Vol.2013-OS-124 No.7
2013/2/28

ページキャッシュの復元とその他のメモリ転送の並列動作による遠隔地ライブマイグレーション高速化

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，これまでデータセンタ内で行われてきた VM の配置最適化をデータセンタ間で行うことができる．一方でデータセンタ間の帯域は LAN に比較して狭いため効率的な遠隔地ライブマイグレーション技術が求められる．我々は VM のメモリの多くをページキャッシュが占める場合に着目する．本論文ではマイグレーション時にページキャッシュを転送せずディスクイメージから復元し，さらにページキャッシュの復元をその他のメモリ内容のネットワーク越しの転送と並列動作させて遠隔地ライブマイグレーションを高速化する．評価の結果，ページキャッシュの更新が少ないワークロードでは並列化が有効に働き遠隔地ライブマイグレーションが高速化できた．また本論文では提案手法の LAN 内でのマイグレーションへの適用可能性についても論じる．

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

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．具体的には，負荷の高まりに応じて仮想マシン (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]，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]，災害発生時に VM を遠隔地の安全なデータセンタへ移動することによる情報システムの維持 [3] などが提案されている．

WAN 環境でのデータセンタ間の帯域は Local Area Network（LAN）環境でのホスト間の帯域よりも一般に狭い．よって遠隔地ライブマイグレーションを活用する研究を実用化するためには効率的なマイグレーション手法が必要である．web サーバなどファイルを多く扱うワークロードでは，VM のメモリ中のページキャッシュが大きな割合を占めることがある．従ってページキャッシュに着目したライブマイグレーション高速化手法が求められる．ライブマイグレーションにかかるメモリ転送量を削減しマイグレーション完了時間を削減する手法は盛んに研究されている [4–10]．しかしページキャッシュに着目した研究 [4, 5] ではページキャッシュを削除し転送しないという手法を採る．従ってマイグレーション完了後のディスク IO 性能が低下することが問題である．

本研究では，ゲスト OS のメモリをディスクイメージから復元可能なページキャッシュとその他の部分に区別し，それらを並列に移動先ホストのメモリに転送することで，遠隔地ライブマイグレーションを高速化する．ディスクイメージは既存のストレージ同期機構等を用いてライブマイグレーション前に同期されており，ページキャッシュは移動先のホストでディスクから読み出される．一方，ページキャッシュ以外のメモリは WAN 越しに移動元から移動先へ転送される．ページキャッシュの復元は VM が移動先で resume する前に VM から透過的に行われるため，マイグレーション完了後の IO 性能低下が少ないと考えられる．提案手法を QEMU/KVM 上に実装しシミュレーションされた WAN 環境で評価した結果，ページキャッシュの更新が少ないワークロードにおいては並列化が有効に働き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．第二章は遠隔地ライブマイグレーションの利用例と課題を挙げる．第三章は提案手法

[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)

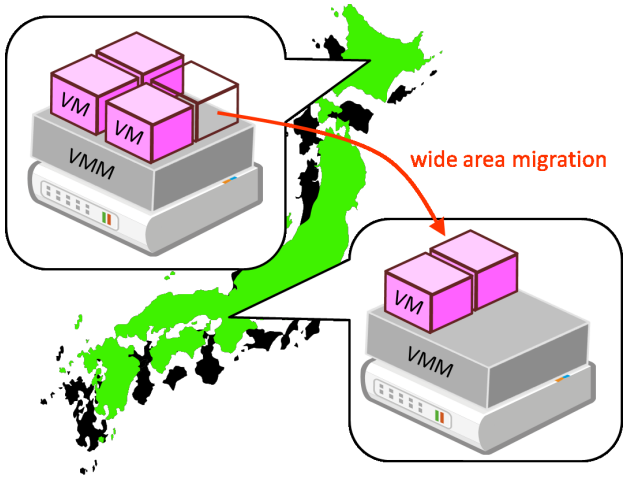

図 1 遠隔地ライブマイグレーション
Fig. 1 Wide area live migration

の概要と提案システムの動作を説明する．第四章は提案手法実装の設計および技術的詳細を述べる．第五章は評価実験によって提案手法が遠隔地ライブマイグレーションを高速化することを示す．第六章は提案手法の欠点や LAN への適用可能性について議論する．第七章で関連研究を挙げ，第八章で本論文を結論する．

2. 遠隔地ライブマイグレーション

2.1 利用例

遠隔地ライブマイグレーションは，仮想化されたデータセンタ間を仮想マシン（VM）がその実行を停止せずに移動する技術である．本技術はデータセンタ内のホスト間でのライブマイグレーション [11] を，WAN をまたいだデータセンタ間に適用したものである．遠隔地ライブマイグレーションの概念図を**図 1** に示す．図では北海道にあるデータセンタから大阪にあるデータセンタへ VM をライブマイグレーションで移動している．

遠隔地マイグレーションは様々な研究で利用が提案されている．文献 [1] では VM をデータセンタ間で移動することによってデータセンタ間負荷分散を実現する．VM のディスクイメージは一般に数 GB 以上と大きい．本文献では複数の VM の集まりを flock を呼び，flock 内の VM のディスクイメージ間でデータの重複排除をすることでディスクイメージの転送を高速化する．文献 [2] では使用エネルギー源の異なるデータセンタ間で VM を移動することで，VM による二酸化炭素排出量を削減する．データセンタのエネルギー源には風力発電や水力発電のように二酸化炭素を排出しないものが考えられ，かつそれらの割合は気象条件によって変化する．VM を二酸化炭素を排出しないエネルギー源の使用割合が多いデータセンタへ移動することで，VM の消費電力は一定のままで二酸化炭素排出量を削減できる．文献 [3] では遠隔地ライブマイグレーションを情報システムの災害復旧に利用することを提案する．災害時に災害発生地付近のデータセンタの VM を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．本文献の調査によれば，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．

2.2 課題

データセンタ間のネットワーク帯域は一般にデータセンタ内のホスト間の帯域よりも狭い場合が多い．近年ではデータセンタ間での 1 Gbps 以上のネットワークも普及しつつあるが，データセンタ内のネットワークに比較してデータセンタ間のネットワークは多くのユーザに共有されるため占有できる帯域は狭い．従って，2.1 で挙げたシステムを実用化するためには，効率的な遠隔地ライブマイグレーションの手法が必要である．

Web サーバや DB などのワークロードでは，VM の使用メモリのうちディスク IO に対するキャッシュであるページキャッシュが多くを占める場合がある．VM のディスクイメージを移動元と移動先で共有・同期する場合，ライブマイグレーション完了時間は移動元から移動先へメモリを転送する時間がほぼ全てを占める．従って効率的な遠隔地ライブマイグレーションを実現するためにはページキャッシュに注目してメモリの転送量を削減することが重要である．

ページキャッシュに着目した既存研究には文献 [4] および文献 [5] が挙げられる．これらの研究ではページキャッシュを削除し未使用ページとして転送しないという手法を採る．従ってライブマイグレーション後に VM が再びディスク IO を行うとキャッシュがないため性能が低下するという問題がある．そこで本研究では，ページキャッシュを含むメモリをマイグレーション時に転送しないことで遠隔地ライブマイグレーションを高速化し，かつ VM の性能低下も少ない手法を提案する．

3. 提案手法

3.1 ページキャッシュ

本研究では VM のメモリ中のページキャッシュに着目する．ページキャッシュはディスク IO に対するオンメモリキャッシュであり，ファイルの読込や書込に対しキャッシングを行い次回のアクセスを高速化する．

ページキャッシュには以下の二つの性質がある．

(1) ディスク上とページキャッシュに同一のデータが存在する場合がある．読込キャッシュおよびディスクに書き戻された書込キャッシュが該当する．
(2) Linux や Windows では空きメモリを可能な限りページキャッシュに割り当てるため，ページキャッシュの量が非常に多くなる場合がある．

性質（1）に該当するページキャッシュは，メモリ上に存

在せずともディスクイメージから復元可能である．本論文ではこのようなページキャッシュを**復元可能である**と称する．性質 (2) は web サーバやデータベースなどの扱うデータ量が多いワークロードを実行する場合に該当する．

3.2 提案手法：ページキャッシュの復元とその他のメモリ転送の並列動作

我々は VM のメモリ中の復元可能なページキャッシュの転送を省略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．メモリ中に復元可能なページキャッシュが多く存在する場合，遠隔地ライブマイグレーションに多大な時間がかかる．実際に我々は web サーバをシミュレートしたワークロードにおいてページキャッシュが VM のメモリの多くを占める場合を観察した [6]．復元可能なページキャッシュは，3.1 節の性質 (1) より，ライブマイグレーション時に転送せずともディスクイメージから移動先のメモリへ直接転送できる．ただしゲスト OS のディスクイメージは事前に DRBD [12] などの同期機構やストレージマイグレーション [13] によって同期されていると仮定する．WAN 環境ではローカルのストレージからの転送はネットワーク越しの転送よりも高速であるから，これにより遠隔地ライブマイグレーションを高速化できる．さらに，ページキャッシュの復元とその他のメモリ内容の転送を並列動作させることでページキャッシュの復元にかかる時間を隠蔽する．

提案手法ではページキャッシュの復元を VM から透過的に行うことで従来研究における VM の性能低下を削減する．ページキャッシュの転送を省略する従来研究 [4, 5] ではページキャッシュをゲスト OS が自ら復元するため，キャッシュが復元されるまでの間のディスク IO 性能が低下する．一方提案手法ではページキャッシュの復元をライブマイグレーション中に行い，VM が移動先で resume する時点ではすでにキャッシュの復元が完了している．従って resume 直後のディスク IO 性能の低下が削減できると考えられる．

3.3 先行研究：Page Cache Teleportation

提案手法は我々の先行研究 [14] において逐次的に行っていたページキャッシュの復元とその他のメモリ転送を並列化したものである．先行研究では，ゲスト OS の知識を用いて復元可能なページキャッシュのメモリ及びディスク上の位置を検出し，復元可能なページキャッシュを WAN 越しに転送せずに移動先での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した．本手法を Page Cache Teleportation とよび，Apache および TPC-C を用いた評価の結果帯域の狭い環境下でライブマイグレーションを高速化することが示された．

先行研究では復元可能なページキャッシュの復元とその他のメモリの転送を**逐次的に**行う点で更なる最適化の余地が残る．すなわち，移動先でページキャッシュの復元が完了してからページキャッシュ以外のメモリを移動元から移動先へ転送する．ディスクからのコピーとネットワークからのパケット受信は並列に行えるためこの点において最適化が不十分であった．オリジナルのライブマイグレーション（全メモリを WAN 越しに転送），先行研究，提案手法におけるライブマイグレーション完了時間を概念的に示すと**図 2** のようになる．先行研究では復元可能なページキャッシュの転送は移動先のディスクから行われるため，オリジナルの手法よりマイグレーション完了時間が短い．本研究ではさらにページキャッシュの復元とその他のメモリの転送を並列に行うことで先行研究以上の高速化を実現する．

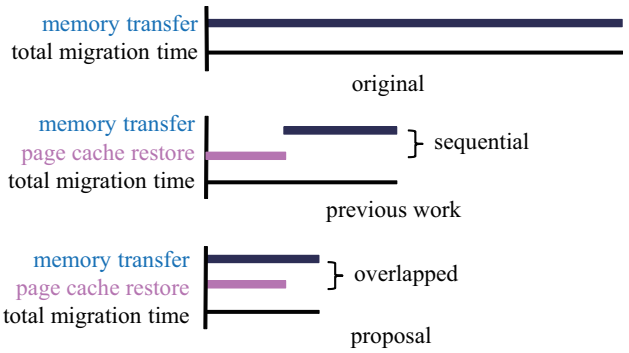

図 2 オリジナルの手法（上段），先行研究（中段），提案手法（下段）におけるライブマイグレーション完了時間の概念図

Fig. 2 Overview of the total migration time in the original live migration (top), the previous work (middle), and the proposal of this paper (bottom)

3.4 システムの動作

提案システムの動作を**図 3** に示す．各ステップは以下のように動作する．

(0) VM のディスクイメージはディスク同期機構やストレージマイグレーションを用い事前に同期されている．
(1) 移動先 VMM で dirty page tracking を有効にする．この機能により更新されたメモリページが検知できる．
(2) VM 内のカーネルモジュールが，復元可能なページキャッシュの物理ページ番号（PFN）及び対応するディスクブロック番号を検出する．
(3) 取得した PFN およびディスクブロック番号を，VM 内のユーザランドプログラムが各 VMM へ転送する．
(4) ライブマイグレーションを開始する．下記の動作を並列に行う．
  (a) 移動元から移動先へ復元可能なページキャッシュ以外のメモリページを転送する．
  (b) 移動先で復元可能なページキャッシュをディスクイメージからメモリへ転送する．
(5) 復元可能なページキャッシュの転送が終了し，かつ転送すべきメモリが十分少なくなると，CPU の状態と

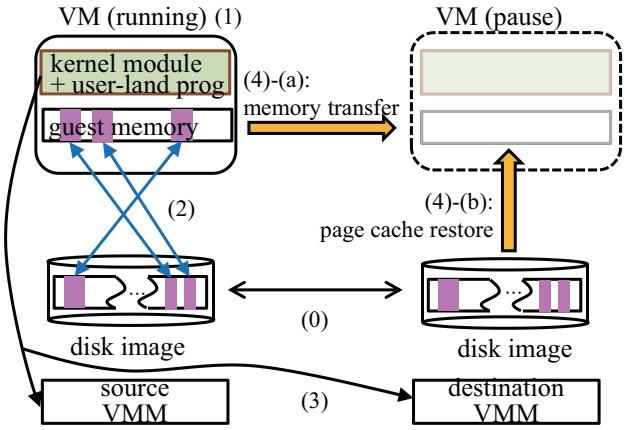

図 3 提案システムの動作 (0) ディスクイメージを事前に同期 (1) dirt page tracking を有効化 (2) 復元可能なページキャッシュと対応するディスクブロックを検出 (3) 検出した情報を VMM に転送 (4) ページキャッシュの復元とその他のメモリの転送の並列動作 (5) CPU 状態を転送しマイグレーションを完了

Fig. 3 Procedure of our proposal: (0) sync disk images (1) enalges dirty page tracking (2) detects restorable page cache and correspoding disk blocks (3) send detected information to VMMs (4) paralle execution of page cache restore and memory transfer (5) transfer CPU states and completes the migration

残りのメモリを転送しマイグレーションを完了する．

4. 設計と実装

4.1 設計

我々は提案手法を以下の構成要素により実現する．

(1) 復元可能なページキャッシュの PFN および対応するディスクブロック番号を検出するカーネルモジュール．

(2) 取得した PFN およびディスクブロック番号を VM から VMM へ送信するユーザ空間プログラム．

(3) 復元可能なページキャッシュをディスクからメモリへ転送する機能．VMM に追加されたスレッドが行う．本スレッドを**ページキャッシュ復元スレッド**と呼ぶ．

(4) ライブマイグレーションにおいて復元可能なページキャッシュを移動元から移動先へ転送しない機能．VMM のライブマイグレーションの機能を改変する．移動先の VMM でメモリを受信するスレッドを**メモリ受信スレッド**と呼ぶ．

VMM には Linux 上の QEMU/KVM を用いた．QEMU/KVM は Linux で標準サポートされていること，コードが Xen に比較して小さく変更が容易なことが理由である．ただし本研究では KVM の独自機能には依存していないため Xen 等のその他の VMM に再実装する際にも技術的障壁は少ないと考えられる．

復元可能なページキャッシュの検出にカーネルモジュールを用いる理由は，実装コストが低く VM ユーザの負担も小さいと考えるからである．カーネルモジュールを用いる他に，VM のメモリをホスト OS が解析する手法が考えられる．ホスト OS からのメモリの解析はカーネルデータ構造のバイナリ配置の理解を必要とし，実装コストが高い．一方カーネルモジュールを用いるとゲスト OS のカーネル関数を利用できるためデータ構造のバイナリ配置を理解する必要がない．実際に我々のカーネルモジュールはエラー処理を除いて 200 行以下である．カーネルモジュールが実際に行っている処理については先行研究 [14] の 4.2 節を参照のこと．

復元可能なページキャッシュの転送は，QEMU が行うのが最適である．まず，QEMU からは VM のディスクイメージおよび VM に確保されたメモリへ簡単にアクセスできるため実装コストが低い．さらに QEMU への変更は VM のユーザからは透過的であるためユーザへの負担もない．

4.2 ページキャッシュの更新への対応

第 3.4 節で述べたように，提案手法では移動元で VM の CPU を動作させたままページキャッシュの復元を行う．従って，あるメモリページは，カーネルモジュールが復元可能なページキャッシュを検出した時点から実際にページキャッシュが復元されるまでの間に更新される可能性がある．更新されたメモリページはディスクイメージ上に存在しないため，ネットワーク越しに転送しメモリの一貫性を保持する．ページキャッシュを復元する際に VM の CPU を停止させればこのような問題は起こらないが，VM の停止時間が長くなるため望ましくない．事象の詳細とその対処については先行研究 [14] の 3.3 節を参照のこと．

4.3 ページキャッシュの復元

復元可能なページキャッシュのディスクからメモリへの転送は，QEMU 内に実装されたページキャッシュ復元スレッドが行う．本スレッドはディスクイメージから復元可能なページキャッシュの含まれるディスクブロックを read システムコールによって転送する．

本論文で復元するページキャッシュは，以下の 2 種類のデータのキャッシュである．

- ファイルの内容そのもの
- ファイルに関する inode

メモリページとディスクブロックの対応のようなページキャッシュに関するメタデータはページキャッシュには含まれない．従ってページキャッシュ復元スレッドはカーネル内のデータ構造を理解・復元する必要はない．

ページキャッシュの復元は，転送元のディスクブロック番号順に行う．復元順序は提案手法の機能的側面には影響しない．しかしその他の復元順序に比較して，ディスクへのアクセスがシーケンシャルになるためページキャッシュの復元が高速に行える．我々の実験では復元を転送先の

Algorithm 1 Memory Transfer

```
while source_host_has_more_pages_to_transfer() do
  (i, data) ⇐ receive_memory_page()
  lock(i)
  received[i] ⇐ 1
  memory[i] ⇐ data
  release_lock(i)
end while
```

Algorithm 2 Page Cache Restore

```
j ⇐ 0
while j < Number_of_Memory_Pages do
  t ⇐ try_lock(j)
  if t and received[i] == 0 then
    restore_page_cache(j)
  end if
  if t then
    release_lock(j)
  end if
  j ⇐ j + 1
end while
```

メモリアドレス順行う場合に比べて転送元のディスクブロック行う方がページキャッシュの復元が 3 倍程度高速であった．

4.4 並列動作のアルゴリズム

ページキャッシュの復元とその他のメモリの転送の並列化をロック機構を用いて実装する．ページキャッシュ復元スレッドとメモリ受信スレッドは以下のように動作する．これらのスレッドは共に移動先の QEMU 内に実装されている．

メモリ受信スレッド メモリページを受信順に VM のメモリに書き込む．全メモリページ分の受信済みフラグを持ち受信したページに対応するフラグを立てる．同一ページにページキャッシュ復元スレッドが書き込み中ならば，その完了を待機して当該ページを上書きする．

ページキャッシュ復元スレッド 復元可能なページキャッシュを，復元元のディスクブロック番号順に復元する．復元先のページにメモリ受信スレッドが書き込み中または書き込み済みならば，当該メモリページへは書き込まず次のディスクブロックへ進む．

Algorithm 1，Algorithm 2 にこれらを疑似コードで表す．$lock(n)$ は n 番目のメモリページについてロック獲得を試行し獲得できるまでブロックする関数，$try\_lock(n)$ は n 番目のメモリページについてロック獲得を試行し獲得できれば真，できなければ偽を直ちに返す関数である．実装では pthread_mutex_lock，pthread_mutex_trylock を用いた．

マイグレーション中に移動元で VM のページキャッシュが更新されると，当該メモリページには二つのスレッドが共に書き込みを行うことがある．あるメモリページに含まれるページキャッシュは，カーネルモジュールによって検出された時点では復元可能でもその後更新され復元可能でなくなる場合がある．このようなメモリページは WAN 越しに転送する必要がある．この時，当該メモリページにはメモリ受信スレッドの方がページキャッシュ復元スレッドよりも後に書き込む必要がある．Algorithm 1 および Algorithm 2 ではロック機構および受信済みフラグを用いてこれを保証する．

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（性能は本文中に記載）
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

5. 評価実験

5.1 評価環境

提案手法が遠隔地ライブマイグレーションを高速化することをアプリケーションベンチマークで示す．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．ホスト間のネットワークは WAN を模すため帯域制限されている．従来手法，我々の先行研究，提案手法でそれぞれマイグレーションを行った場合のマイグレーション完了時間およびメモリ転送量を比較した．従来手法にはオリジナルの QEMU/KVM を用いた．これの手法は全てのメモリをネットワーク越しに転送し圧縮等は行わない．実行したベンチマークは以下である．

web サーバ 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，httperf [15] を用いてネットワーク越しに当該ファイルを読み出す．ファイルの読み出しは実験結果への影響をなくすためライブマイグレーションとは別のネットワークから行った．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．

実験に用いたホストの仕様を**表 1** に示す．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．HDD の性能は bonnie++ [16] を用いた測定の結果シーケンシャルアクセス時の読み出しが 90MB/秒程度，書き込みが 25MB/秒程度であった．またディスクイメージは DRBD を用いてホスト間で同期されている．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．ゲスト OS は Linux であることが要件で，ホスト OS と同一のカーネル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．

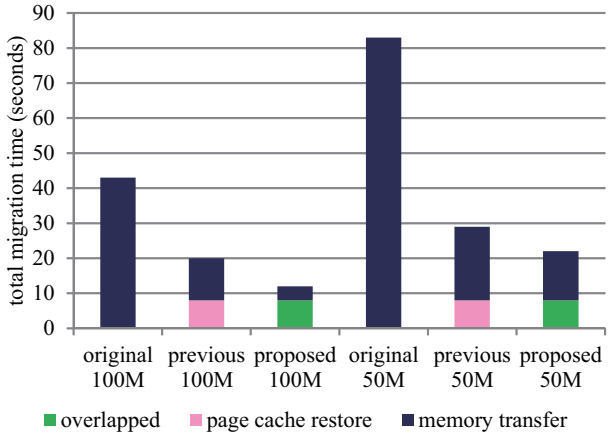

図 4 web サーバベンチマークにおけるマイグレーション完了時間
Fig. 4 Total migration time in the web server benchmark

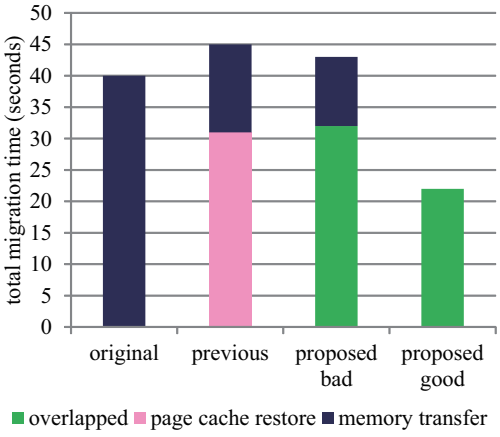

図 5 DB ベンチマークにおけるマイグレーション完了時間
Fig. 5 Total migration time in the DB benchmark

表 2 web サーバベンチマークにおけるメモリ転送量
Table 2 Memory transfer in the web server benchmark

	original	previous	proposed
100 Mbps	487 MB	110 MB	108 MB
50 Mbps	498 MB	110 MB	114 MB

5.2 評価結果

図 4 は，web サーバベンチマークにおけるマイグレーション完了時間を従来手法，我々の先行研究，提案手法で比較した結果である．数値は 3 回の実験の平均をとった．各棒グラフの original，previous，proposed はそれぞれ従来手法，先行研究，提案手法による結果を表し，100M，50M はホスト間の帯域幅を表す．図の page cache restore は復元可能なページキャッシュが転送されている時間，memory transfer はその他のメモリページが WAN 越しに転送されている時間，overlapped はこれらが並列に行われている時間である（図 2 も参照のこと）．提案手法は 100M，50M のいずれの帯域幅においても全手法で最も短いマイグレーション完了時間を実現した．提案手法での overlapped の時間と先行研究での page cache restore の時間が等しい．また提案手法でのマイグレーション完了時間と先行研究でのページキャッシュ以外のメモリ転送時間が等しい．これは提案手法による並列化がオーバーヘッドなく有効に働いていることを意味する．

表 2 は web サーバベンチマークにおけるメモリ転送量を各手法および帯域幅ごとに示したものである．数値は 3 回の実験の平均をとった．従来手法に対して先行研究および提案手法では 380MB 程度の削減ができた．web サーバの保持するファイルは各 300KB で 1024 個だから合計 300MB であり，380MB よりも有意に小さい．これは復元可能なページキャッシュが web サーバの保持するファイル以外にもゲスト OS の持つプログラムコードや共有ライブラリを含むからである．また我々の先行研究と提案手法では転送されたメモリ量はほぼ同じである．これは提案手法がメモリ転送量自体を先行研究より削減する手法ではなく，メモリ転送とページキャッシュの復元を並列化する手法だからである．

6. 議論

6.1 ロック競合

提案手法のアルゴリズムでは，ページキャッシュの更新が頻繁に起こる場合にロックの競合が多発する可能性がある．あるメモリページが，カーネルモジュールが動作した時には復元可能なページキャッシュを含んでいたが，その後更新された場合を考える．この時当該メモリページはメモリ受信スレッドとページキャッシュ復元スレッドの両方によって書き込まれる．メモリ受信スレッドは最新のデータを上書きする必要があるため，ページキャッシュ復元スレッドが書き込み中のメモリページに書き込みを試行した場合にはロックが解放されるまでブロックされる（Algorithm 1，Algorithm 2 も参照のこと）．

図 5 は DB ベンチマークを第 5 章と同一の環境で実行した時のマイグレーション完了時間である．当該ベンチマークは DB を用いたトランザクションシステムのベンチマーク [17] であり，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．ただしここでは帯域が 100 Mbps の場合のみ示す．図の original は既存手法での結果，previous は我々の先行研究での結果，proposed good は提案手法でロック競合が少ない場合，proposed bad は提案手法でロック競合が多発している場合の結果である．以下に現象の詳細を述べる．

proposed good ロック競合が多発しなかったため，ライブマイグレーション完了時間が削減できた．ページキャッシュの復元に約 22 秒かかり，メモリ受信スレッドはページキャッシュ復元スレッドの終了を待って

終了するためグラフ全体が overlapped になっている．previous に比較してページキャッシュの復元にかかっている時間が短い理由は，メモリ受信スレッドによって既に転送されたメモリページはページキャッシュ復元スレッドによって復元する必要がないからである．

proposed bad ロック競合が多発したため，ライブマイグレーション完了時間が削減できなかった．ページキャッシュの復元に約 32 秒かかり，その他のメモリ転送に 43 秒かかっている．previous ではページキャッシュの復元とその他のメモリは逐次的に行われており，前者が 31 秒で完了した後に後者が 14 秒で完了した．従って提案手法でロック競合が多発する場合にはメモリ受信スレッドが長時間待たされ並列化が有効に働かない．

6.2 提案手法によるオーバーヘッド

6.2.1 ネットワークオーバーヘッド

提案手法では復元可能なページキャッシュの PFN および対応するディスクブロック番号を移動先へ転送するが，その量は十分小さい．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 PFN は 52 ビットで表せる．また 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．よって PFN 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 の復元可能なページキャッシュが 4GB あるとき，そのメモリページ数は 4GB÷4KB=1M ページだから，提案手法で転送される情報は 15M バイト未満である．これは 4GB をすべて転送するのに比較して十分小さく，実用上十分である．

6.2.2 カーネルモジュールによるオーバーヘッド

提案手法では VM にカーネルモジュールをインストールする必要があるが，そのオーバーヘッドは以下の理由により十分小さいと言える．

- カーネルモジュールのバイナリサイズは約 155 KB であり，メモリ上に常駐しても問題ない．
- カーネルモジュールはマイグレーション開始時に一度動作するのみで，それ以外の時点では何もしない．
- マイグレーション時のカーネルモジュールの動作（全メモリページの情報を走査）は 1 GB のメモリを持つ VM で 1 秒以内で終了する．

6.2.3 ロックデータ構造によるオーバーヘッド

提案手法ではページキャッシュの復元とその他のメモリ転送の並列化のためにロック機構を利用する．現在の実装では各メモリページごとに pthread_mutex_t 型を 1 つ利用する．当該型のサイズは Linux 上では 40 バイトであり，1 メモリページは 4KB であるため，VM のメモリ量の $40B \div 4KB \approx 1\%$ のメモリを余分に消費する．しかし，ロックに関する情報はネットワーク越しに転送する必要はない，マイグレーション完了後には必要ないため解放できる，という性質があるため，これは実用上問題にならない．

6.3 データセンタ内マイグレーションへの適用

提案手法のデータセンタ内のライブマイグレーションへの適用について述べる．データセンタ内のマイグレーションでは VM のディスクイメージが NFS 等を用いて共有されるため，ディスクイメージからのページキャッシュの復元はネットワークを通して行われる．しかし，データセンタ内のノード間ネットワークは，管理用，ストレージ共有用，一般通信用と一般的に複数存在する．このようなシステムに提案手法を適用するとページキャッシュはストレージ共有用のネットワーク，その他のメモリは一般通信用のネットワークと二本のネットワークを通してメモリを分割転送でき，マイグレーションの高速化が期待できる．

しかし上述の適用では，例えば復元可能なページキャッシュ 900 MB，その他のメモリ 100 MB の場合，全てのメモリを転送する所用時間は 1 GB÷ 帯域幅，提案手法での所用時間は 900MB÷ 帯域幅となり差は小さい．そこで復元可能なページキャッシュの一部を意図的に復元可能でないとして，一般通信用ネットワークとストレージ共有用のネットワークで 500MB ずつを転送する手法が考えられる．本手法は，復元可能なページキャッシュを検出する際に全使用メモリ量の半分を越えたものは復元可能でないとするだけで実装できると考えられる．

7. 関連研究

ページキャッシュに着目してライブマイグレーションを高速化する研究には，文献 [4] や文献 [5] がある．Xen の純仮想化環境ではバルーンドライバを用いて VM が VMM に未使用なメモリページを返却し，VMM はライブマイグレーション時に未使用ページの転送を省略する．VM がページキャッシュを削除しページキャッシュに割り当てられたページを VMM に返却することでライブマイグレーションを高速化する．しかしこれらの研究では削除されたページキャッシュの復元は VM が行うため，ページキャッシュに頻繁にアクセスするワークロードでは VM の性能が低下すると考えられる．一方，本論文の提案手法では VM が resume する前にページキャッシュを外部から復元するため，VM の性能低下は小さいと考えられる．具体的な比較評価は将来の課題である．

遠隔地ライブマイグレーションに着目した研究には文献 [9] や文献 [10] がある．CloudNet [9] はレイヤ 2 の VPN を用いて複数データセンタにまたがった仮想的なリソースプールを構成する．仮想リソースプール内での柔軟なリソース割り当てを実現するため，以下の手法によって遠隔地ライブマイグレーションにおけるメモリ転送を高速化する．

- pre-copy マイグレーションにおいて，1 イテレーション中に転送したメモリ量と更新されたメモリ量を比較しそれらが近くなった時点で VM の CPU を停止する．
- pre-copy マイグレーションにおいて同一メモリページを再送信する際に差分のみを転送する．
- ハッシュを用いて同一内容のメモリページを発見し転送を省略する．

また Shrinker [10] は複数の VM を同時に遠隔地ライブマイグレーションする．同じ OS が動作中の二台の VM には同一内容のメモリページが多く存在する．従って，複数 VM がマイグレーション中の場合を考えると，今から転送するあるメモリページと同一のデータが既に移動先データセンタに存在する可能性が高い．Shrinker ではこれを分散ハッシュテーブルで検知しデータセンタ間の転送をデータセンタ内の転送に振り替えることでメモリ転送を高速化する．

8. 結論と今後の課題

本論文では遠隔地ライブマイグレーションの利用例を複数挙げ，それらを実現するためには効率的な遠隔地ライブマイグレーション手法が必要であることを述べた．我々は特にページキャッシュが VM のメモリの多くを占有する場合に着目した．事前に同期されたディスクイメージからページキャッシュを復元することで WAN 越しのメモリ転送量を削減し，さらにページキャッシュの復元とその他のメモリ転送を並列に動作させることで遠隔地ライブマイグレーションを高速化した．

今後の課題は，提案手法によるマイグレーション後のディスク IO 性能低下が少ないことを実験により示すこと，およびページキャッシュが頻繁に更新される場合にロック競合が多発しマイグレーション時間が短縮できない問題の解決である．IO 性能低下が少ないことを示すことで関連研究からの優位性が示され，またロック競合を解決することで様々なワークロードが動作中の VM に提案手法を適用可能になる．

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] 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).
[4] 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).
[5] Koto, A., Yamada, H., Ohmura, K. and Kono, K.: Towards Unobtrusive VM Live Migration for Cloud Computing Platforms, *Asia-Pacific Workshop on Systems (APSys)*, pp. 7:1–7:6 (2012).
[6] 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).
[7] 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).
[8] 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).
[9] Wood, T., Ramakrishnan, K. K., Shenoy, P. and Van der Merwe, J.: CloudNet: dynamic pooling of cloud resources by live WAN migration of virtual machines, *International Conference on Virtual Execution Environments*, pp. 121–132 (2011).
[10] Riteau, P., Morin, C. and Priol, T.: Shrinker: Efficient Wide-Area Live Virtual Machine Migration using Distributed Content-Based Addressing, Technical Report RR-7198, INRIA (2010).
[11] Clark, C., Fraser, K., Hand, S., Hansen, J. G., Jul, E., Limpach, C., Pratt, I. and Warfield, A.: Live Migration of Virtual Machines, *Symposium on Networked Systems Design & Implementation*, pp. 273–286 (2005).
[12] DRBD: `http://www.drbd.org/`.
[13] 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).
[14] 穐山空道, 広渕崇宏, 高野了成, 本位田真一：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，情報処理学会研究報告，Vol. 2012-OS-123 (2012).
[15] Mosberger, D. and Jin, T.: httperf: A Tool for Measuring Web Server Performance, *Performance Evaluation Review*, Vol. 26, No. 3, pp. 31–37 (1998).
[16] bonnie++: `http://www.coker.com.au/bonnie++/`.
[17] TPC-C: `http://www.tpc.org/tpcc/`.