの処理能力に余裕が生じていても，余剰計算資源を有効利用できない．しかし，我々のプロジェクトでは，VM の CPU 消費量をモニタしながら，動的に配置を最適化する．VM の CPU 消費量が少なければ，一つの物理ノードに対して，性能保証値から導かれる数よりも多数の VM を配置（*overcommit*）できる．また，VM の CPU 消費量が大きくなれば，性能保証を満たすべく VM を多数の物理ノードへ分散させる．VM の処理性能を保証しつつもハードウェア資源の稼動効率を向上させることで，一歩進んだ省エネルギー・効率化が可能になると考えている．

我々は既に遺伝的アルゴリズムを用いた動的な配置決定システム [2] を試作している．しかし，既存のライブマイグレーション実装 [3, 4] では，VM をホストしている物理ノードの変更に数十秒単位の時間がかかり，VM の CPU 負荷の増加に追随して，配置状態を速やかに変更することが困難であった．性能保証値を満たせない期間が長時間にわたってしまい，実際的な商用ホスティングサービスへの応用が難しかった．

先行研究 [5, 6] においては，実行ホストの切り替え時間を短縮するため，ポストコピー型と呼ばれるライブマイグレーション手法が提案されている．ライブマイグレーション開始時には，CPU レジスタやデバイスの状態のみを移動先ホストにコピーし，すぐさま VM の実行ホストを切り替える．その後，移動元ホストからオンデマンドにメモリページを取得している．しかし，いずれの手法も VMM のメモリ管理部分を大きく改変する必要があり，実用化レベルの品質にするには大きなコストを要する．また，ゲスト OS カーネルに対しても，メモリ管理部分を変更したり特殊なドライバを組み込むことが必要なため，データセンタのように顧客に VM をホストする環境では望ましくない．

そこで，本研究では，VM の迅速な実行ホスト切り替えを可能とする，新たなライブマイグレーション機構を提案する．マイグレーション開始後 1 秒以内に実行ホストを変更できる．VMM のひとつである KVM [4] を対象に，VM のメモリ領域を提供する特殊なデバイスドライバを開発し，移動元ホストからのオンデマンドなメモリ取得機構を実現している．VMM のコードをほとんど変更することなく，ゲスト OS を一切変更する必要もない．さらに，オンデマンドなメモリ取得と並行して，VM のメモリアクセスパターンに応じて重要領域を優先的にコピーすることで，ウェブサーバ等のワークロードにおいて再配置後の性能低下を抑制できる．提案手法は先行研究の手法よりも実用性が高く，近い将来，オープンソースコミュニティで開発されている VMM へ開発成果を統合できると考える．

2 節で，現在利用可能なマイグレーション技術では不十分であることを述べる．3 節で既存のポストコピー型マイグレーション機構を概観する．4 節で我々の提案機構について説明し，5 節でプロトタイプ実装の状況について述べる．6 節で評価実験を示し，7 節でまとめる．

2 VM 配置の最適化と既存ライブマイグレーションの問題点

我々の研究プロジェクトでは，顧客の VM をホスティングする仮想化データセンタを対象として，計算機クラスタにおける効率的な VM 配置手法を開発している．VM の性能値だけで VM の配置を決めるのではなく，実際の資源消費量によって VM 配置を動的に決定する．VM の資源使用量が少ないときには，顧客に示した性能値に関わらず，ひとつ物理ノード上に多数の VM を配置して効率的な運用が可能になる．余剰物理ノードを節電したり，あるいは別の目的のために利用できる．一方，VM の CPU 資源消費量が増大し，物理ノードの CPU 資源を超えたときには，顧客に示した性能値を満たすべく，すぐさま新たな配置状態に変更し過負荷状態を解消しなければならない．ゆえに，VM を実行しているホストを迅速に変更できる仕組みが必要となる．また，その仕組みは顧客に対して透過的である必要がある．

しかし，現在実用化レベルに達しているライブマイグレーションの実装は，いずれも VM のすべての

状態を移動先に転送してから実行ホストを切り替える．それゆえに，迅速に実行ホストを変更することができない．ライブマイグレーションの開始から実際に実行ホストを切り替えるまでの時間は，メモリサイズやネットワーク帯域，またメモリの更新頻度に依存するものの，例えば，Amazon EC2 の標準的な VM サイズ（メモリ 1.7GB）であれば，Gigabit Ethernet の環境で最低 13.6 秒（1.7GB / 1Gbps）を要する計算になる．メモリイメージのコピー中は VM は移動元で動作中であり，変更されたメモリページを再帰的に転送するため，実際にはさらに時間がかかってしまう．

また，メモリページの更新頻度が高い VM の場合，ページの転送速度がページの更新速度に追いつかず，有限時間内でマイグレーションが完了できない場合もある（6.3 節参照）．VMM の実装によっては，強制的にページ更新頻度を抑制するため，性能低下が顕著になる場合もある．また，更新ページの再送によって，移動が完了するまで長時間にわたって大量のデータ転送が発生してしまう．

以上のように，現在利用できるマイグレーション機構は迅速な実行ホストの切り替えには程遠く，新たに別の仕組みが求められる．

3 関連研究

先行研究においては，実行ホストの切り替え時間を短縮するため，ポストコピー型と呼ばれるライブマイグレーション機構が提案されている．ライブマイグレーション開始時には，CPU レジスタやデバイスの状態のみを移動先ホストにコピーし，すぐさま VM の実行ホストを切り替える．そして，移動元ホストからオンデマンドにメモリページを取得している．いずれの先行研究の手法も実行ホストの切り替え時間の短縮に成功している．また，更新ページを再帰的に送る必要がないため，データ転送量も削減できている．

Snowflock [5] は，動作中の VM を他の物理ホスト上に複製する機構を設けている．プロセスの `fork()` 処理に類似した VM 複製を制御する API（VM Fork）を利用することで，複数台のノードを用いた分散処理プログラムを容易に記述できる．VM を動的に複製する機構は，Snowflock が VM Fork 後に複製元 VM の実行を継続する点を除けば，ポストコピー型のライブマイグレーションの仕組みに近い．複製の際には，複製元の VM の動作を停止して，CPU レジスタやページテーブル等[*1]を複製先ホストに転送する．そして，すぐさま複製元 VM の実行を再開する．このとき複製時点のメモリイメージを上書きしないように，メモリの変更は新たに割り当てたページ上で行う．複製先ホストで VM の実行を再開し，メモリアクセスがあるたび複製元のホストからメモリページを取得する．Xen の準仮想化モードに対して実装されており，ゲスト OS のメモリ割り当ての振る舞いを改変することで，複製元ホストからのメモリ取得を減らしている．しかし，その反面ゲスト OS に対する改変が必要になっている．また，メモリアクセスをトラップするためにページフォルトを利用しているため，複製元および複製先双方の性能低下が懸念される．

文献 [6] 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，Xen の準仮想化モードに対して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．ゲスト OS のメモリ領域の一部をスワップデバイスにすることができ，Xen の準仮想化モードにおけるメモリ抽象化を使って[*2]，スワップアウト・イン時にメモリコピーが発生しないようにしている．マイグレーションする際には，ゲスト OS はスワップアウト可能なメモリページをすべてスワップデバイスに書き出す．次に VM の実行を停止し，スワップアウトできなかったメモリ領域や CPU 状態を移動先ホストへコピーする．移動先ホストで VM が実行を再開すると，大半のメモリページがスワップアウトされた状態になっている．VM が新たにメモリページにアクセスするたびにスワップインが発生し，専用ドライバが移動元ホストから徐々にメモリページを取得していく．ゲスト OS のスワップ機能を使用することで VMM そのものに対する変更量を抑えている．しかし，ゲスト OS に対して特殊なドライバを組み込む必要があり，Xen の準仮想化モード特有の機能

[*1] 1GB メモリの VM の場合 1MB 程度である．

[*2] ゲスト OS から見た物理アドレス（Pseudo Physical Address）と実際の物理アドレス（Machine Frame Number）とのマッピングを変更する．

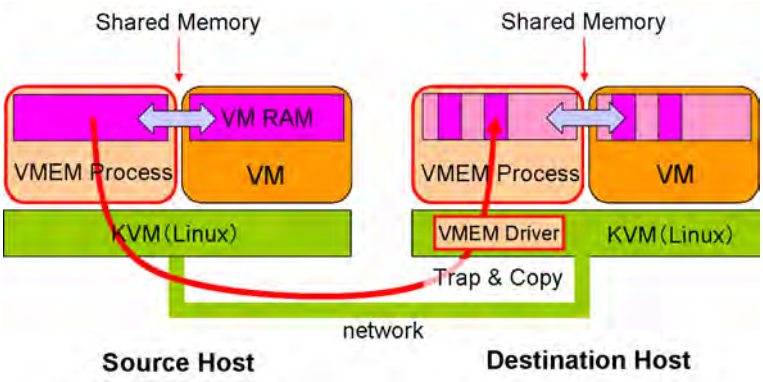

図 1　提案機構の概要

に依存してしまっている．

いずれの手法もあらかじめゲスト OS 内部を改変しなければならない．Linux 以外へのゲスト OS に対応するためには新たな開発作業が必要になり，OS がバージョンアップするたびに対応作業を要する可能性がある．Amazon EC2 など VM 内部の自由なカスタマイズを許す IaaS サービスプロバイダにおいては，特定のゲスト OS やその内部の機構に依存したマイグレーションは好ましくない．

また，Snowflock については実装が公開されているものの，試験的な実装でありそのままでは実証的な環境で使用するのは難しい．現状，実際に利用できるポストコピー型ライブマイグレーションは存在していない．

4　提案機構

仮想ホスティングデータセンタを対象とした動的な VM 配置最適化に対して，ポストコピー型ライブマイグレーションは有用であると考える．しかし，既存機構では不十分であり，新たに以下の要件を満たす仕組みが必要となる．

- 実用化レベルの品質を容易に達成できる仕組みであること．既存のポストコピー型機構は VMM への変更点が大きく，しかも仮想化の要となるメモリ管理部分のコードを改変してしまっている．結果，今日まで実用化できていない．
- VM 内部の仕組みに依存しない仕組みであること．既存のポストコピー型機構はゲスト VM 内部の対応が必要になる．VMM を運用するデータセンタ側と，VM 内部にサービスを構築する顧客の側の，両方の管理ドメインにまたがった仕組みとなってしまい，現実的ではない．

そこで，我々は，VM 内部への変更が一切不要であり，VMM 自体への変更も小さいポストコピー型ライブマイグレーション機構を新たに提案する．VMM として KVM（Kernel-based Virtual Machine）を利用し，VM メモリ領域を提供する特殊なデバイスファイルを通して，実行ホスト切り替え後のオンデマンドなメモリ取得を実現する．以下，本節では KVM を簡単に説明した後，提案機構の概要を述べる．

4.1　KVM

KVM は，オープンソースで開発が進められている VMM であり，そのドライバを組み込んだ Linux カーネルをハイパーバイザーとする．Intel VT あるいは AMD-V という CPU の仮想化機能を使用することで，物理ハードウェア向けの OS を改変することなく小さなオーバヘッドで VM 上で起動できる．ユーザランドのハードウェアエミュレータである QEMU に対する拡張として実装されており，VM はホスト OS からは通常のプロセスとして見える．従来型のライブマイグレーションに対応している．

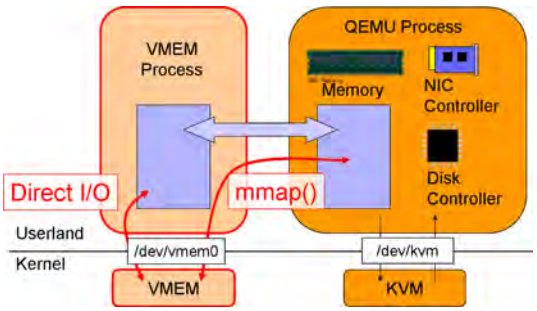

図 2　VMEM デバイスの概要

4.2　提案機構の概要

提案機構では，VM のメモリ領域を特殊なデバイスファイルにマップすることで，実行ホスト切り替え後のオンデマンドなメモリ取得を VMM のコードをほとんど改変することなく実現する．その概要を**図** 1 に示す．本稿では，オンデマンドなメモリ取得を実現する仕組みを VMEM デバイスとよぶ．

4.2.1　VMEM デバイス

我々が開発した VMEM デバイス（**図** 2）は，Direct I/O および `mmap()` によって，VMEM プロセスと`/dev/vmem0` を `mmap()` したプロセスの間で，メモリ領域を共有する仕組みである．しかも，`mmap()` したプロセスが，あるページに最初にアクセスした瞬間をトラップできる．

ホスト OS に対して VMEM ドライバ（`vmem.ko`）をロードすると，専用キャラクタデバイス（`/dev/vmem0` [*3]）が作成される．また，この時 `/dev/vmem0` に対してメモリを割り当てるために，補助プログラムである VMEM プロセスを起動する．VMEM プロセスは，`mmap()` 用のメモリをユーザ空間で確保し，その領域を VMEM ドライバに通知する．ユーザ空間の VMEM プロセスとカーネル空間の VMEM ドライバは，Direct I/O で確保したメモリページを共有する．

そして，あるプロセスが`/dev/vmem0` を `mmap()` すると，VMEM ドライバは VMEM プロセスが確保したメモリ領域を提供する．結果，`mmap()` したプロセスと VMEM プロセスが，同一のメモリ領域を共有することになる．

[*3] 実際には複数の VM に対応するため`/dev/vmem1` や `/dev/vmem2` 等も作成される．

また，`mmap()` したプロセスが，メモリ領域中のあるページに初めてアクセスした際には，ページフォルトが発生して，カーネルによって対象ページ番号が VMEM ドライバに通知される（VMEM ドライバのページフォルトハンドラが呼ばれる）．この時，VMEM ドライバのページフォルトハンドラは，VMEM プロセスに対してページ番号を通知し，対象ページの内容を用意するよう命令できる．VMEM デバイスは KVM のライブマイグレーションに特化した仕組みではなく，特定メモリページへのアクセスをトラップし，その中身を用意するための汎用的な機構といえる．

4.2.2　KVM による VMEM デバイスの利用

KVM は，VM のメモリ領域を QEMU プロセス中のユーザランドメモリ空間に確保する．そこで，移動先ホストで KVM が VM のメモリ領域を割り当てるコードにおいて，`/dev/zero` の代わりに `/dev/vmem0` を `mmap()` するよう変更する．また，移動元ホストでは，VMEM ドライバを用いないで単にメモリファイルシステム上で確保したファイル（`/dev/shm/kvm/mem0`）を `mmap()` するよう変更する．このファイルを `open()` すれば VM のメモリ領域を読み込める．実行ホスト切り替え後には，このファイルから切り替え直前のメモリページ内容を取得する．

4.2.3　VM 移動処理

提案手法による VM の移動処理を**図** 3 に示す．実行ホスト切り替え後には，VM のメモリアクセスに応じて 2 段階目から 5 段階目の処理が繰り返される．

0. 移動元ホストで VM を停止して，ゲスト OS の実行を中断する．QEMU プロセスのメモリ空間内には，停止時点のメモリ内容，CPU レジスタやデバイスの状態が存在する．
1. CPU レジスタとデバイスの状態を取り出し，移動先ホストに転送し QEMU プロセス中にロードする．そして移動先ホストで VM の実行を再開する．
2. VM がメモリにアクセスする．
3. もしそのメモリページが移動先ホストにおいて最初にアクセスされたものであれば，VMEM

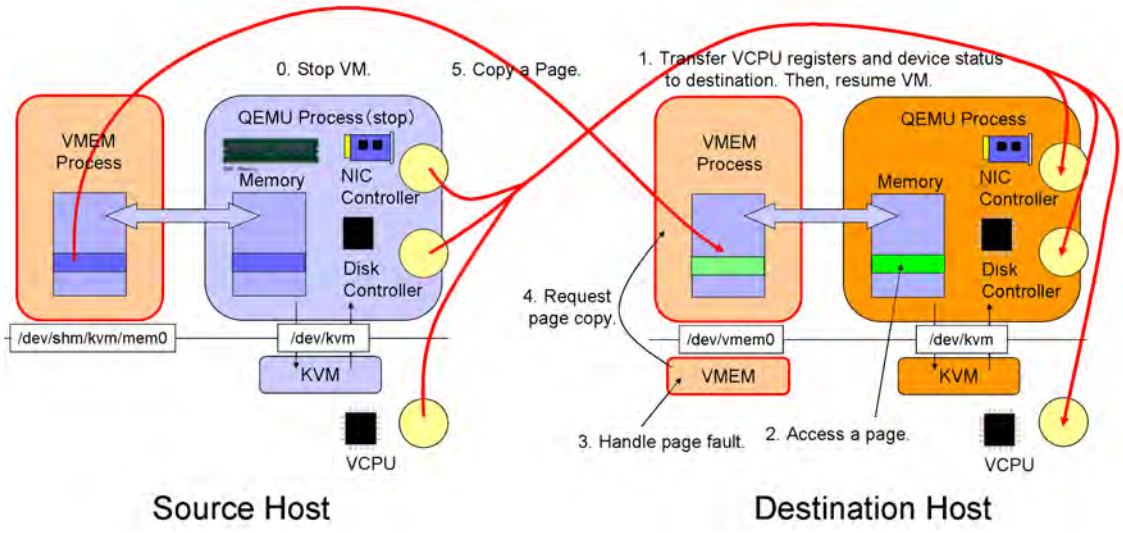

図 3　再配置動作の概要

ドライバのページフォルトハンドラがよばれる．VM の実行を一時停止し（4）（5）の処理を行う．

4. VMEM ドライバのページフォルトハンドラが VMEM プロセスにメモリページ取得を要請する．
5. 移動元ホストからメモリページ内容を取得し VM メモリ領域に書き込む．VM の実行を再開する．

このオンデマンドのメモリ取得処理および後述するバックグラウンドでのメモリ取得処理によって，最終的にすべてのメモリページが転送されると，移動元ホストへの依存性がなくなる．以後，移動先ホストのみで動作できる．移動先ホストでは，移動元ホストへの接続を終了する．また，移動元ホストでは，停止中の VM を破棄し，確保したメモリ領域を開放する．

また，メモリの遠隔取得による性能低下を抑制するため，以下の工夫を行っている．まず，移動元からのメモリ取得の際には，要求があった 1 ページ（4KB）のみ転送するのではなく，そのページに続く複数のページを一度にコピーする．あるページに対するページフォルト発生よりも前に，あらかじめ移動先ホストに内容を保存しておくことで，ネットワーク越しのページ転送回数を削減する．現在の実装では，連続する 128 ページの範囲において未転送のページを一度にコピーしている．

さらに，オンデマンドのメモリ取得と並行してバックグラウンドでのメモリ取得も動作させる．オンデマンドコピーを妨げないように優先度を低くしながら，残りのメモリページを全てコピーしメモリ領域全体の再配置を完了する．すばやく全メモリページの再配置を完了すれば，ネットワーク越しのページ転送回数を削減して性能低下を抑制できる．そこで，バックグラウンドのメモリ取得においては，ページフォルトが頻出する領域から先にコピーすることが考えられる．[*4]

なお，VM を 2 回以上移動することも可能である．この場合，`/dev/shm/kvm/mem0` の代わりに，`/dev/vmem0` や`/dev/vmem1` 等が移動元の VM メモリ領域となる．

5　実装のステータス

KVM-88 を対象に提案機構のプロトタイプを実装した．KVM においてはユーザランドのコードに対して，約 200 行の変更を加えている．変更点は主

[*4] バックグラウンドコピーにおける重要領域からの先行コピー機能は，本稿執筆時点においては実装作業中である．

に，ポストコピー型マイグレーションを有効にした場合に，前述した mmap() 対象を変更する点と，ライブマイグレーションにおいて VM メモリ領域のコピーを省略する点である．いずれの変更も本質的には単純な条件分岐を追加するのみであり，既存のコードをほとんど変更しない．

VMEM ドライバのコードは約 500 行であり，ページフォルトハンドラの記述が中心となる．VMEM プロセスでは，移動元ホストに対するページ取得リクエストに NBD（Network Block Device）プロトコルを用いており，我々の先行発表 [7] であるストレージマイグレーションの実装 [8] を一部利用している．

Linux および Windows 7 (RC Build 7100) をゲスト OS として，ポストコピー型ライブマイグレーションの動作を確認している．例えば，ゲスト OS の Windows 7 上で Youtube の動画を再生しながら，数百ミリ秒で異なる物理ホストへ VM を移動できる．実行ホストの切り替え直後には，遠隔のメモリページ取得が大量に発生するものの，Gigabit Ethernet の LAN 環境では，もたつくことなくスムーズに再生できている．

実行ホスト切り替え時点で，移動先ホストに転送するデータは約 8MB であり，大半を VGA デバイスの状態が占める．VGA デバイスがなければ，256KB 程度まで小さくできる．

6 評価実験

提案手法の性能を検証すべく，評価実験を行った．ウェブサーバのベンチマークである SPECweb2005 [9] を動作させながら，ライブマイグレーションを行った．実験環境を**図** 4 に示す．移動元・移動先の物理ホスト[*5]間はそれぞれ Gigabit Ethernet の LAN に接続されている．データセンタのプライベートネットワークを想定したネットワークには，移動元・移動先双方のホストからアクセスできる共有ストレージを用意し，マイグレーション前後に仮想ディスクアクセスを継続できるようにしている．また，SPECweb においてデータベースの振る舞いをエミュレーションする

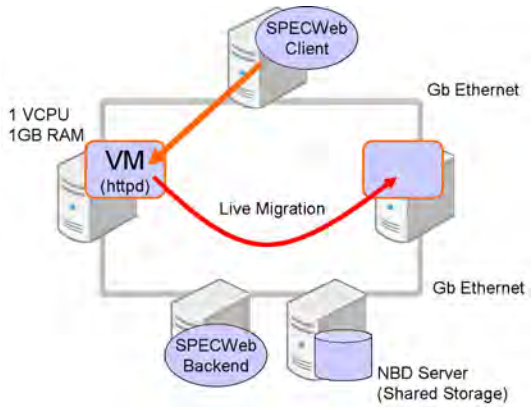

図 4　実験環境

バックエンドシミュレータノードも設置している．VM[*6]上では Apache ウェブサーバを動作させ，パブリックネットワークを想定したネットワークから SPECweb クライアントで接続する．

6.1 提案機構

SPECweb クライアントからの同時接続数を 200 と設定して，インターネットバンキングのウェブサイトをシミュレーションするベンチマーク（Banking）を走らせた．開始後 150 秒付近でライブマイグレーションを行い，実行ホストを切り替えている．ポストコピー型の実行結果を**図** 5 から**図** 9 までに示す．この実験ではオンデマンド取得のみ動作させバックグラウンドコピーは無効にしている．

リクエスト応答時間の時間遷移（図 5）等が示すように，実行ホストの切り替え自体は 1 秒以内で完了している．しかし，切り替え直後には，リクエスト応答時間が悪化している．ページ転送数の時間遷移が示すように（**図** 8），移動元からのページ取得が大量に発生している．その後 10 秒程度経過すると，ページ取得数が緩やかになり始め，実行ホスト切り替え前程ではないものの，リクエスト応答時間が改善されてくる．

図 6 が示す QoS の時間遷移とは，インターネットバンキングを利用しているユーザが，どの程度ストレスを感じるかを示している．実行ホスト切り替え後は，応答時間の悪化によって，ユーザがストレスを感じる可能性がある．しかし，この実験の同時

[*5] Intel Core 2 Duo E6305, 4GB RAM

[*6] 1 CPU Core, 1GB RAM

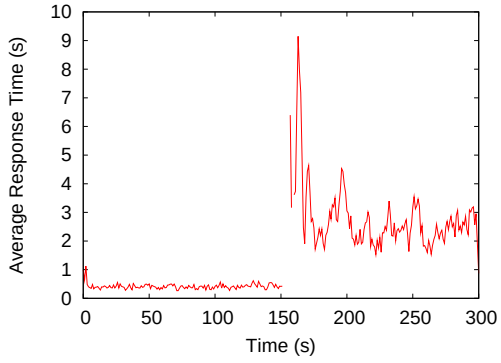

図 5　リクエスト応答時間（提案機構）

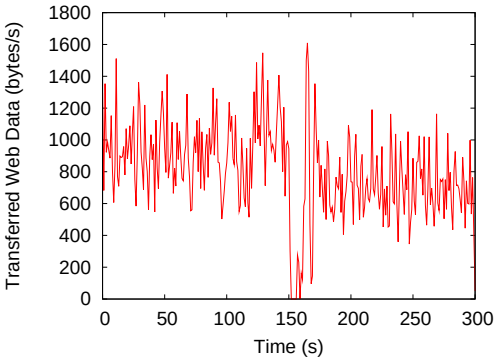

図 7　ウェブサーバ・スループット（提案機構）

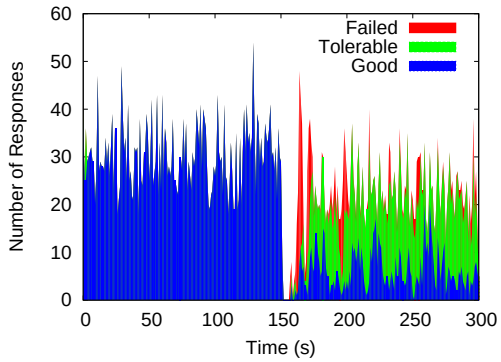

図 6　QoS（提案機構）．150 秒付近以前はほぼ Good（青）が占める．150 秒経過後はグラフ下部より Good，Tolerable（緑），Failed（赤）の順で描かれている．

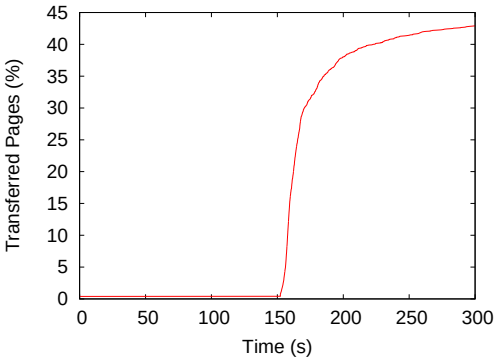

図 8　移動元からのページ取得数（提案機構）

接続数においては，使い物にならないほど悪化しているわけではない．一般的に VM のメモリアクセスが頻繁ではないほど，実行ホスト切り替え後の性能低下は軽微になるはずである．つまり，今後ポストコピー型ライブマイグレーションを VM の再配置機構に用いる際には，メモリアクセスが頻繁ではない VM を優先して再配置することが検討に値すると考えている．

取得ページオフセットの時間遷移（図 9）が示すように，必ずしもすべてのメモリ領域が実行ホスト切り替え後即座に必要になるわけではない．重要なメモリ領域を判別して，それらをページフォルトよりも前にコピーできれば，実行ホスト切り替え後の性能低下を抑制できると考えれる．

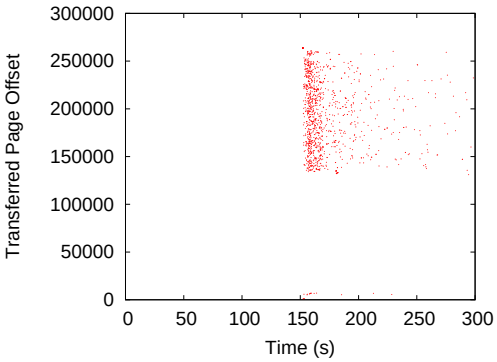

図 9　取得ページオフセット（提案機構）

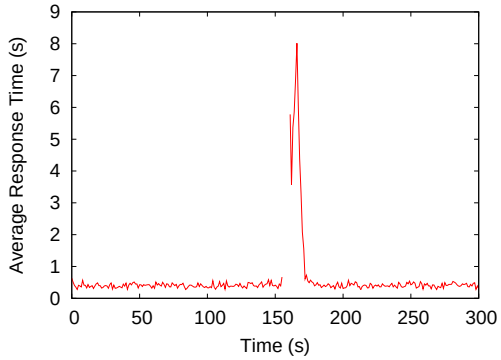

図 10　リクエスト応答時間（提案機構，バックグラウンドコピー有効）

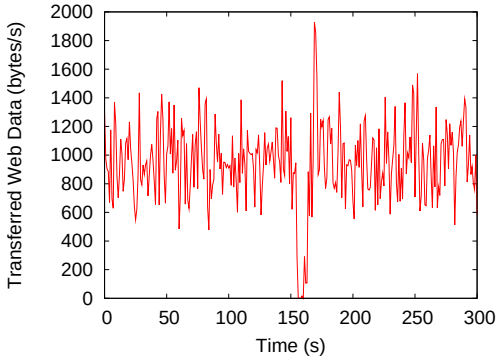

図 12　ウェブサーバ・スループット（提案機構，バックグラウンドコピー有効）

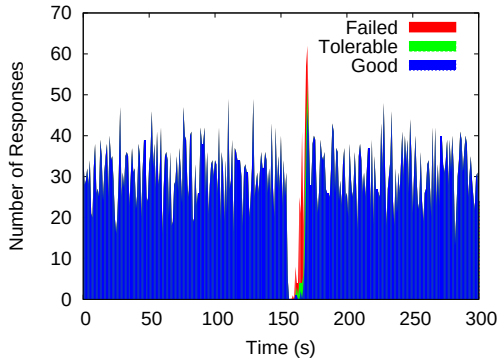

図 11　QoS（提案機構，バックグラウンドコピー有効）．150 秒直後では Tolerable（緑）および Failed（赤）が出現している．それ以外の時刻には Good（青）のみ出現している．

6.2　バックグラウンドコピーの効果

次に，提案機構のバックグラウンドコピー機能を有効にして，再度実験を行った．実行ホストを切り替えて 5 秒経過した後から，バックグラウンドコピーを 800Mbps の転送速度を設定して実行した．メモリ領域の先頭からコピーしており，10 秒程ですべてのページの転送を完了した．**図** 10 から**図** 12 が示すように，バックグラウンドコピー完了後はすべてのメモリページが移動先ホストにキャッシュされることになるため，性能低下が発生していない．今後，実行ホスト切り替え直後からバックグラウンドコピーを自動的に開始し，またページフォルトが頻出している領域からコピーすることで，一時的な性能低下期間をさらに短くできると考えている．

6.3　既存機構

KVM 本来のプレコピー型のライブマイグレーション機能を利用した場合を**図** 13 および**図** 14 に示す．開始後 150 秒付近でライブマイグレーションを開始した．プライベートネットワークの通信量（図 14）が示すように，この実験中にはマイグレーションが完了することはなかった．KVM のプレコピー型マイグレーションの実装においては，マイグレーション開始後，移動元で更新されたページを繰り返し移動先に転送する．そして，残りページが一定以下になれば，移動元で VM を停止し残りのページを CPU やデバイスの状態と共に移動先へ転送する．しかし，KVM は実行ホスト切り替え前後のダウンタイムを一定時間以下に抑えるため，残りのページ数が残りわずか[*7]になるまでは，移動元で VM を停止することはない．このベンチマークにおいては，比較的メモリの更新が頻繁であったため，残りのページ数が一定以下に減少することがなく，有限時間内においてマイグレーションを完了できなかった．

7　まとめ

本稿では，迅速な実行ホストの切り替えを可能とする新たなライブマイグレーション機構を提案した．VM 内で実行中のワークロードに関係なく，常に 1 秒以内で実行ホストを切り替えられる．VM

[*7] デフォルトの設定では転送レートから換算してダウンタイムが 3 秒以内となるページ数である．

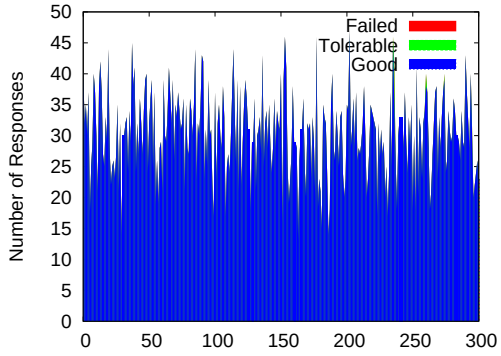

図 13　QoS（既存機構）．Good（青）のみ出現している．

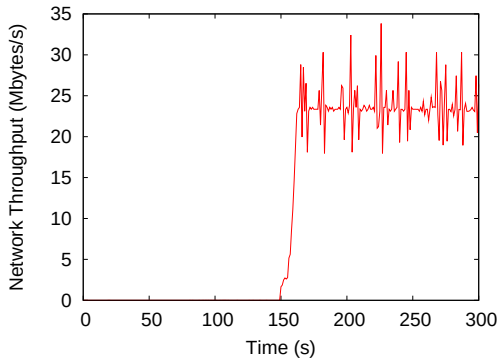

図 14　プライベートネットワーク通信量（既存機構）

のメモリ領域を特殊なデバイスファイルにマップすることで，VMM をほとんど変更することなく利用できる．またゲスト OS を一切変更する必要もない．先行研究と比べて，実用化が容易であり近い将来データセンタ等に導入可能な技術である．十分なネットワーク帯域が存在する LAN 環境においては，実行ホスト切り替え後の一時的な性能低下期間も短く，ワークロードへの影響も小さい．

今後は一時的な性能低下期間をさらに短縮すべく，VM のメモリアクセスパターンに応じた先行キャッシュ機能を開発する．また，提案機構を利用した VM の動的再配置アルゴリズムを検討していく．

本研究は科研費（20700038）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）の助成を受けたものである．

参考文献

[1] Amazon Elastic Compute Cloud. `http://aws.amazon.com/ec2`.

[2] Hidemoto Nakada, Takahiro Hirofuchi, Hirotaka Ogawa, and Satoshi Itoh. Toward virtual machine packing optimization based on genetic algorithm. In *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living (Proceedings of International Symposium on Distributed Computing and Artificial Intelligence 2009)*, Vol. 5518 of *Lecture Notes in Computer Science*, pp. 651–654. Springer, Jun 2009.

[3] 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 conference on Symposium on Networked Systems Design and Implementation*, pp. 273–286. USENIX Association, 2005.

[4] 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.

[5] 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.

[6] 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.

[7] 広渕崇宏, 小川宏高, 中田秀基, 伊藤智, 関口智嗣. 仮想計算機遠隔ライブマイグレーションのための透過的なストレージ再配置機構. 情報処理学会論文誌:コンピューティングシステム, Vol. ACS26, pp. 152–165, Jul 2009.

[8] Takahiro Hirofuchi. xNBD. `http://bitbucket.org/hirofuchi/xnbd/`.

[9] Standard Performance Evaluation Corporation. SPECweb2005. `http://www.spec.org/web2005/`.

[10] 広渕崇宏, 中田秀基, 伊藤智, 関口智嗣. 仮想計算機メモリの遅延再配置による高速ライブマイグレーション. 情報処理学会研究報告 (2009-OS-112). 情報処理学会, Jul 2009.