Vol.2009-OS-112 No.3
2009/8/5

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

広渕崇宏[†1]　中田秀基[†1]
伊藤智[†1]　関口智嗣[†1]

仮想計算機（VM）のライブマイグレーション機能は有用であるものの，負荷バランスのため広く利用されているとは言い難い．一般的なライブマイグレーション手法は，VM のメモリをすべて転送してから実行ホストを切り替えるので時間がかかる．結果，負荷バランスにも長時間を要してしまい，過負荷状態をすぐに解消できない．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 メモリを実行ホスト切り替え後からオンデマンドに転送することで，約 1 秒程度で稼動ホストの変更を実現できる．既存の実装と比較して，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．プロトタイプを実装してオンデマンドなメモリアクセスが正しく動作することを確認した．

Rapid Virtual Machine Relocation Based on Delayed Memory Transfer

Takahiro Hirofuchi ,[†1] Hidemoto Nakada ,[†1]
Satoshi Itoh [†1] and Satoshi Sekiguchi [†1]

Live migration of virtual machines (VMs) is a potential key technology for next-generation datacenters, which enables dynamic load-balancing of VMs for optimizing power consumptions and performance. Existing live migration mechanisms, however, are not suitable for dynamic load-balancing. They cannot quickly resolve overloaded states of physical hosts; because all memory pages need to be transferred before a migrating VM is restarted at a destination host. In this paper, we propose an advanced live migration mechanism which enables very quick (approximately 1 second) switching of a host node. The key of this mechanism is a delayed memory transfer after execution of a host change. Our prototype implementation supports any guest operating systems without any modification to them. It only needs small modification to a virtual machine monitor. We confirmed that the on-demand memory access mechanism of our implementation worked correctly for guest operating systems of Linux and Windows 7.

1. はじめに

今日データセンタの省電力化と利用効率向上が求められており，仮想計算機技術を導入してサーバ計算機を集約する機構に注目が集まっている．仮想計算機モニタ（VMM）によって物理計算機を論理的に分割・共有することで，一つの物理計算機上で複数のサーバ実行環境を稼動できる．より多くの仮想計算機（VM）を物理計算機上で稼動できれば，必要な物理資源を減らすことができ，更なる省電力化とコスト削減が可能になる．

多くの VMM はライブマイグレーション機能を備えており，ゲスト OS を起動したまま VM 実行ホストを変更できる．そこで，既存研究においては，VM の負荷に応じて物理計算機クラスタ上の VM 配置を最適化する手法が提案されている．VM の CPU 負荷が低く物理計算機の処理性能に余裕があれば，一つの物理計算機上に，より多くの VM を集約する．また，ある VM の CPU 負荷が上昇した際には，その VM が物理計算機の処理能力の多くを使用できるように，VM の配置を分散させる．

しかし，このようなライブマイグレーションを利用した動的な負荷分散機能は，Amazon EC2[1)] 等の VM ホスティングサービスにおいて広く用いられているとは言い難い．物理計算機の資源を静的な割合で VM に割り当てるのみである．既存のライブマイグレーション機構は，動作中の VM のメモリイメージを移動先ホストに完全に転送してから，稼動ホストを切り替える．そのため，再配置中は，ゲスト OS が書き込んだメモリページを常に記録する必要があり，そのオーバヘッドにより性能低下が発生してしまう．また，すべてのメモリイメージを移動先ホストに転送するのに時間を要する．VM の CPU 負荷上昇に反応して，即座に VM 配置を変更することが難しい．つまり，現状，ライブマイグレーションを利用した負荷分散機能は，VM の処理性能に対して一定の目安や保証を与える VM ホスティングサービスにおいて，導入しがたいという問題がある．

そこで，本研究では，新たにメモリコピー後行型のライブマイグレーション機構を開発し，その迅速な VM 移動機能を利用した，VM 配置の最適化手法に取り組む．メモリコピー後

†1 産業技術総合研究所 / National Institute of Advanced Industrial Science and Technology（AIST）

行型のライブマイグレーションとは，CPU 状態等の最小限の情報のみを移動先ホストにコピーして VM の実行を再開する手法である．実行再開後に VM がはじめてアクセスしたメモリページ内容はオンデマンドに移動元ホストから取得する．従来のメモリコピー先行型のライブマイグレーションと比較して，極めて迅速に実行ホストを切り替えることができる．CPU 負荷に応じた動的な負荷分散に適しているはずであり，VM の処理性能を維持しながらも VM の集約率を高められる可能性がある．

本稿では，研究の第一段階として，メモリコピー後行型のライブマイグレーション機構の設計について述べる．今日広く使用されている VMM の一つである KVM[5] に対してプロトタイプを実装した．既存のメモリコピー後行型のライブマイグレーション機構と比較して，VMM に対する変更が小さく実用化が容易である．ホスト OS のカーネルドライバとその補助プログラムにより実装しており，ゲスト OS に対して一切手を加える必要がない．KVM が対応する Linux や Windows などさまざまな OS に対して利用できる．

第 2 節でメモリコピー後行型のライブマイグレーションが必要とされる理由を述べ，第 3 節でメモリコピー後行型に関する既存研究をまとめる．第 4 節で我々の実装手法を提案し，第 5 節でその動作確認についてのべる．第 6 節でまとめる．

2. ライブマイグレーションの仕組みと VM 配置最適化

本節では，はじめにライブマイグレーションの仕組みについて説明する．そして，メモリコピー後行型のライブマイグレーションが仮想化データセンタにおいて必要である理由について述べる．

2.1 メモリコピー先行型

今日広く使用されている VMM（Xen[2]，KVM や VMware ESX 等）で実装されているライブマイグレーション機構は，VM のメモリイメージを移動先ホストにすべてコピーしてから，実行ホストを移動先ホストに切り替えている．実行ホストの切り替えよりも前にメモリを完全にコピーすることから，本稿ではこの機構をメモリコピー先行型のライブマイグレーションと呼ぶ．VM を稼動させたままメモリイメージのコピーを行うので，コピー中に新たに変更されたメモリページを再度転送することを繰り返す．最終的にほぼすべてのメモリページが再現された段階，あるいは更新ページの再帰的コピー回数が上限値に達した段階で，移動元ホストの VM を停止する．そして，残りのメモリページと CPU 状態等を転送し，移動先ホストで実行を再開する．メモリの再帰コピー中には変更ページを常時記録する必要があり性能低下が起きる可能性がある．

メモリコピー先行型においてライブマイグレーションを開始してから完了するまでの時間は，VM のメモリサイズとネットワーク帯域，およびメモリページの更新量に大きく依存する．仮に VM のメモリサイズが 1GB であれば 100Mbps のネットワークを介した場合，少なくとも 80 秒以上かかる計算になる．実際には更新ページを再帰的に転送するため更に時間が長くなる．

2.2 メモリコピー後行型

一方，本稿でメモリコピー後行型と呼んでいるライブマイグレーションにおいては，VM を移動元ホストで停止しメモリページ以外の状態を移動先ホストに転送する．このとき停止した VM のメモリイメージは移動元ホストに残しておく．そして，VM の実行を移動先ホストで再開し，VM にアクセスされたメモリページを移動元ホストから取得しながら実行を進めていく．さらに，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のコピーにより残りのメモリページを移動先ホストに転送していく．最終的にすべてのメモリページを移動先ホストに転送できた時点でライブマイグレーションは完了し，VM は移動元ホストに依存しなくなる．メモリページの転送にはネットワーク遅延がともなうので，オンデマンドなメモリ転送のみでは VM の性能低下が懸念される．しかし，バックグラウンドコピーにより重要メモリページを VM によるアクセスよりも先行して取得することで，性能低下を抑止できる．我々は VM が使用するストレージの再配置機構を過去に提案[10] しており，その中で同様の仕組みを用いた．ストレージの再配置機構ではバックグラウンドコピーが性能低下軽減に貢献している．

ライブマイグレーションを開始してから実行ホストの切り替えまでにかかる時間は，メモリコピー先行型に比べて大幅に短縮でき，ほぼ一秒以内に完了する．実行ホスト切り替えのために転送するデータは，CPU や周辺機器の状態のみであり，我々のプロトタイプ実装においては，最小の構成であれば，わずか 140KB である．

2.3 VM 配置最適化機構への適用

VM の再配置機構は，複数の VM をできる限り少ない台数の物理計算機上に，各 VM の性能要件を満たすように配置しなければならない．VM に対するワークロード負荷が上昇したことにより，物理計算機が過負荷状態になれば，配置状態を変更して過負荷状態を解消する必要がある．

例えば，物理計算機 1 ノードの CPU 資源の 50%を，ひとつの VM に対して割り当てる最大 CPU 資源であるとする．VM に対する負荷が低く物理計算機の CPU 資源を 10%しか消費しないのであれば，10 台の VM を一つの物理計算機上で稼動できる．このとき，あ

る VM の負荷が急激に上昇し VM に割り当てた 100%の CPU 資源（物理計算機の 50%の CPU 資源）を消費しようとすると，物理計算機は過負荷な状態となってしまうので，すぐさま計算機クラスタ上での VM 配置状態を変更する必要が生じる．

2.3.1 メモリコピー先行型再配置の問題点

既存の配置状態変更手法においては，先行型のメモリ再配置を用いているので，過負荷状態の解消までに時間がかかってしまっている．例えば，Sandpiper[8] では，過負荷状態を検知する際に VM 内部のゲスト OS 上で行う場合と，ゲスト OS には一切手を加えずにホスト OS からのみから行う場合について比較している．ゲスト OS 内部で過負荷状態を検知した方がよい結果が得られている点が興味深い．しかし，配置状態の変更に対して，2 回のマイグレーションをともなう場合で約 40 秒の時間を要している．その間，過負荷状態の物理計算機上の VM は，十分な CPU 資源を確保できないという問題がある．また，Entropy[3] は，2 つの制約問題に対する解を求めることで，負荷に応じて VM 配置状態を変更する．ひとつは，各 VM が要求する CPU 資源とメモリ使用量に関する各 VM の要求をすべて満たす，最小の物理計算機の台数を求める問題である．もう一方は，マイグレーションによって配置状態を変更する際のコストを最小にする問題である．ワークロードの変化に応じた負荷バランスに成功しているものの，ある実験では 10 台の過負荷状態の VM を解消するために 6 分程度要してしまっている．

2.3.2 メモリコピー後行型再配置の必要性

しかし，我々が想定する仮想化データセンタにおいては，各 VM は顧客に対して提供されるものであり，その性能は常に何らかの形で担保されている必要がある．ある VM の負荷が上昇したことによって，物理計算機が過負荷状態になれば，VM 配置状態を変更することで，すぐさま解消する必要がある．既存手法で用いているメモリコピー先行型のライブマイグレーションでは，VM 移動に時間がかかってしまい我々の想定環境への対応が難しい．一方，メモリコピー後行型であれば，極めて迅速に実行ホストを切り替えられる．負荷変動に応じてきめ細かく配置状態を変更することで，顧客の VM の処理性能を維持できる可能性がある．

メモリコピー後行型の再配置手法では，前述したように実行ホスト切り替え後に性能低下が生じる可能性がある．ワークロード負荷が上昇した VM は，メモリアクセスも頻繁に発生するはずなので，実行ホストを切り替えてしまうと性能低下が大きいと予想される．しかし，代わりにワークロード負荷が低い VM を動かせば，実行ホスト切り替え後のメモリアクセスも少ないままであり，バックグラウンドコピーによる事前キャッシュが有効に働き，性能低下を抑止できるのではと考えている．またワークロード負荷が低い VM にとっては，一時的にメモリアクセス遅延が大きくなったとしても，アプリケーションの性能に対する影響は小さく看過できる可能性がある．

以上のように，後行型の再配置手法には十分な将来性が存在し，その有用性は仮想化データセンタの VM 配置最適化に対して実際の実装を通して検証されなければならない．そこで，我々は後行型の再配置手法のプロトタイプ実装を進めている．

3. 関連研究

メモリコピー後行型のライブマイグレーション手法について関連研究をまとめる．Snowflock[6] は，動作中の VM を他の物理計算機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，メモリコピー後行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 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，Xen の準仮想化モードに対してメモリコピー後行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

⋆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．

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，実際に利用できるメモリコピー後行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

以上に述べたように，メモリコピー後行型のライブマイグレーション機能は，仮想化データセンタ内のVM配置最適化において有用であるものの，現状利用できる実装は存在しない．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メモリコピー後行型ライブマイグレーション機構を提案する．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ブロックデバイスを実装する．

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．メモリコピー先行型のライブマイグレーションに対応している．

仮想マシンを起動すると，QEMUプロセスはユーザランドメモリ空間に仮想マシン用のメモリ領域を確保する．KVMドライバに対して仮想CPUの割り当てを要請し，また確保したメモリ領域を通知する．エミュレートするハードウェアの初期化等が完了すると，KVMドライバに対してVMの実行を命令する．KVMはShadow Page Table（SPT）を作成して，VMのメモリアクセスをエミュレートする．

x86アーキテクチャにおいては，プロテクトモードに移行したOSは，仮想アドレスでメモリアクセスするために，仮想アドレスと物理アドレスの変換テーブル（ページテーブル）を作成する．しかし，ゲストOSのページテーブル中の物理アドレスは，ゲストOSが本物の物理計算機上で動いていると思い込んで設定した物理アドレスであり，そのままでは利用できない．そこで，ゲストOSのページテーブルに対応したSPTをKVMが用意し，ゲストOS実行時にはSPTを利用してアドレッシングを行う．KVMは最初空のSPTを用意する．ゲストOSがアクセスした仮想アドレスに対するSPTエントリが存在しなければ，ページフォルトによりVMの実行が停止（VM Exit）する．KVMはVM Exitの原因を調べて，SPTエントリの不在によるページフォルトが理由であれば，SPTエントリを追加して，VMの実行を再開する．一度SPTエントリが追加されると，次回対応する仮想アドレスにアクセスした際にはページフォルトが発生しない．ゲストOSがコンテキストスイッチ等により新たなページテーブルに切り替えると，SPTは再び空になり再度SPTエントリを追加する必要が生じる[⋆1]．

4.2 メモリコピー後行型ライブマイグレーション機構

我々はKVMに対して2種類の後行メモリコピー型のVM再配置機構を試験的に実装した．SPT実装を拡張する方式とVMメモリ領域を特殊なブロックデバイスにマップする方式である．本稿では，前者については簡単な説明にとどめ，後者を中心に取り上げる．どちらもKVMに対する変更量は小さく性能面での差もほとんどないと予想される．しかし，前者は，KVMにおいて最も重要な処理であるメモリアクセスのエミュレーション部分のコードに手を入れる必要があり，我々が念頭においている実環境に適用するためには，動作検証作業により多くの時間を要すると判断した．

4.2.1 SPT実装を拡張する方式

この方式では，仮想計算再配置後のメモリエミュレーション処理に遠隔メモリページの取得処理を追加する．上述したように，VMがメモリアクセスするためには，対応するSPTエントリが作成されておらねばならず，作成されていなければページフォルトが発生する．そこで，ページフォルト発生時にVMがアクセスしようとした（VMが本物だと思い込ん

⋆1 最近のKVMの実装ではSPTエントリを空にせずにキャッシュしている．

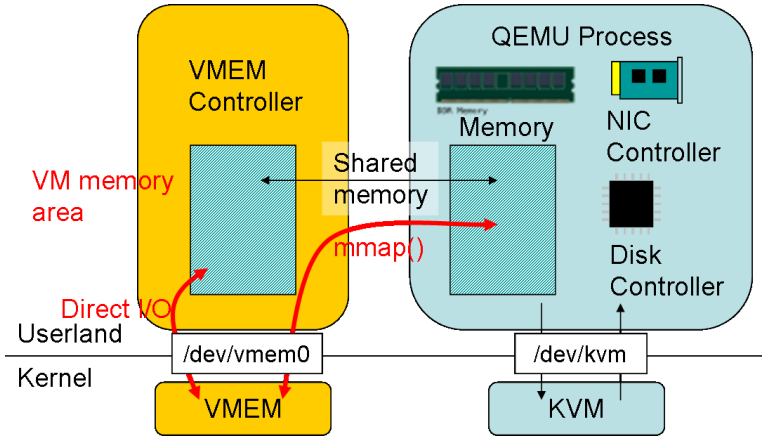

図 1 提案機構の概要

でいる）物理アドレスを調べ，未転送であれば対応するメモリページを移動元ホストから転送することで，オンデマンドなメモリページ取得を実現する．メモリページの転送には，ユーザランドの補助プログラムを使用している．また，エミュレーションしているデバイスが VM のメモリを直接アクセスする場合もある．そこで，QEMU がエミュレーションデバイス用に提供しているメモリアクセス用関数の内部を拡張して，必要があればメモリページを移動元ホストから転送するようにしている．試験的な実装においてオンデマンドなメモリ取得部分の基本的な動作を確認しているものの，先に述べた理由により次の方式を本稿では取り上げる．

4.2.2 VM メモリ領域を専用ブロックデバイスにマップする方式

VM メモリ領域を専用ブロックデバイスにマップする方式の概要を**図 1** に示す．本稿ではこの機構を VMEM とよぶ．ホスト OS に対して我々が開発したドライバ（`vmem.ko`）をロードすることで，専用キャラクタデバイス（`/dev/vmem0`）を作成する．また補助プログラムプロセス（VMEM プロセス）を実行することで，`/dev/vmem0` に対してメモリを割り当てる．VMEM プロセスはユーザ空間で VM 用のメモリを確保し，Direct I/O 機能によりカーネルと共有する．このメモリ空間は`/dev/vmem0` を `mmap()` したプロセスと共有される．

KVM は，仮想マシンのメモリ領域を 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 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．結果，QEMU プロセスと VMEM プロセスは VM メモリ領域を共有しており，VMEM プロセスは常に VM メモリ領域を直接読み書きできる．

VMEM には，`mmap()` した領域へのメモリアクセスに関して，2 つの動作モードが存在する．一般に，`mmap()` したメモリ領域は，プロセスが初めて実際にアクセスした時に，アクセス対象となるメモリページが割り当てられる．ページフォルトが発生してドライバが用意したページフォルトハンドラがカーネルに呼ばれる．

VM が移動元計算機で動作している場合に用いられるモードでは，あらかじめすべてのメモリ領域が割り当てられている．QEMU プロセスが VM メモリ領域中のどのメモリページにアクセスしても，VMEM ドライバのページフォルトハンドラが呼ばれることはない．

一方，VM が実行ホストを切り替えた後に利用されるモードでは，初期状態においては QEMU プロセスに対していずれの VM メモリページも割り当てられていない．QEMU プロセスがメモリページに初めてアクセスすると，VMEM ドライバのページフォルトハンドラが，QEMU プロセスの実行コンテキストでカーネルにより呼び出される．ページフォルトハンドラは QEMU プロセスの実行を一時停止して，VMEM プロセスに移動元ホストからメモリページ内容を取得するよう要請する．VMEM プロセスは取得したメモリページを自身のプロセス空間中の共有メモリ領域に書き込み，VMEM ドライバに完了を通知する．VMEM ドライバは停止していた QEMU プロセスの実行を再開する．

メモリコピー後行型ライブマイグレーションの全体動作を**図 2** 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) 移動元計算機で 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，移動先計算機に転送し QEMU プロセス中にロードする．そして移動先計算機で VM の実行を再開する．
(2) VM がメモリにアクセスする．
(3) もしアクセスしたメモリページが移動元ホストから未取得のページであれば，VMEM 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．

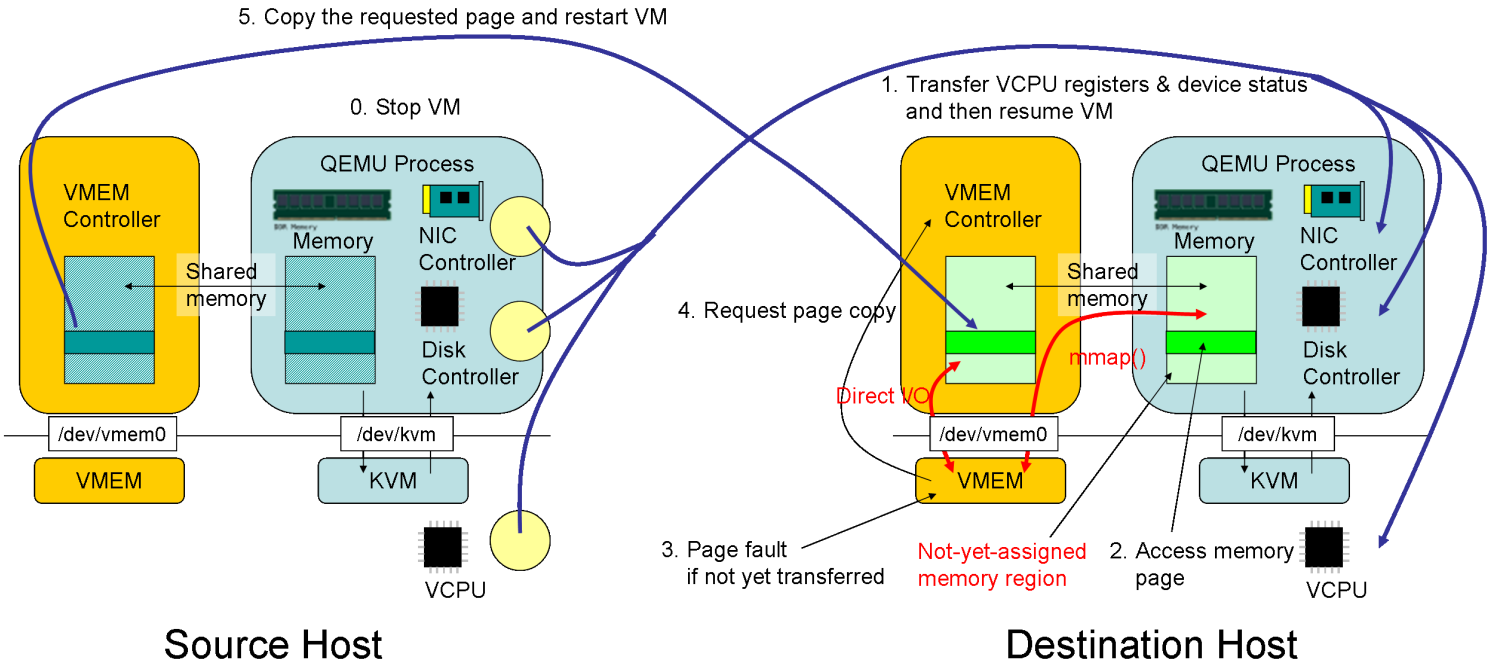

図 2 再配置動作の概要

(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．

最終的にすべてのメモリページが転送されると，移動元ホストからの VM 移動が完了する．以上に述べた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．迅速に VM メモリ領域全体を再配置して移動先のホスト単体で VM を稼動できるようにする．またバックグラウンドのメモリ取得においては，VM の性能低下を抑制するために，重要メモリページから優先的にコピーする予定である．

5. 動作確認

メモリコピー後行型ライブマイグレーションにおけるオンデマンドなメモリページロード部分の動作を確認した．提案機構におけるネットワーク転送処理を省略し，その代わりにホスト OS 上のファイルからメモリページをオンデマンドに読み込んだ．動作中の VM を停止し，その時点の CPU レジスタやデバイス状態，および VM メモリ領域の内容をそれぞれファイルに保存する．そして，オンデマンドメモリアクセスを有効にして，保存した CPU レジスタとデバイス状態から VM を復帰させた．[⋆1]

図 3 に，VM 復帰開始からのメモリページ読み込み数の変化を示す．3 本のグラフはそれぞれ以下に相当する．

- **Linux** Debian/Lenny を最小構成でインストールした VM．システムのデーモンプロセスやコンソールのシェル以外に実行タスクなし．
- **Linux with running jobs** 同上の VM．CPU インテンシブなタスク[⋆2]を実行中．
- **Windows 7** Windows 7 RC Build 7100 をインストールした VM．ユーザがログオンした状態．

Linux および Windows 7 とも我々のプロトタイプ実装は正しく動作した．いずれの VM も約 1 秒後には実行画面が復帰した．Linux においてはその直後から操作を受け付けるようになった．また Windows 7 においても若干もたつくものの数秒後から操作可能になった．Linux においては，おおよそ 5500 ページ（約 21MB）[⋆3]のメモリをロードした時点で，VM の実行が再び始まっている．オンデマンドなメモリアクセスの仕組みがなければ，VM の搭載メモリが 512MB の場合，実行再開時に 131072 ページものメモリを読み込まなければならない．しかし，再開時にすべてのメモリページを一度に読み込むのではなく，ゲスト OS がアクセスしたメモリから徐々に読み込んでいくことで，復帰時間を大幅に短縮できている．

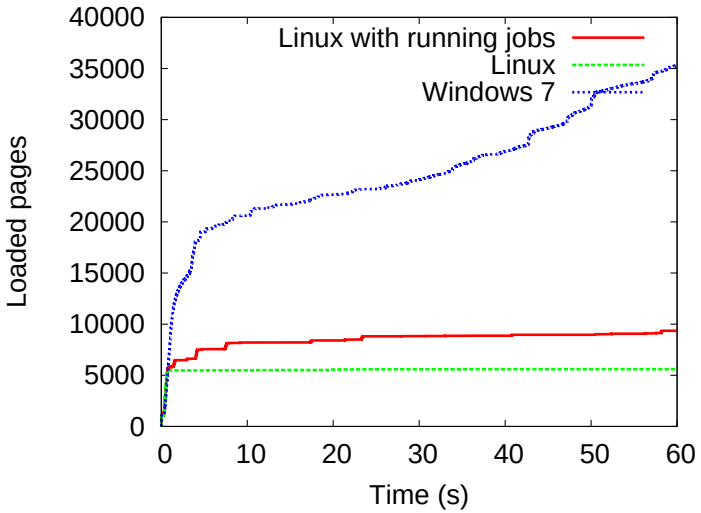

図 3 メモリページ読み込み数の時間遷移

Windows 7 は 10 秒程度でメモリページの読み込みが緩やかになっている．Linux に比べれば実行中のタスクが多いため，読み込まれるメモリも大きい．60 秒後には約 35000 ページ（約 140MB）ものメモリが読み込まれている．しかし，VM のメモリ領域全体に比べれば 27%程度であり，すべてのメモリページが復帰直後から必要とされていないことがわかる．

6. まとめ

VM のライブマイグレーションによって，ゲスト OS を稼動させたまま VM 実行ホストを切り替えらる．仮想化データセンタにおいて動的な負荷バランスに利用でき，計算機クラスタ全体の利用効率を高めながら省電力化もはかれる．負荷が低い VM を同一ホスト上に集約して余った計算機を停止でき，VM の CPU 負荷が突然上昇したときには配置状態を変更して処理性能を維持できる．

⋆1 ホスト計算機: Intel Core2 6300，2GB RAM
VM: 1CPU，512MB RAM
⋆2 2 の 1234567890 乗を bc コマンドで計算している．
⋆3 フレームバッファと BIOS 用メモリ領域が 20MB を占めている．VGA デバイスを VM から除外すれば，さらに削減できる．

しかし，現在利用できるライブマイグレーション機構は，VM のメモリをすべて移動先ホストに転送してから実行ホストを切り替えてしまう．実行ホストの切り替えに時間がかかるので，負荷バランスに用いても過負荷状態の解消に時間を要してしまう．顧客の VM の性能低下を招いてしまうので，ホスティングサービスプロバイダでの適用を困難にしている．

そこで，本稿では，迅速に実行ホストを切り替えられるライブマイグレーション機構を提案した．VM のメモリを実行ホスト切り替え後からオンデマンドに転送する．大量のメモリを割り当てた VM であっても，約 1 秒程度で実行ホストの切り替えを実現できる．競合する実装と比較しても，VMM への改変が少なくゲスト OS への変更も不要である点に優位性がある．

プロトタイプ実装により，オンデマンドなメモリアクセス機構の動作確認を行った．Linux および Windows 7 をインストールした VM において，約 1 秒程度で VM を復帰できる．今後，ネットワーク転送やバックグラウンドコピーの処理を追加することで，ライブマイグレーションとして動作するように実装を進める．また，我々の仮想クラスタ管理システム[9),11)] や VM 配置決定機構[7)] に統合したい．

謝辞 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) Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I. and Warfield, A.: Xen and the art of virtualization, *Proceedings of the nineteenth ACM symposium on Operating systems principles*, ACM Press (2003).
3) Hermenier, F., Lorca, X., Menaud, J.-M., Muller, G. and Lawall, J.L.: Entropy: a consolidation manager for clusters, *Proceedings of the 5th International Conference on Virtual Execution Environments*, ACM Press, pp.41–50 (2009).
4) Hines, M.R. and Gopalan, K.: Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning, *Proceedings of the 5th International Conference on Virtual Execution Environments*, ACM Press, pp.51–60 (2009).
5) Kivity, A., Kamay, Y., Laor, D. and Liguori, A.: kvm: the Linux Virtual Machine Monitor, *Proceedings of the Linux Symposium*, The Linux Symposium, pp.225–230 (2007).
6) Lagar-Cavilla, H.A., Whitney, J.A., Scannell, A., Patchin, P., Rumble, S.M., de Lara, E., Brudno, M. and Satyanarayanan, M.: SnowFlock: Rapid Virtual Machine Cloning for Cloud Computing, *Proceedings of the fourth ACM european conference on Computer systems*, ACM Press, pp.1–12 (2009).
7) Nakada, H., Hirofuchi, T., Ogawa, H. and Itoh, S.: Toward Virtual Machine Packing Optimization Based on Genetic Algorithm, *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living (Proceedings of International Symposium on Distributed Computing and Artificial Intelligence 2009)*, Lecture Notes in Computer Science, Vol.5518, Springer, pp.651–654 (2009).
8) Wood, T., Shenoy, P.J., Venkataramani, A. and Yousif, M.S.: Black-box and Gray-box Strategies for Virtual Machine Migration, *Proceedings of the 4th Symposium on Networked Systems Design and Implementation*, USENIX Association, pp.229–242 (2007).
9) 中田秀基，横井威，江原忠士，谷村勇輔，小川宏高，関口智嗣：仮想クラスタ管理システムの設計と実装，情報処理学会論文誌:コンピューティングシステム，Vol.ACS19, pp.13–24 (2007).
10) 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 (印刷中) (2009).
11) 広渕崇宏，中田秀基，横井威，江原忠士，谷村勇輔，小川宏高，関口智嗣：複数サイトにまたがる仮想クラスタの構築手法，第 6 回先進的計算基盤システムシンポジウム SACSIS 2008，pp.333–340 (2008).