Vol.2012-OS-123 No.9
Vol.2012-EMB-27 No.9
2012/12/5

ページキャッシュの復元による遠隔地ライブマイグレーションの高速化

穐山 空道[1]　広渕 崇宏[2]　高野 了成[2]　本位田 真一[1,3]

概要：遠隔地ライブマイグレーションによって，データセンタ間負荷分散や災害時の情報システム維持などが可能になる．しかし遠隔地ライブマイグレーションではネットワーク帯域が狭いことが課題である．ディスクイメージは比較的アクセスが遅いためリアルタイムでの同期が可能だが，メモリはマイグレーション時に転送する必要がある．我々はゲスト OS のメモリのうちページキャッシュが多くを占める場合に着目する．ページキャッシュをネットワーク越しに転送せず移動先でディスクイメージから復元することで，遠隔地ライブマイグレーションを高速化する手法を提案する．提案手法を QEMU/KVM 上に実装し，ネットワーク帯域の狭い環境下で total migration time を削減することを示した．

A Preliminary Study on Boosting Wide Area Live Migration with Page Cache Regeneration

Akiyama Soramichi[1]　Hirofuchi Takahiro[2]　Takano Ryousei[2]　Honiden Shinichi[1,3]

Abstract: Wide area live migration achieves cross-datacenter load balancing or disaster recovery of IT systems. The greatest issue of wide area live migration is the bandwidth bottleneck of wide area network (WAN). The memory of a virtual machine (VM) must be transferred via slow network while the disk image can be synchronized background as it is not updated frequently. We focus on a case that page cache occupies the large portion of the VM's memory usage. We proposes a fast wide area live migration mechanism, where the page cache is restored from the disk image at the destination and is not transferred via narrow network. Preliminary evaluations showed that our prototype reduced total migration time of live migration in a narrow bandwidth network. Future work includes considering the network delay and further analysis of the evaluation results.

1. 序論

Wide Area Network（WAN）をまたいだ遠隔地ライブマイグレーションを活用する研究が広く行われている．負荷の高まりに応じてあるデータセンタの仮想マシン (VM) を別のデータセンタへ移動させるデータセンタ間負荷分散 [1] や，発電方法による二酸化炭素排出量の違いを考慮しクリーンなデータセンタへ VM を移動する低炭素仮想プライベートクラウド [2] が提案されている．また災害発生時にデータセンタ内の VM を遠隔地の安全なデータセンタへ移動することで重要な情報システムを維持できる．

遠隔地ライブマイグレーションの課題は，WAN 環境では Local Area Network（LAN）環境に比較して通信帯域が狭い点である．LAN 環境では 1Gbps 以上の帯域が一般的であるのに対し，WAN 環境ではより狭い帯域が一般的である．

ライブマイグレーションでは VM のディスクイメージおよびメモリを移動元ホストから移動先ホストへ転送する．ディスクイメージの転送に関しては様々な研究 [1], [3], [4] が行われており，また IO mirroring [5] や DRBD [6] 等を

[1] 東京大学大学院 情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo
[2] 独立行政法人産業技術総合研究所
National Institute of Advanced Industrial Science and Technology (AIST)
[3] 国立情報学研究所
National Institute of Informatics (NII)

用いリアルタイムに同期することで転送を省略することもできる．一方でメモリはディスクに比較してアクセスが速いため，WAN 環境において移動元と移動先でリアルタイムに同期することは現実的ではない．メモリ転送量削減についても様々な研究 [7], [8], [9], [10] が行われている．しかし既存研究では VMM がゲスト OS のメモリを単なるバイト列として扱うため，ゲスト OS によるメモリの使われ方を活用した最適化が行えない．

本研究ではゲスト OS の持つメモリに関する情報を活用することで，遠隔地ライブマイグレーションを高速化する手法を提案する．VM のメモリ内に含まれる様々なデータのうち，特にページキャッシュが大きな割合を占める場合が観察される [7]．ページキャッシュは比較的低速なディスクへのアクセス速度を隠蔽するためのオンメモリキャッシュである．ページキャッシュにはディスク上に同一内容のブロックが存在する場合がある．これは読み込みキャッシュまたは書き込みキャッシュのうちディスクに書き戻された部分である．我々はディスク上に同一内容のブロックが存在するページキャッシュを，移動先でディスクイメージからゲスト OS のメモリへ復元することでライブマイグレーションを高速化する．本手法を *Page Cache Teleportation* と呼ぶ．実装では特定の種類のゲスト OS への依存を少なくすることを目標とした．データセンタのユーザは様々な OS を用いるため，特定 OS に依存しない手法がより有用であると言える．我々はページキャッシュのメモリ内での位置判定にのみ OS 依存の手法を用いた．ディスクからメモリへのページキャッシュの復元はゲスト OS に非依存，透過的に行われる．QEMU/KVM を用いて Page Cache Teleportation を実装し，web サーバ及び TPC-C を用いた評価の結果，提案手法は遠隔地ライブマイグレーションを高速化できることが示された．

本論文の構成を以下に示す．第二章では関連研究およびライブマイグレーション高速化の問題点について述べる．第三章では遠隔地ライブマイグレーションのユースケースと要件について述べ，提案手法を概観する．第四章では提案手法の実装およびシステムの動作を詳解する．第五章で提案手法が遠隔地ライブマイグレーションにおける total migration time を削減することを実験により示す．第六章でオーバーヘッドの議論および類似手法との比較を行い，第七章で本論文を結論する．

2. 関連研究

2.1 従来のライブマイグレーション高速化

ライブマイグレーションを高速化する研究は数多く行われている．KVM による完全仮想化環境では文献 [7] や文献 [8]，Xen による準仮想化環境では文献 [9] や文献 [10] が挙げられる．文献 [7] ではマイグレーション時に仮想マシンのメモリを移動元ホストに保持し，当該仮想マシンが同じホストに再度マイグレーションする際に更新されたメモリページのみを転送することでライブマイグレーションを高速化する．文献 [8] は一度転送されたメモリページを再度転送する際には差分のみを圧縮・転送することで pre-copy 型ライブマイグレーションにおける繰り返しコピーを削減する．文献 [9] はライブマイグレーション前に未使用ページを送信しないことでメモリ転送量を削減する．ゲスト OS が VMM に対し新たなメモリページを要求，あるいは不要なメモリページを返却する Xen の機構を利用する．文献 [10] ではゲスト OS のメモリ転送時に重複を排除することでライブマイグレーションを高速化する．メモリページをブロックに分割しハッシュを取ることで同一のページのみならず部分的に同一なページも検知する．本文献では web サーバやネットバンキングサービスなどの実用的なワークロードで多くのページが 80%以上の同一性を持つと報告されている．

これらの研究ではゲスト OS のメモリを単なるバイト列として扱うため，メモリのデータの内容や特性に応じた最適化が行えない．ゲスト OS はメモリスワップアウトのための LRU リストやメモリページの使われ方を示すフラグなど様々な情報を持つ．我々はこれらの情報を活用することでライブマイグレーションの更なる最適化が可能だと考える．本研究では特にメモリ内のページキャッシュに着目し，ゲスト OS の持つページキャッシュの位置情報を利用することで遠隔地ライブマイグレーションを高速化する．

2.2 ストレージマイグレーション

VM のディスクイメージを移動元から移動先へ転送することをストレージマイグレーションと呼ぶ．遠隔地ライブマイグレーションではホスト間ネットワークの信頼性が低いため NFS などの共有ストレージを用いずストレージマイグレーションを行うことが一般的である．文献 [1] は複数の VM のディスクイメージの共通部分を一度だけ転送することおよびアプリケーションの起動に必要な部分を先に転送することでストレージマイグレーションを高速化する．文献 [3] では VM の起動時にディスクイメージの転送完了を待たず，VM の動作中に必要になったブロックを on-demand に取得することでストレージマイグレーションを見掛け上高速化する．文献 [4] では VM が一度実行されたホストにディスクイメージを保持し，当該 VM がそのホストに再度マイグレーションする際に更新されたディスクブロックのみ転送する．

ディスク書き込みはメモリ書き込みに比較して遅いため，VMWare の IO mirroring [5] や DRBD [6] を用いた同期など，ディスクイメージへの書き込みをリアルタイムに同期できる．これらの手法を用ればストレージマイグレーションを行う必要はない．本研究でもディスクイメージは DRBD を用いてリアルタイムに同期されていると仮定

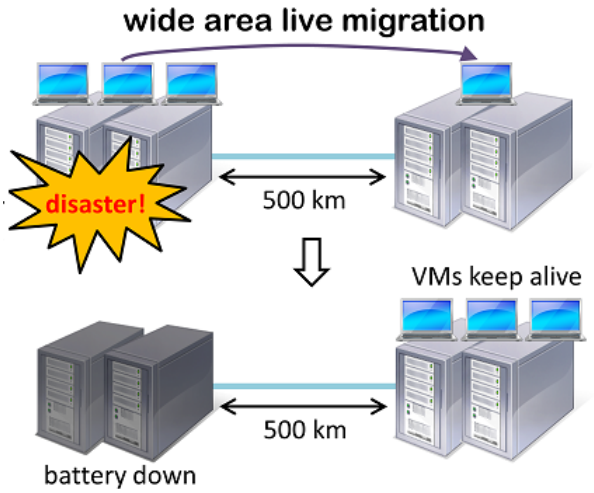

図 1 災害時の遠隔地ライブマイグレーションによるシステム維持
Fig. 1 Continuation of IT systems on a disaster by wide area live migration

する．

3. Page Cache Teleportation

3.1 遠隔地ライブマイグレーション

遠隔地ライブマイグレーションには様々な利用例がある．文献 [1] では，遠隔地ライブマイグレーションを用いてデータセンタ間負荷分散を実現する．一つのデータセンタ内でのライブマイグレーションを用いたホスト間負荷分散は様々な研究や製品で行われている．本文献では遠隔地ライブマイグレーションを用いることで，あるデータセンタの負荷を別のデータセンタへ分散することを可能にした．文献 [2] では遠隔地ライブマイグレーションによって仮想プライベートクラウドのエネルギー源のうち二酸化炭素を排出しないものの割合を最大化する．データセンタのエネルギー源は風力発電等の二酸化炭素を排出しないものと火力発電等の二酸化炭素を排出するものに分けられる．本文献では前者の使用割合が高いデータセンタへ VM を移動する．また我々の調査 [11] によれば，東北地方太平洋沖地震の発生時に東北大学（震央から 150 km）においてデータセンタの電源およびインターネットへの接続が数十分間維持された．災害時に発生地付近のデータセンタの VM を遠く離れた安全なデータセンタへ遠隔地ライブマイグレーションによって移動すれば重要な情報システムを維持できる．この様子を**図 1** に概念図で示す．

遠隔地ライブマイグレーションではネットワーク帯域が狭いことが最も重要な課題である．従ってデータ転送量をできる限り削減しライブマイグレーションを高速化することが WAN 環境で満たすべき要件である．ライブマイグレーションで転送すべきデータのうち，我々はゲスト OS のメモリに注目する．サイズではディスクイメージがメモリよりも数十倍から数百倍大きいが，第 2 章で述べたようにディスクイメージは書き込みが比較的低速であるためリアルタイムでの同期が可能である．一方メモリは書き込み速度が速いため，メモリイメージを移動元と移動先で常に同期することは現実的ではない．従って遠隔地ライブマイグレーションではメモリ転送量を削減することが特に重要である．

3.2 提案手法：Page Cache Teleportation

本研究では，ゲスト OS の持つページキャッシュの転送を省略し，移動先でディスクイメージから復元することで遠隔地ライブマイグレーションを高速化する．本手法を Page Cache Teleportation と呼ぶ．ゲスト OS がメモリ内に持つ様々なデータのうち，ページキャッシュの占める割合が多い場合が観察される [7]．ページキャッシュはディスクアクセスに対するキャッシュである．ページキャッシュのうちディスクイメージに同一のデータが存在する場合があり，ディスクイメージからコピーすることで復元可能である．通常の遠隔地ライブマイグレーションでは，このようなデータはディスク同期時とライブマイグレーション時に重複転送される．我々はゲスト OS のメモリ内のページキャッシュのうち，ディスクイメージから復元可能なページの転送を省略する．どのような場合に転送が省略できるかについては次節で詳しく述べる．転送を省略したページキャッシュは移動先ホストでゲスト OS から透過的に復元する．従ってゲスト OS から見れば本手法によるメモリ状態は変化せず，ページキャッシュを削除した場合に発生するようなゲスト OS の性能低下は発生しない．

3.3 ページキャッシュの復元と転送

本研究では，ページキャッシュのうちネットワーク越しの転送を省略しディスクイメージから復元可能な部分を判定することが技術的課題である．まず，ページキャッシュは OS によって二つの場合に分けられる．これらは Linux ではゲスト OS が管理する PG_dirty フラグで簡単に判定できる．

(1) ディスクに同一の内容が存在するもの．読み込みキャッシュ，または書き込みキャッシュがすでにディスクに書き戻された場合が該当する．PG_dirty フラグがオフである．
(2) ディスクに同一の内容が存在しないもの．書き込みキャッシュがディスクに書き戻されていない場合が該当する．PG_dirty フラグがオンである．

ライブマイグレーションではゲスト OS への影響を小さく抑えることが重要であり，提案手法は VM を動作させたまま行う．そのためあるメモリページは PG_dirty ではないと判定されても，判定の後に更新される可能性がある．これにはページキャッシュに書き込みが行われた場合，ページキャッシュが解放されメモリページが他の目的に使われ

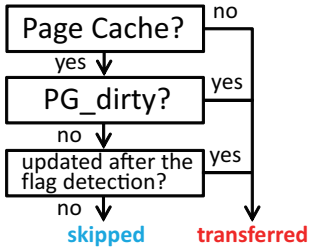

図 2 メモリページの状態による転送の実行，省略
Fig. 2 A memory page is skipped or transferred depending on the three conditions.

た場合の二通りがある．以上により，ページキャッシュに該当するメモリページは PG_dirty であるかないか，またフラグの判定後に更新されたかどうかによって 4 通りに分類できる．それぞれの場合についてネットワーク越しに転送すべきか転送を省略できるかを**図 2** に示す．

これらのメモリページは，PG_dirty による判定とは別の機構によって検知する必要がある．そこで本研究では，VMM のもつ dirty page tracking 機能を用いた．本機能は x86 アーキテクチャのもつ dirty bit をユーザ空間からも読み出し可能にし，QEMU/KVM に標準で実装されている．本手法の開始時に移動元で前に dirty page tracking を有効にし，メモリへの書き込みを監視する．

3.4 システムの動作

提案手法による遠隔地ライブマイグレーションの様子を**図 3** に示す．各ステップは以下のように動作する．

(1) 移動元ホストで dirty page tracking を有効化する．
(2) 我々の実装したカーネルモジュールが，ゲスト OS のメモリ内のページキャッシュのアドレス及び当該ページと同一内容のディスクブロック番号を取得する．
(3) モジュールから移動元と移動先の VMM へ，(2) で得たアドレスとディスクブロック番号を転送する．転送には簡単かつゲスト OS 非依存な方法として TCP/IP を用いる．
(4) 移動先の VMM がディスクイメージからゲスト OS のメモリへページキャッシュに対応するディスクブロックをコピーする（ページキャッシュの復元）．
(5) 移動元の VMM が移動先へメモリを転送する．ただしページキャッシュでありかつ (1) から (4) の間に更新されていないメモリページは転送しない．

4. 実装

4.1 実装方針

提案手法の実装は特定 OS への依存を可能な限り少なくすることが重要である．これはデータセンタでは Linux 以外にも BSD や Windows 等の OS も広く利用されることによる．提案手法を実現するために必要な主な機能は以下である．

(1) ゲスト OS のメモリ内のページキャッシュの位置と対応するディスクブロックの番号を取得する．
(2) 取得した情報を VMM へ伝達する．
(3) ディスクからゲスト OS のメモリへページキャッシュを復元する．

本研究では (1) のみをゲスト OS に依存する方法で実装した．具体的にはページキャッシュの位置および対応するディスクブロック番号を取得するカーネルモジュールを実装した．本モジュールの詳細については次節で述べる．

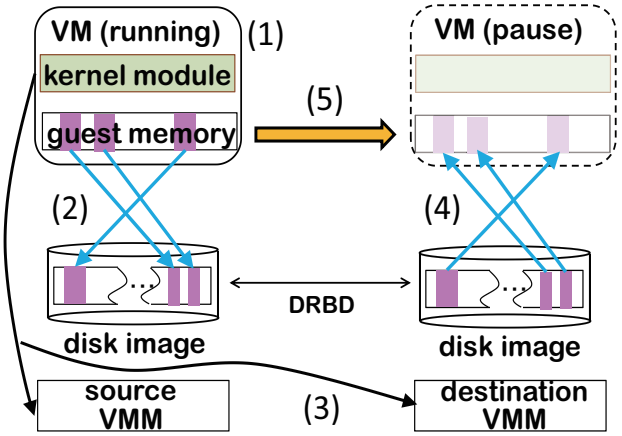

図 3 提案手法によるライブマイグレーション．(1) 移動元で dirty page tracking を有効化． (2) カーネルモジュールを用いてページキャッシュのゲストメモリとディスク上の位置を取得. (3) 取得したページキャッシュの位置を VMM へ送信． (4) 移動先でページキャッシュをディスクから復元． (5) メモリをネットワーク越しに転送する．ただしページキャッシュであり (1) から (4) の間に更新されていないページは転送しない．
Fig. 3 Proposed live migration procedure. (1) Dirty page tracking is enabled at the source host. (2) The kernel module detects where the page cache is on the guest memory and the disk. (3) The module sends them to the VMMs. (4) The destination VMM copies the disk blocks to the guest memory. (5) The source VMM sends memory pages, skipping page cache that is not dirtied during steps (1) through (4).

4.2 ページキャッシュ位置の判定

メモリ上のページキャッシュの位置をゲスト OS 内のドライバを用いて判定する．具体的に本稿では対象 OS を Linux とし，カーネルモジュールを実装することで判定した．本モジュールは 200 行程度（エラー処理等を除く）であり，特定 OS への依存は十分小さいと言える．

Linux ではメモリページの情報はページ構造体（`struct page`）によって管理される．ある物理ページ番号に対応するページ構造体は，カーネルソース内の `pfn_to_page` マクロを用いることでメモリ環境に非依存に取得できる．また `bmap` カーネル関数は，ある物理ページが

ページキャッシュであれば対応するディスクブロック番号を，ページキャッシュでなければ 0 を返す．struct page の flags メンバにはメモリページの状態を表すフラグが格納されている．PG_dirty フラグはページキャッシュのうちディスクに書き戻されていないページを表すため，当該フラグがオンのページはページキャッシュであってもディスクから復元することはできない．

取得したページキャッシュの位置と対応するディスクブロック番号は，ゲスト OS および VMM の種類に非依存に VMM へ送信される．ゲスト OS 内のユーザ空間プログラムがカーネルモジュールから取得した情報を VMM へ渡す．ユーザ空間プログラムと VMM の通信方法には制限がないため，本稿では簡単のため TCP/IP を用いた．例えば VMM とゲスト OS の間で共有メモリ空間を作る手法もありうるが，特定 VMM へ依存すること，利点が少ないことから適切ではない．

4.3 ディスクイメージの同期

ディスクイメージは DRBD の dual primary mode[*1] を用い同期する．提案手法ではゲスト OS の性能低下を防ぐためにページキャッシュの復元はゲスト OS が移動元で動作している間に行われる．すなわちディスクイメージに複数のホストが同時にアクセスする必要がある．一般に分散ファイルシステムにおいて複数ホストが単一のファイルへ同時にアクセスする場合，ロック機構を用いて同時書き込みが発生しないよう制御する．しかし提案手法ではページキャッシュの復元は読み込みしか行わない，すなわちディスクイメージへの複数ホストからの同時書き込みは起こらない．従って本研究では分散ファイルシステムのようなロック機構を導入する必要はない．

5. 評価実験

5.1 アプリケーションベンチマーク

本項では Page Cache Teleportation が WAN 環境における total migration time を削減することを実験により示す．VM 上でベンチマークを実行し，当該 VM を 2 台のホスト間でライブマイグレーションする．オリジナルの QEMU/KVM と提案手法でそれぞれマイグレーションを行った場合の total migration time およびメモリ転送量を比較した．実行したベンチマークは以下である．

Apache DB を用いない静的な web サイトを模したベンチマーク．Apache を用いてファイルを公開し，外部からネットワーク越しに当該ファイルを読み出す．ファイルサイズは画像や flash 等のコンテンツを想定し 300KB とし，ファイル数は 1024 個とした．

TPC-C DB を用いたトランザクションシステムのベンチマーク [12]．ショッピングサイトを模したデータおよびアクセスを発生させる．データの量を表す warehouse 数は 20 とした．

[*1] http://www.drbd.jp/users-guide/s-dual-primary-mode.html

表 1 ホストのマシン仕様
Table 1 Machine specifications of the hosts

CPU	Intel Xeon X5460 (4 cores)
Memory	8 GB
Disk	250GB HDD
Network	100Mbps または 50Mbps に制限
OS	Debian GNU/Linux 6.0.5
Kernel	Linux 2.6.32
QEMU	0.13.0

実験に用いたホストの仕様を**表 1** に示す．ゲスト OS は Debian GNU/Linux 6.0.5 で，一つの仮想 CPU および 1GB のメモリを割り当てた．ただしゲスト OS は Linux であることが現在の要件で，ホスト OS と同一の kernel バージョンやディストリビューションである必要はない．またネットワーク帯域は WAN 環境を再現するためにホスト上で tc コマンドを用いて 100Mbps および 50Mbps に制限した．

図 4 と**図 5** は各ベンチマークにおける total migration time であり，各棒グラフは提案手法の使用，未使用およびネットワーク帯域を 100Mbps，50Mbps で切り替えて行った結果である．また棒グラフの色が濃い部分はメモリ転送にかかった時間，色が薄い部分は移動先でページキャッシュをディスクから読み出してゲストのメモリへ復元するのにかかった時間である．Apache による結果では，100Mbps，50Mbps のいずれにおいても total migration time が削減された．特に 50Mbps では 65%の削減率が得られた．ページキャッシュを復元するのにかかる時間はネットワーク転送速度によらず一定である．これはディスクイメージが事前に同期されており，復元にかかる時間がディスクの読み出し速度のみに依存するからである．また TPC-C による結果では，50Mbps では提案手法が total migration time を削減しているが 100Mbps では削減できなかった．

表 2，**表 3** は各ベンチマークにおけるメモリ転送量を表す．メモリ転送量には移動先でディスクから読み出したページキャッシュの量は含まない．表 2 は Apache ベンチマークにおける結果である．300KB のファイルが 1024 個あるため，ページキャッシュの合計は約 300MB である．100Mbps，50Mbps のいずれの場合にも 300MB より有意に多い転送量が削減された．これは Apache のコンテンツの他に，カーネル，共有ライブラリ，Apache のプログラムコード等のためのページキャッシュが削減されているからである．参考にゲスト OS の起動直後にベンチマークを何も実行せずライブマイグレーションした場合の結果を**表 4** に示す．表より OS が起動しただけの状態でも約 45MB のページキャッシュを削減できることが分

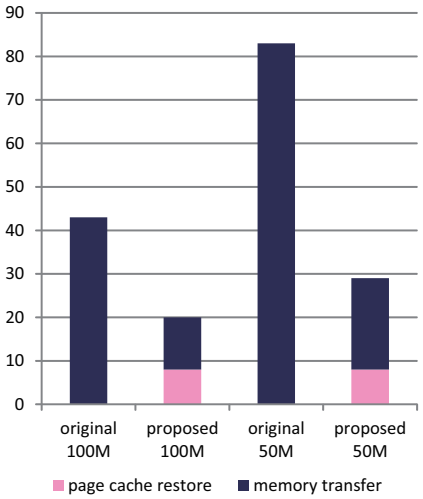

図 4 Apache ベンチマークにおける total migration time（秒）
Fig. 4 Total migration time (seconds) in Apache benchmark

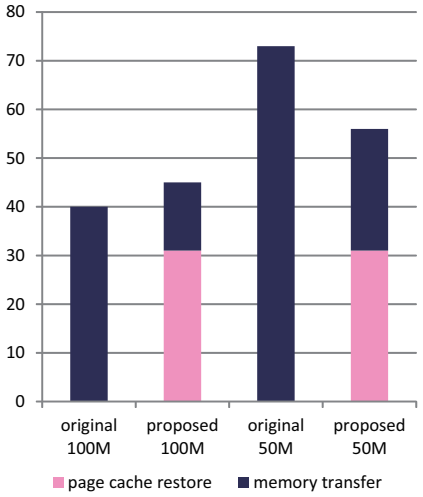

図 5 TPC-C ベンチマークにおける total migration time（秒）
Fig. 5 Total migration time (seconds) in TPC-C benchmark

かる．また表 3 は TPC-C ベンチマークにおける結果である．TPC-C においても 100Mbps，50Mbps ともに 300MB 程度の転送量を削減できた．ただし 100Mbps ではディスク読み込み速度とネットワーク帯域の関係から従来手法の方が total migration time は短い．また TPC-C ではネットワーク帯域が狭い場合に広い場合よりもメモリ転送量が少なかった．これはネットワーク帯域が狭いために DRBD の同期速度が下がり，ディスク書き込みを伴う TPC-C の性能が下がったからである．

5.2 ゲスト OS の性能低下

遠隔地ライブマイグレーションでは LAN 内のライブマイグレーションと同様にゲストの性能低下を最小限に留めることが必要である．例えば災害時の遠隔地ライブマイグレーションによる情報システムの維持では，災害時にシステムの利用者が急増すると考えられることからゲストの性能低下は重大な問題である．

表 2 Apache ベンチマークにおけるメモリ転送量
Table 2 Memory transfer in Apache benchmark

	proposed	original
100Mbps	110 MB	487 MB
50Mbps	110 MB	498 MB

表 3 TPC-C ベンチマークにおけるメモリ転送量
Table 3 Memory transfer in TPC-C benchmark

	proposed	original
100Mbps	142 MB	440 MB
50Mbps	139 MB	415 MB

表 4 ベンチマークを実行しない場合のメモリ転送量
Table 4 Memory transfer with no benchmarks

	proposed	original
100Mbps	95 MB	140 MB
50Mbps	95 MB	141 MB

本項では提案手法によるゲストの性能低下が十分小さいことを示す．ページキャッシュは比較的低速なディスクへのアクセスを軽減するため，ページキャッシュを削除するとアプリケーションの性能が低下する．しかし提案手法では移動先でゲストを再開する前にページキャッシュが移動先のメモリに復元されている．従ってページキャッシュの復元はゲストから透過的であり，ページキャッシュの転送を省略することによるゲストへの性能低下は発生しない．

ライブマイグレーションの直前にページキャッシュを削除し未使用ページを送信しなければ，提案手法と同等のメモリ転送量が削減されると考えられる．例えば Linux では/proc インターフェースを用いてページキャッシュを削除できる．提案手法およびライブマイグレーション直前にページキャッシュを削除した場合について，ファイル読み込み速度（ブロック/秒）を**図 6** に示す．図の上段が提案手法での結果，下段がマイグレーション直前にページキャッシュを削除した場合の結果である．ただし QEMU/KVM には未使用ページを転送しない機能[*2]がないため，下図では削除したページキャッシュはゲスト OS から見れば未使用であるが転送されている．0 秒でファイルの読み込みを開始し，150 秒でマイグレーションを行った．また読み込みがファイルの終端に達すると再び先頭から読み込みを開始した．開始直後はいずれの場合もファイルがページキャッシュに格納されておらずスループットが低い．15 秒後からの二周目以降の読み込みはページキャッシュがヒットするため高速である．提案手法ではマイグレーション時にメモリコピー等の負荷により読み込み速度が下がるが，移動先でページキャッシュが復元済みのため数秒で回復する．一方マイグレーション直前にページキャッシュを削除すると

[*2] 本機能も，メモリを単なるバイト列として扱うと実現できない

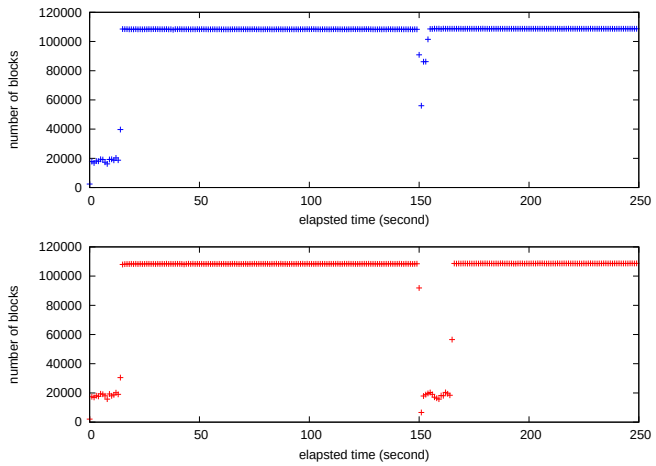

図 6 提案手法（上）およびマイグレーション直前にページキャッシュを削除した場合（下）のファイル読み込み速度

Fig. 6 File read throughput in proposed method (top) and dropping page cache just before a migration (bottom)

読み込み速度が回復するまで 15 秒かかる．以上により提案手法によるゲスト OS の性能低下は十分小さいと言える．

6. 議論

6.1 ネットワークオーバーヘッド

提案手法によるネットワークオーバヘッドが十分に小さいことを示す．具体的には，ページキャッシュに該当するメモリページ番号とディスクブロック番号の転送量が，実験で示したメモリ転送量に比して十分小さい．Linux ではディスクブロック番号は `sector_t` 型で表され，そのサイズは一般に 64 ビットである．また一般にページサイズは 4KB（=12 ビット）だから，64 ビットアーキテクチャではメモリページの番号は 52 ビットで表せる．よってメモリページ番号とディスクブロック番号の組は 116 ビット（<15 バイト）である．ゲスト OS の物理メモリが 4GB のとき，メモリページ数は 4GB÷4KB=1M ページだから，全メモリページがページキャッシュの場合でも提案手法で転送される情報は 15M バイト未満である．

6.2 類似研究との比較

本項では提案手法と類似する手法を比較する．Symbiotic Virtualization [13] ではゲスト OS 及び VMM を改変することで，ホストからゲスト OS のシステムコールを呼び出せる．本研究はホストがゲスト OS の持つ情報を利用するため，文献 [13] と着眼点が類似する．例えばゲスト OS にページキャッシュの位置を返すシステムコールを定義し，文献 [13] を用いてホストから呼び出すことでも Page Cache Teleportation を実装できる．文献 [13] では本研究と異なり，ホストからのシステムコールの呼び出しからゲストでの実行完了までが同期的に行われ割り込みが発生しない．これにより VM exit のハンドラ内など割り込み不可能なコンテキストにおいてもゲストのシステムコールを呼び出し，ゲスト OS 内の情報を用いた動的な動作ができる．一方，本研究では VMM とゲスト OS の通信にユーザ空間プログラムを用いる．この方法は実装が簡単で特定 OS への依存が少ないが，実行中に他のコンテキストが実行される可能性がある．

文献 [14] ではライブマイグレーション時の CPU やネットワークへの負荷をマイグレーションノイズと呼び，ライブマイグレーションにかかるメモリ転送量を減らすことでマイグレーションノイズを削減する．本研究と同様に，ゲスト OS を用いてページキャッシュの位置を判定しその転送を省略することでメモリ転送量を削減する．本研究と文献 [14] の主な相違点は，ページキャッシュを移動先でゲスト OS のメモリへ復元するかどうかである．本研究ではディスクイメージが常に同期されているため，ゲスト OS を再開する前にページキャッシュを復元しゲスト OS の性能低下を最小限に抑える．一方文献 [14] では送信しなかったページキャッシュは存在しないとしてゲスト OS のデータ構造を書き換え，ゲスト OS が再びファイルにアクセスするとページフォルトが発生し性能が低下する．

7. 結論と今後の課題

本稿では遠隔地ライブマイグレーションの課題について述べ，ゲスト OS のメモリのうちページキャッシュが多くを占める場合に着目した．ページキャッシュをマイグレーション時に転送せずディスクイメージから復元する手法 Page Cache Teleportation を提案した．データセンタでは様々な OS が利用されることから，特定のゲスト OS への依存をできる限り少なくするよう実装を行った．評価実験の結果，提案手法は遠隔地ライブマイグレーションを高速化しかつゲスト OS の性能低下も小さいことが示された．

今後の課題はディスクイメージのリアルタイム同期よるゲスト OS への影響（6.1 に記述）の軽減およびセキュアな環境での提案手法の実現である．クラウド環境ではホストの管理者とゲストの管理者が異なるため，ホストからゲスト OS のメモリやディスクイメージを自由に閲覧することは好ましくない．例えば CloudVisor [15] では，Nested Virtualization [16] を用いて IO をゲストおよび VMM から透過的に暗号化する．VMM は VM exit の原因となった命令の処理など限られた場合にのみゲストのメモリを読み出せる．提案手法をセキュリティを考慮した実用的な方式へ発展させることは重要な課題である．

参考文献

[1] Al-Kiswany, S., Subhraveti, D., Sarkar, P. and Ripeanu, M.: VMFlock: Virtual Machine Co-migration for the Cloud, *International Symposium on High Performance Distributed Computing*, pp. 159–170 (2011).
[2] Moghaddam, F. F., Cheriet, M. and Nguyen, K. K.:

Low Carbon Virtual Private Clouds, *IEEE International Conference on Cloud Computing*, pp. 259 –266 (2011).
[3] 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. 2, No. 2, pp. 152–165 (2009).
[4] Takahashi, K., Sasada, K. and Hirofuchi, T.: A Fast Virtual Machine Storage Migration Technique Using Data Deduplication, *International Conference on Cloud Computing, GRIDs, and Virtualization*, pp. 57–64 (2012).
[5] Mashtizadeh, A., Celebi, E., Garfinkel, T. and Cai, M.: The design and evolution of live storage migration in VMware ESX, *USENIX Annual Technical Conference* (2011).
[6] DRBD (online): `http://www.drbd.org/`.
[7] Akiyama, S., Hirofuchi, T., Takano, R. and Honiden, S.: MiyakoDori: A Memory Reusing Mechanism for Dynamic VM Consolidation, *IEEE International Conference on Cloud Computing*, pp. 606 –613 (2012).
[8] Svärd, P., Hudzia, B., Tordsson, J. and Elmroth, E.: Evaluation of delta compression techniques for efficient live migration of large virtual machines, *International Conference on Virtual Execution Environment*, pp. 111–120 (2011).
[9] Hines, M. R. and Gopalan, K.: Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning, *International Conference on Virtual Execution Environments*, pp. 51–60 (2009).
[10] Zhang, X., Huo, Z., Ma, J. and Meng, D.: Exploiting Data Deduplication to Accelerate Live Virtual Machine Migration, *International Conference on Cluster Computing*, pp. 88–96 (2010).
[11] Tsugawa, M., Figueiredo, R., Fortes, J., Hirofuchi, T., Nakada, H. and Takano, R.: On the Use of Virtualization Technologies to Support Uninterrupted IT Services, *IEEE ICC 2012 Workshop on Re-think ICT infrastructure designs and operations*, pp. 7892–7896 (2012).
[12] TPC-C (online): `http://www.tpc.org/tpcc/`.
[13] Lange, J. R. and Dinda, P.: SymCall: Symbiotic Virtualization Through VMM-to-Guest Upcalls, *International Conference on Virtual Execution Environments*, pp. 193–204 (2011).
[14] 古藤明音, 山田浩史, 大村圭, 河野健二：仮想マシン移送における移送ホストの負荷軽減手法，情報処理学会研究報告，Vol. 2012-OS-122 (2012).
[15] Zhang, F., Chen, J., Chen, H. and Zang, B.: CloudVisor: Retrofitting Protection of Virtual Machines in Multi-tenant Cloud with Nested Virtualization, *ACM Symposium on Operating Systems Principles*, pp. 203–216 (2011).
[16] Ben-Yehuda, M., Day, M. D., Dubitzky, Z., Factor, M. and Har'El, N.: The Turtles Project: Design and Implementation of Nested Virtualization, *USENIX Symposium on Operating Systems Design and Implementation* (2010).