Vol.2021-OS-152 No.2
2021/5/27

データベースと連動する仮想マシンライブ移送

浅沼 昂太[1]　山田 浩史[1]

概要：仮想マシン (VM) のライブ移送は VM を稼働させたまま別の物理マシンへ移動させる技術である．ライブ移送を用いて VM の再配置を行うことで負荷が少ない VM をまとめることや逆に増大した VM を分散することで効率的なリソースの使用が可能となる．また，物理マシンの停止を伴うメンテナンスも VM を無停止で行うことができるため重要な役割を果たす技術である．しかし，データベース管理システム (DBMS) を含む VM の移送は長期化する傾向にある．DBMS では高速化のためにキャッシュ用に多くのメモリを割り当てるが，ライブ移送で行う処理の大半がメモリの転送処理であることから必然的に移送処理に時間がかかるためである．本研究では DBMS を含む VM のライブ移送を高速化する手法を提案する．提案手法の性能を評価するため，MySQL 5.7.30，Linux 4.18.20 及び QEMU 5.1.0 上にプロトタイプを実装した．実験結果より，移送時間が最大で 2.3 倍短縮されることがわかった．また，ライブ移送のメモリ転送処理だけでなく共有ストレージから復元する提案手法の有用性を確認した．

キーワード：仮想マシン，ライブ移送，DBMS

1. はじめに

仮想化技術とは物理マシン上でソフトウェアによりハードウェアをエミュレートして仮想マシン (VM を作り出しその上で動作するソフトウェアにあたかも 1 台の物理マシン上で動作しているかのように見せる技術である．VM は仮想マシンモニタ (VMM) によって管理さている．1 台の物理マシン内でリソースが許す限り，複数で別々のオペレーティングシステム (OS) を稼働させることができる．複数の物理マシンを 1 台の物理マシンへ統合することにより効率的なリソースの使用やトータルの消費電力の削減が期待できる．現在では Amazon EC2 [1] や，Google Compute Engine [2] などの大規模なクラウドサービスから，企業のオンプレミスレベルまで幅広く利用されている．

ライブ移送技術 [3] [4] [5] とは VM を稼働させたまま，別の物理マシンへ停止させること無く移動させる技術である．VM を物理マシンの負荷に応じて別の物理マシンへ分散，統合させること [6] [7] や物理マシンの停止を伴うメンテナンスを行うために移送すること [8] で安定稼働とリソースの節約を実現することができる．クラウドサービスでは Google が実際に使用しておりメンテナンスに伴う VM の停止を回避している [9] [10].

しかし，データベース管理システム (DBMS) を含む VM の移送は長期化する傾向にある．ライブ移送で行う処理のうち大半を占めるのはメモリの転送処理である．しかし，DBMS ではパフォーマンス向上のため多くのメモリを使用していることから，その転送処理に時間がかかり長期化する．VM には数 GiB から数 TiB までメモリを割り当てることが可能で 実際に Amazon EC2 [1] では最大で 24TiB まで割り当て可能なマシンが存在する．移送が長期化すると，移送処理による同じ物理マシン上の他の VM との資源競合，ならびに再配置ポリシーへの追従が困難となる．

本研究では DBMS を含む VM のライブ移送を高速化する手法を提案する．DBMS が使用しているメモリの大半はキャッシュであることとストレージから読み込んだ内容である点に着目して高速化する．従来のライブ移送は DBMS のキャッシュ領域に関係なくメモリの内容を転送する．提案手法では DBMS のキャッシュとして確保しているメモリ領域を移送元から転送せず，移送先でメモリのキャッシュ部分を共有ストレージから直接読み込み復元することで高速化を図る．

2. 背景

2.1 ライブ移送技術

ライブ移送技術とは VM を稼働させたまま別の物理マシンへ移動させる技術 [3] である．移送は VM を稼働させたままでネットワークのセッションは継続したままであるため外部からは別の物理マシンへ移動したことに気づかない．

[1] 東京農工大学
Tokyo University of Agriculture and Technology

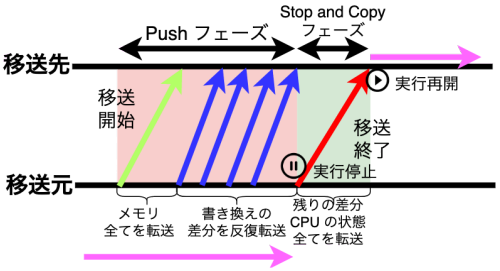


図 1: Pre-Copy 方式

ライブ移送を実現する手法として Pre-Copy 方式 [3] [11] があり内部の動作は次の 2 つのフェーズに分けることができる．

(1) Push フェーズ

(2) Stop and Copy フェーズ

Pre-Copy 方式の概要を図 1 に示す． Push フェーズでは移送元で VM を稼働させている状態である．最初は移送可能なすべてのメモリ領域を転送する．移送元では稼働したままの状態であるため，次に前回転送した時点との差分だけを転送する．差分の転送を繰り返すことで移送元と移送先でほぼ同一状態のメモリを実現することができる．ある一定以上の状態に達した時，次の Stop and Copy フェーズに移る．Stop and Copy フェーズでは移送元の VM を一時停止し Push フェーズで転送できなかったメモリ領域や CPU のレジスタなどを転送する．このフェーズの時間をダウンタイムと呼び VM が停止しているためできる限り短くすることが求められる．このフェーズを終える時には移送先に全てのデータが転送されており，移送先で VM を稼働させることでライブ移送を終える．

3. 提案

本研究では DBMS を含む VM の移送が長期化する傾向にあるという問題点に対して， DBMS と VMM が連動することで VM ライブ移送の高速化を目標とする．すべてのメモリ領域を移送元から転送せず DBMS が使用しているキャッシュのうち移送先でストレージから復元可能な領域については移送先でストレージから直接復元することで高速化を図る．提案手法の概要図について図 2 に示す．DBMS のキャッシュ情報を用いて移送元 VMM からキャッシュの大半を転送せず，移送先の VMM で共有ストレージから復元する．ライブ移送を行う範囲は LAN 内に限定する．共有ストレージを使用しているため WAN 経由ではレイテンシが高くもともとの VM の動作に大きく影響を与えることが理由である．

提案手法を実現するにあたり以下の 2 つの課題を解決する必要がある．

- DBMS のキャッシュ領域を VMM から取得できない
  DBMS が管理しているキャッシュ情報は DBMS プロセスのみが知り，外部プロセスや VMM が知ることは出来ない．
- メモリの転送はネットワークがボトルネック
  ライブ移送に伴うメモリの転送はできる限り短時間で完了することが求められる．転送に時間がかかるとその間に書き換えられた内容を再度転送する必要があり更に時間がかかるためである．その転送はネットワーク経由で行われるためネットワークの回線速度によって移送時間が左右される．回線を 1Gbps から 10Gbps へ増強するようなことが根本的な解決策ではあるが簡単には出来ない．

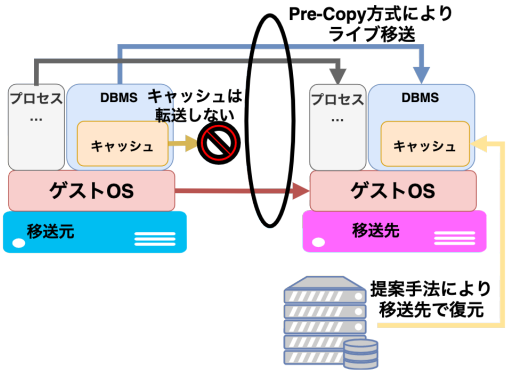


図 2: 提案手法

4. 設計

本章では提案手法に向けた設計を示す．前章で示した課題について次の 2 つの手段によって解決する．

- DBMS プロセスと VMM の通信路
- 移送先でのキャッシュ復元

4.1 DBMS プロセスと VMM の通信路

DBMS が管理しているキャッシュ情報は DBMS プロセスのみが知るため，外部から読み取ることは出来ない．そのため VMM がキャッシュ情報を取得できるよう VM 内部のプロセスから VMM へ通信可能な通信路を用意する．また，移送先 VMM で復元するためにキャッシュ情報を移送元から転送するための通信路も必要である．想定する通信路の概要を図 3 に示す．キャッシュ情報を転送するにあたり，通信路は次の 2 つが必要となる．

- DBMS プロセスと移送元 VMM の間
- 移送元と移送先 VMM の間

前者はゲスト OS 内で動作するプロセスとそれをホストする VMM の立場での通信である．VMM の機能によりキャラクタデバイスをゲスト OS 内に配置し，VMM 側ではストリームソケットを配置する．この通信路を通して VMM はキャッシュ情報を取得する．

後者はライブ移送での通信用にコマンドを送信する機能にキャッシュ情報を送信するコマンドを追加することで移送元から移送先への通信が可能になる．移送先でキャッ

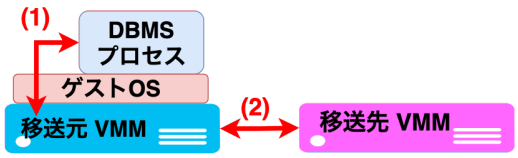


図 3: 連携時の通信路

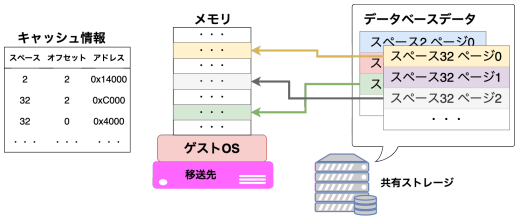


図 4: キャッシュ復元の概要図

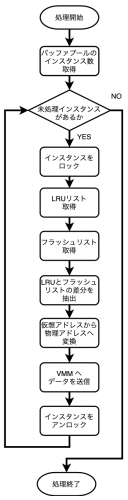


図 5: ページリストの取得フローチャート

シュの復元が完了したことを移送元へ通知することも必要であるため双方向の通信路を確保した

4.2 移送先でのキャッシュ復元

キャッシュ復元の概要図を図 4 に示す．移送先の VMM ではキャッシュ情報が移送元から転送されてくるのでその情報を元に復元を行う．DBMS のデータは共有ストレージを介してアクセスする．従来ではライブ移送用の回線のみを用いて転送を行っていたが，共有ストレージ用の回線を用いてキャッシュ領域を復元することでボトルネックを改善することが可能となる．

今後の課題として以下の 2 点が挙げられる．

- 転送のボトルネック
- キャッシュ領域の転送最適化

5. 実装

本章では提案手法の実装について述べる．実装は VMM として QEMU 5.1.0 [12]，ゲスト OS は Linux 4.18.20，DBMS は MySQL 5.7.30 [13]，ストレージエンジンは標準の InnoDB を使用する．

5.1 InnoDB のキャッシュ情報取得

InnoDB が管理しているキャッシュ情報を取得し，VMM で使用できるように変換する．キャッシュ情報として必要なデータは何のキャッシュであるかとどこに格納されているかのアドレスである．前者はどのテーブルのキャッシュかを表すスペース番号とそのテーブルデータの何番目かを表すオフセットである．後者のアドレスは物理アドレスである．リストを取得する手順を図 5 に示す．また，InnoDB が管理しているキャッシュ情報は LRU リストから完全なページ一覧を，フラッシュリストからディスク上とデータが異なるページ一覧を得ることができ，その差分を抽出することで移送先で復元可能なページリストを作成する．ページの情報としてそのページのスペース番号やオフセット情報及びページのアドレスが含まれる．しかし MySQL はユーザ空間で動作するプロセスでありそのままで得られるページのアドレス情報はそのプロセス内だけで有効な仮想アドレスであり，VMM から特定の位置のメモリへアクセスする際には物理アドレスが必要となる．別途 Linux カーネルに仮想アドレスから物理アドレスへ変換する機構を追加することで実現する．

5.2 プロセスの仮想アドレスと物理アドレスの変換

ゲスト OS 内のプロセスで得られるアドレスはそのプロセス内だけで有効な仮想アドレスである．その仮想アドレスを Linux カーネル内で物理メモリ上のアドレスに変換を行うシステムコールを追加する． 64bit の Linux では 4 段のページテーブルを順に追うことで物理アドレスへ変換するページテーブルウォークという仕組みとなっている．基本的にはページテーブルを順に追うことで仮想アドレスを物理アドレスへ変換する．Linux には Transparent Huge Page(THP) と呼ばれるページサイズが大きい Huge Page を用いてメモリを有効利用する機能がある．この機能によりメモリが再配置されると 4 段のページテーブルウォークでは追うことができないため，別途 Huge Page 向けに処理を行うことで物理メモリアドレスの変換を実現した．

5.3 移送元の転送取り消しと移送先でのキャッシュ復元

キャッシュは移送先の VMM で復元するため，移送元から転送しないように qemu-kvm のライブ移送機構で転送済みか否かを判定するビットマップを利用する．ビットマップはライブ移送時に現時点でのメモリ状態が転送済みかどうかを判定するメモリのページごとに存在するマップである．初期化時にすべてをダーティとセットしておき，メモリを転送するときにクリアすることでどのメモリページが転送済みかどうかを把握する事ができる．今回は DBMS のキャッシュとして使用されているメモリ領域は

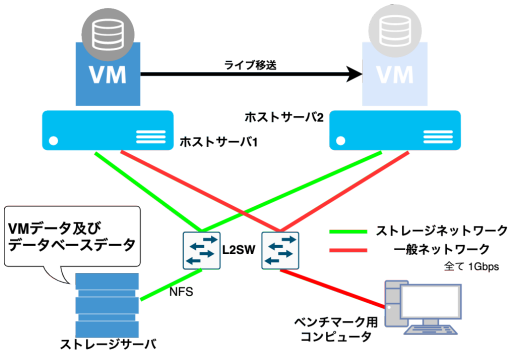


図 6: ネットワーク構成

ライブ移送直前に転送済みであるとビットマップをクリアすることでキャッシュ部分の転送を取り消す．移送先ではキャッシュを復元するために共有ストレージに保存されているデータベースの元データにアクセスし，適切なメモリ位置へ復元する．まずはデータベースのデータが保存されている共有ストレージをマウントする．マウント後，転送されたキャッシュ情報を元に復元する．InnoDB ではテーブルごとに 1 ファイルで保存されているため，スペース番号をもとにファイルを特定しオフセットを元にファイル内の位置を特定する．復元先の VM のメモリは qemu-kvm により *pc.ram* という名称で管理されている．その実態は qemu-kvm プロセスのメモリ領域であるため特殊な操作なく読み書きが可能である．先ほど特定したファイルの領域からデータを読み出しコピーすることで復元が可能となる．すべてのキャッシュ情報を処理し終えることで復元が完了する．

6. 評価実験

6.1 実験環境

評価実験に使用するサーバマシンの構成を表 1 に示す．VM をホストするマシンは同構成を 2 台，VM とデータベースのストレージとして NFS によるサーバを 1 台構築した． ネットワーク構成は図 6 に示し回線速度は全て 1Gbps で接続されている． 提案手法は Linux カーネル 4.18.20，MySQL 5.7.30，QEMU 5.1.0 上に実装した． ベンチマークソフトウェアは Sysbench 1.0.17 [14] を使用し継続的な負荷を DBMS へかけ続けることで性能の指標となる毎秒あたりのトランザクション数を算出するものである．同じネットワーク内に接続したコンピュータで実行する．

6.2 実験方法

従来手法によるライブ移送における DBMS への影響を把握し，提案手法によるライブ移送の高速化について検証を行う．実験では VM に vCPU を 2 コア割り当て，ストレージは NFS ストレージ上に 20GB，メモリは 4GB，8GB，16GB，24GB と変化させた．VM 割り当てメモリに応じてバッファプールの領域は約 8 割とする．VM のメモリ割り当て量に応じたバッファプールの割り当て量を表 2 に示す．

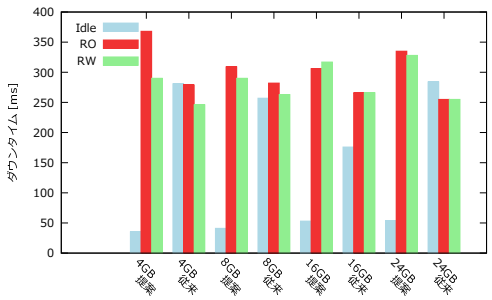


図 7: ダウンタイム

テストはメモリごとに表 3 の 3 種類のパターンを従来手法と提案手法を用いて行う． DBMS のバッファプールのキャッシュをウォームアップした状態で何も負荷をかけない状態 (Idle)，負荷をかけるベンチマークとして Sysbench のデータベース向けシナリオである oltp_read_only (RO) と oltp_read_write (RW) の 2 種類を使用した． RO は読み出しのみ，RW は読み書きの負荷をかけることができる．実行の手順はまずデータベースのバッファプールにテーブルのデータ全てを載せることでバッファプールをウォームアップする．Idle ではライブ移送を開始する．RO 及び RW では Sysbench を動かし始めた 300 秒後にライブ移送を開始する．Sysbench は合計で 900 秒間動かす．

6.3 実験結果と考察

6.3.1 ダウンタイムについて

複数のメモリサイズでパターンを試した結果，ダウンタイムを図 7 に示す．ダウンタイムは短いほど VM への影響が少なくて済むがパターンによって大きな違いは生まれなかった．今回の提案手法ではメモリの転送の高速化に重きをおいているため短くなることはなく，ダウンタイム時の処理を増やしていないため，伸びることもなかった．今回の提案手法によるダウンタイムの劣化はないといえる．

6.3.2 総移送時間について

総移送時間を図 8 に示す．総移送時間はどのパターンにおいても提案手法により短縮された．割り当てメモリが多い VM ほど短縮の効果は高く，少なくとも 1.3 倍から最大で 2.3 倍の高速化が実現されたため，提案手法の目的である DBMS を含む VM のライブ移送の高速化は達成された．

ネットワークのスループットを図 9，10，11 に示す．ネットワークのスループットによると，従来手法及び提案手法において QEMU のライブ移送に使用する回線はどちらも帯域を目一杯使用していることから回線がライブ移送のボトルネックであることがわかる．提案手法では共有ストレージ用の回線も併用することでボトルネックを改善

表 1: 実験マシン環境

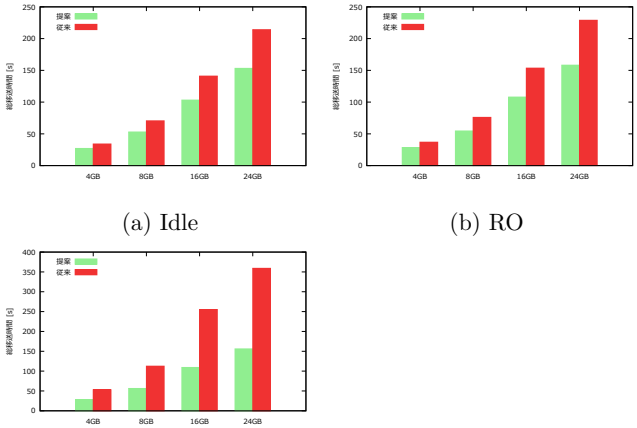
項目	qemu-kvm ホストマシン	ストレージマシン
CPU	Intel Xeon E5-2630v3 (2.40GHz 8C/16T 20MB キャッシュ)	Intel Xeon E5-2630v2 (2.60GHz 6C/12T 15MB キャッシュ)
メモリ	32GB(DDR3 UDIMM 1866MHz)	48GB(DDR3 RDIMM 1600MHz)
HDD	500GB	RAID 500GB

表 2: メモリ量に応じたバッファ領域の割り当て

VM 割り当てメモリ [GiB]	バッファプールサイズ [GiB]	バッファプール実際の使用量 [GiB]
4	3.2	2.8
8	6.4	5.5
16	12.8	11.3
24	19.2	16.2

表 3: 実験シナリオ

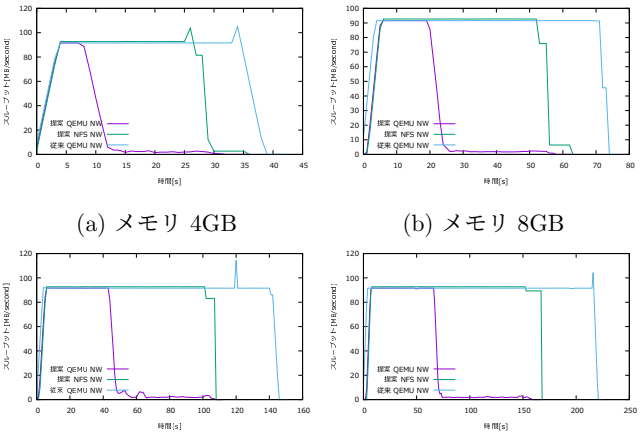
パターン	手法	シナリオ
Idle	従来	ベンチマークなし
RO	従来	oltp_read_only
RW	従来	oltp_read_write



(a) Idle (b) RO

(c) RW

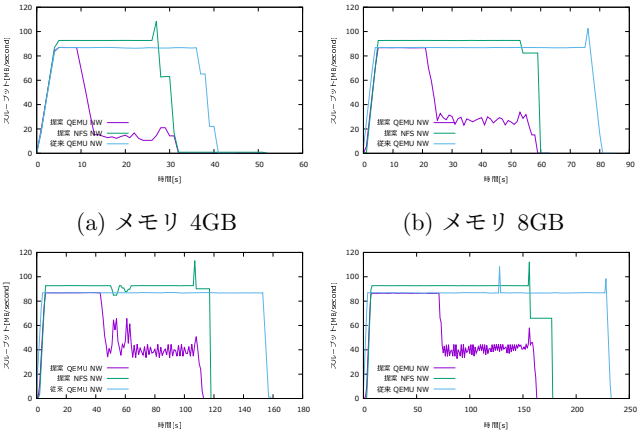
図 8: 総移送時間



(a) メモリ 4GB (b) メモリ 8GB

(c) メモリ 16GB (d) メモリ 24GB

図 9: ネットワークスループット (Idle)



(a) メモリ 4GB (b) メモリ 8GB

(c) メモリ 16GB (d) メモリ 24GB

図 10: ネットワークスループット (RO)

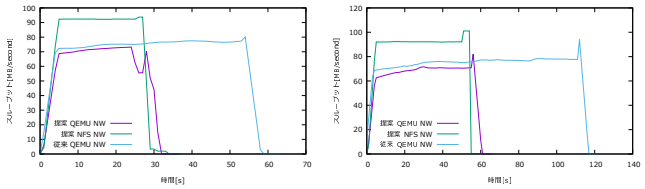
された．ただし，QEMU のライブ移送よりも移送先での復元に時間がかかり，逆にボトルネックであることがわかる．ネットワーク帯域幅の増強が必要不可欠であると考えられる．

6.3.3 メモリの転送について

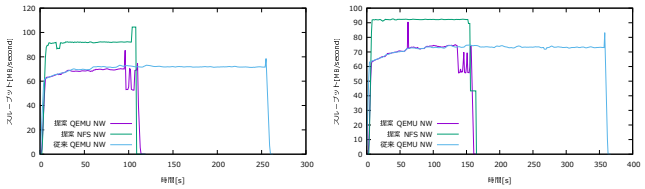
qemu-kvm における未転送メモリの残りについてのグラフを図 12，13，14 に，ライブ移送に伴う総転送量のグラフを図 15 に示す．提案手法ではキャッシュ領域分のメモリは転送しないため初期段階で転送すべき量は大幅に少ないことがわかる．初回の転送が短時間で終わるため，次以降の転送量を少なく抑えることができている．ただし，総転送量でみると従来手法に比べ提案手法のほうが増加傾向にある．提案手法では移送先での復元に時間がかかっており，その待ち時間の間に書き換わるデータを逐一転送しているため，転送量が増えたと考えられる．

6.3.4 ベンチマークについて

Sysbench を用いたベンチマーク結果について図 16，17 に示す．開始 300 秒秒後にライブ移送を開始するためどの状況においても大幅なスループットの低下が，ライブ移送終了前後のダウンタイムによる多少のスループット低下が見受けられる．ライブ移送による DBMS への影響は大きいことがわかる．従来手法では長期間にわたり低下が続くが，提案手法では移送開始直後は同様に低下するが回復は大幅に早い事がわかる．早い段階である程度回復する点に

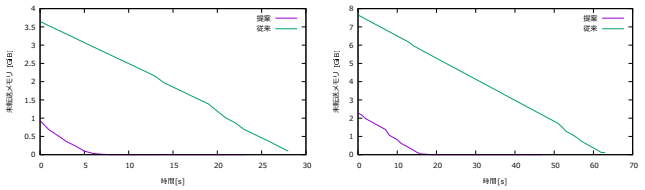


(a) メモリ 4GB　(b) メモリ 8GB

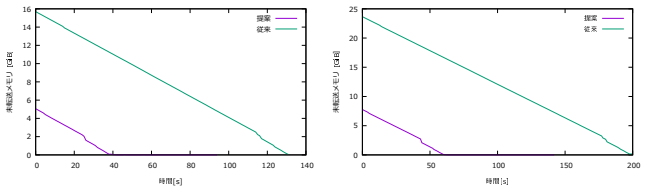


(c) メモリ 16GB　(d) メモリ 24GB

図 11: ネットワークスループット (RW)

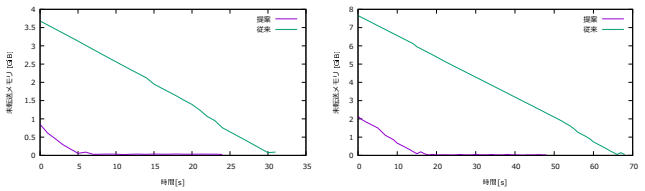


(a) メモリ 4GB　(b) メモリ 8GB

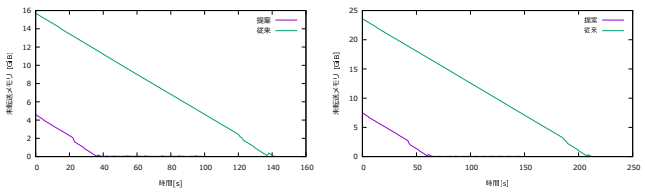


(c) メモリ 16GB　(d) メモリ 24GB

図 12: 未転送メモリ (IDLE)



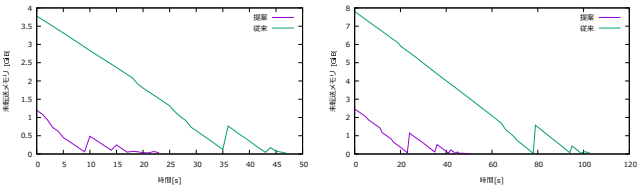
(a) メモリ 4GB　(b) メモリ 8GB



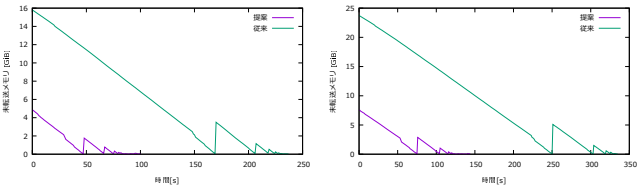
(c) メモリ 16GB　(d) メモリ 24GB

図 13: 未転送メモリ (RO)

ついては移送元から移送先へメモリを転送する処理が終わるとともに回復するため提案手法による改善であると考えられる．また総移送時間が短くなることでスループットの低下をできる限り抑えられることがわかる．



(a) メモリ 4GB　(b) メモリ 8GB



(c) メモリ 16GB　(d) メモリ 24GB

図 14: 未転送メモリ (RW)

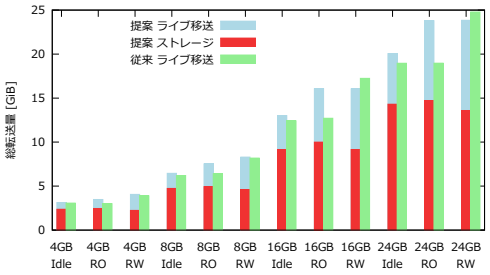
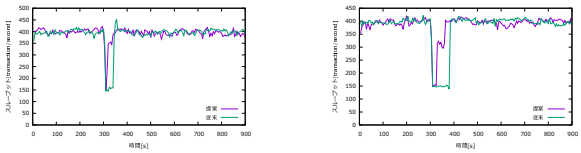
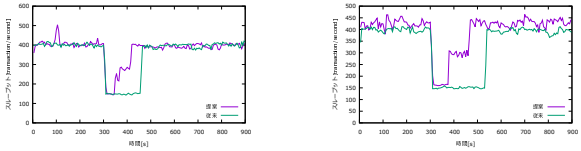


図 15: 総転送量の合計



(a) メモリ 4GB　(b) メモリ 8GB

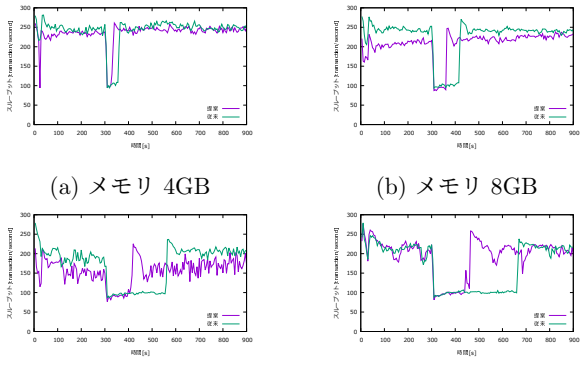


(c) メモリ 16GB　(d) メモリ 24GB

図 16: Sysbench TPS (RO)

6.3.5 DBMS のキャッシュ領域の取得について

キャッシュ領域のリストを取得にかかる時間について図 18 に示す．一番時間のかかる処理は VMM へリストを転送する処理である．大量のリストを転送するにあたりキャラクタデバイスを経由する転送処理がボトルネックになっていると推測される．キャラクタデバイスを用いない別の手法を考案する必要があると考える．次に時間のかかる処理は仮想アドレスを物理アドレスへ変換する処理である．この処理はシステムコールを用いて変換することから呼び出しコストが高いと推測される．ただし，これらの時間



(a) メモリ 4GB　(b) メモリ 8GB

(c) メモリ 16GB　(d) メモリ 24GB

図 17: Sysbench TPS (RW)

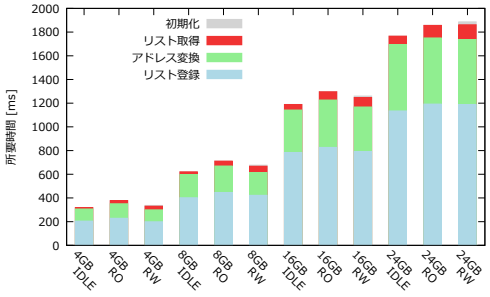


図 18: キャッシュ領域の取得にかかる時間

は 8 個のインスタンスにかかるトータルの時間であるため，ひとつあたりのインスタンスではわずかな時間でありDBMS への影響は少ないと考えられる．事実，図 16，17 のベンチマーク結果からも移送処理開始時の落ち込みはほぼ見受けられない．キャッシュ領域が増加した場合ではインスタンス数を増加させることで影響を最小限に抑えることができると考えられる．

7. 関連研究

7.1 Albatross

Albatross [15] は共有プロセスモデル [16] [17] を用いたデータベースプロセスのみを Pre-Copy 方式を用いてライブ移送を実現したものである．この手法が目標としていることはライブ移送中のダウンタイムを最小限に抑えることである．今まではデータベースを移送するには一度停止し移送先で起動する方法しかなく，従来のライブ移送ではVM 自体でありデータベースのみのライブ移送は存在しなかった．ダウンタイムを抑えたライブ移送を実現することでデータベースのサービス品質を低下させることなく運用コストを抑えることができる．本論文では DBMS を含む VM 全体のライブ移送である点が異なる．

7.2 Zephyer

Zephyer [18] は Pre-Copy 方式を用いて共有ストレージを使用しないデータベースプロセスのみのライブ移送である．この手法が目標としていることは共有ストレージを用いないライブ移送でダウンタイムを最小限に抑えることである．移送元と移送先で同時にトランザクションを処理するデュアルモードを追加することでダウンタイムを抑制する．本論文では DBMS を含む VM 全体のライブ移送であり共有ストレージを用いることで高速化を図る点が異なる．

7.3 Java-Aware VM Migration

Java-Aware VM Migration (JAVMM) [19] は VM 内の Java 仮想マシンと VMM が連携し Pre-Copy 方式を用いたライブ移送を高速化する手法である．この手法が目標としているのはライブ移送における総転送量の削減と移送時間の短縮である．Java 仮想マシンが使用するメモリの特徴を活かし転送量を削減する．Java アプリケーションには手を加えず実行する Java 仮想マシンに対して適用することで既存のアプリケーション資産を活かすことができる．本論文では DBMS と VMM が連携する点で異なる．

8. おわりに

本研究では DBMS を含む VM の移送が長期化する傾向にある問題点に対して， DBMS のキャッシュ領域を転送せず移送先で復元するライブ移送を提案した．提案手法を用いることで複数のパターンにおいても移送時間の短縮を実現した．提案手法では qemu-kvm での転送処理を大幅に減らし全てのデータを移送元から転送するのではなく一部は移送先で復元可能であることが明らかとなった．従来のライブ移送では使用されていなかった共有ストレージネットワークも有効活用するという新たな手法の有用性が確認された．

参考文献

[1] Amazon Web Service: Amazon EC2, `https://aws.amazon.com/jp/ec2/`. (accessed 2021-01-02).

[2] Google Cloud: Compute Engine, `https://cloud.google.com/compute`. (accessed 2021-01-02).

[3] Clark, C., Fraser, K., Hand, S., Hansen, J. G., Jul, E., Limpach, C., Pratt, I. and Warfield, A.: Live migration of virtual machines, *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, USENIX Association, pp. 273–286 (online), DOI: 10.5555/1251203.1251223 (2005).

[4] Nelson, M., Lim, B. H. and Hutchins, G.: Fast transparent migration for virtual machines, *Proceedings of the annual conference on USENIX Annual Technical Conference*, pp. 391–394 (online), DOI: 10.5555/1247360.1247385 (2005).

[5] Hines, M. R., Deshpande, U. and Gopalan, K.: Post-copy live migration of virtual machines, *Jounarl of ACM SIGOPS Operating Systems Review*, Vol. 43, No. 3, pp. 14–26 (online), DOI: 10.1145/1618525.1618528 (2009).

[6] do Lago, D. G., Madeira, E. R. M. and Bittencourt, L. F.: Power-aware virtual machine scheduling on clouds using active cooling control and DVFS, *Proceedings of the*

9th International Workshop on Middleware for Grids, Clouds and e-Science, New York, New York, USA, ACM Press, p. 2 (online), DOI: 10.1145/2089002.2089004 (2011).

[7] Yang, C.-T., Liu, J.-C., Chen, S.-T. and Huang, K.-L.: Virtual machine management system based on the power saving algorithm in cloud, *Journal of Network and Computer Applications*, Vol. 80, pp. 165–180 (online), DOI: 10.1016/j.jnca.2016.11.026 (2017).

[8] Zhang, X., Zheng, X., Wang, Z., Li, Q., Fu, J., Zhang, Y. and Shen, Y.: Fast and Scalable VMM Live Upgrade in Large Cloud Infrastructure, *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, New York, NY, USA, ACM, pp. 93–105 (online), DOI: 10.1145/3297858.3304034 (2019).

[9] Google Cloud: ライブマイグレーション, `https://cloud.google.com/compute/docs/instances/live-migration`. (accessed 2021-01-02).

[10] Ruprecht, A., Jones, D., Shiraev, D., Harmon, G., Spivak, M., Krebs, M., Baker-Harvey, M. and Sanderson, T.: VM Live Migration At Scale, *Journal of ACM SIGPLAN Notices*, Vol. 53, No. 3, pp. 45–56 (online), DOI: 10.1145/3186411.3186415 (2018).

[11] Ibrahim, K. Z., Hofmeyr, S., Iancu, C. and Roman, E.: Optimized pre-copy live migration for memory intensive applications, *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, New York, New York, USA, ACM Press, p. 40 (online), DOI: 10.1145/2063384.2063437 (2011).

[12] QEMU: QEMU, `https://www.qemu.org/`. (accessed 2021-01-02).

[13] Oracle Corporation: MySQL, `https://www.mysql.com`. (accessed 2021-01-02).

[14] Alexey Kopytov: Sysbench, `https://github.com/akopytov/sysbench`. (accessed 2021-01-02).

[15] Das, S., Nishimura, S., Agrawal, D. and El Abbadi, A.: Albatross: Lightweight elasticity in shared storage databases for the cloud using live data migration, *Proceedings of the VLDB Endowment*, Vol. 4, No. 8, pp. 494–505 (online), DOI: 10.14778/2002974.2002977 (2011).

[16] Das, S., Agrawal, D. and El Abbadi, A.: ElasTraS: An elastic, scalable, and self-managing transactional database for the cloud, *Journal of ACM Transactions on Database Systems*, Vol. 38, No. 1, p. 5 (online), DOI: 10.1145/2445583.2445588 (2013).

[17] Bernstein, P. A., Cseri, I., Dani, N., Ellis, N., Kalhan, A., Kakivaya, G., Lomet, D. B., Manne, R., Novik, L. and Talius, T.: Adapting microsoft SQL server for cloud computing, *Proceedings of the IEEE 27th International Conference on Data Engineering*, IEEE, pp. 1255–1263 (online), DOI: 10.1109/ICDE.2011.5767935 (2011).

[18] Elmore, A. J., Das, S., Agrawal, D. and El Abbadi, A.: Zephyr: Live migration in shared nothing databases for elastic cloud platforms, *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pp. 301–312 (online), DOI: 10.1145/1989323.1989356 (2011).

[19] Hou, K. Y., Shin, K. G. and Sung, J. L.: Application-assisted live migration of virtual machines with Java applications, *Proceedings of the 10th European Conference on Computer Systems*, pp. 1–15 (online), DOI: 10.1145/2741948.2741950 (2015).