様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

令和　２ 年　 ６ 月 １６ 日現在

機関番号：１２６０５
研究種目：基盤研究(C)（一般）
研究期間：2017～2019
課題番号：１７Ｋ００９４

研究課題名（和文）ビッグメモリDBMSを考慮した仮想マシン移送に関する研究

研究課題名（英文）Live Virtual Machine Migration for Big Memory Databases

研究代表者
山田　浩史（Yamada, Hiroshi）

東京農工大学・工学（系）研究科（研究院）・准教授

研究者番号：００５７１５７２

交付決定額（研究期間全体）：（直接経費）　　3,500,000円

研究成果の概要（和文）：本応募課題では，大量のメモリを活用しているデータベース管理システム(DBMS)を稼動した仮想マシン(VM)を稼動させたまま移送する方式について研究する．DBMS レイヤのメモリ管理機構と VM モニタのメモリ管理機構とを連動させることによって，総移送時間の短縮を狙う．結果として，仮想マシン移送による VM 再配置を従来通り可能にし，負荷分散やハードウェア管理の容易さ，VM 再配置による消費電力削減といった大きな恩恵を，ビッグメモリ DBMS が稼働する環境にもたらす．プロトタイプを Xen 4.4.2, Linux 3.18.20, MySQL 5.6.26 上に実装し，定量的評価を行った．

研究成果の学術的意義や社会的意義
当該分野における応募課題の学術的意義は 1).DBMS と VM モニタとを連動させる手法を示している点，2)．提案方式をソフトウェアとして実現する点，3)．提案の有効性を定量的に評価している点などにある．社会的意義は，本方式によって，今後も重要となるビッグメモリDBMSが稼働する状況においてもVM移送を用いた VM の再配置を可能にし，現行のデータセンタの効率的な管理に導くことに寄与する．また，VM モニタとDBMS が連動可能であることを示し，新たなデザインスペースの提示することで当該分野の学術分野，産業分野を活性化していく．

研究成果の概要（英文）：Live migration, which enables us to move running VMs between different physical hosts, is useful for administrative tasks such as physical host maintenance, load balancing, and energy consumption. Although it is desirable for administrators that the live migration completes as quick as possible, the widely used live migration scheme, named Pre-copy, does not satisfy this demand on the current trend that VMs on which database management systems (DBMSs) have a large amount of memory. To address this issue, we explore an approach to shortening total time for migrating VMs running a DBMS. Our approach exploits DBMS-level resource management information to build the VM states on the destination by transferring the VM memory on the source and fetching DB items from the shared storage in parallel. We prototyped database-assisted live migration on Xen 4.4.2, Linux 3.18.20, and MySQL 5.6.26 and evaluated the prototype.

研究分野：オペレーティングシステム

キーワード：仮想マシン移送　データベース管理システム　仮想化技術

様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９（共通）

１．研究開始当初の背景

ビッグデータという言葉が一般的となった今日において，データベース管理システム(DBMS)が担う役割は日に日に大きくなってきている．近年の DBMS は，性能を稼ぐために，数 GB から数百 GB のメモリを確保してその領域内にデータを配備して処理を進めるという特徴がある．このような大量のメモリを利用する DBMS をビッグメモリ DBMS と呼ぶ．たとえば，MySQL や PostgreSQL といった伝統的な関係 DBMS においても，バッファプールと呼ばれるデータキャッシュをメモリ上に大量に展開して低速なディスクアクセスを避け，可能な限りメモリアクセスを増やそうとする．また，RocksDB や LevelDB などの NoSQL 型 DBMS においても，Log-structured merged-tree などのデータ構造を用いて，更新が発生した際にも低速なディスクアクセスを最小限に抑える工夫が施されている．昨今，宇宙観測データなどのセンシングデータが増加しており，また各企業が有するビッグデータを複数企業で共有する動きもあることから，ビッグデータ処理の実施が閉じた環境ではなく，複数ユーザで大規模並列処理が可能なクラウド環境へと移行しつつある

２．研究の目的

本応募課題では，ビッグメモリ DBMS が稼動している仮想マシン(VM)を移送する方式について研究する．VM 移送は VM を稼働させたまま別の物理ホストに移動できる技術であり，VM の再配置を可能にする．これによって負荷分散や消費電力削減，物理マシンのメンテナンスが容易となるなどのメリットがあり，クラウド環境を構成するデータセンタの円滑な管理を促す必須の技術である．実際に Google などのデータセンタでは仮想マシン移送を使って大量の物理サーバや VM を管理している．VM 移送は VM の状態を維持するためにメモリの転送が必須であるため，既存手法は数 GB 程度の仮想マシン(VM) を想定しており，ビッグメモリ DBMS が扱うメモリを搭載する VM の移送は苦手とするところである．具体的には，1). 仮想マシン移送に要する時間が長くなりこれまでのような迅速な再配置が困難になる，2). 移送時間が長くなることに伴って CPU 使用量が増大してしまい移送に関わるホスト上で稼働している VM の性能を劣化させる，という問題が生じる．これらによって，VM の再配置を用いたデータセンタの管理が従来通りにできず，VM 移送を用いたデータセンタの管理が困難になってしまう．応募課題では上記 2 点の問題点を克服し，DBMS を搭載する VM が稼働する環境においても，仮想 VM 移送による恩恵を享受できるようにし，円滑なビッグデータ利活用を促進する．

実現する移送方式では，これまで独立性の高かった DBMS レイヤと VM モニタレイヤのメモリ管理を互いに協調させることによって総移送時間の短縮を試みる．本研究では，ビッグメモリ DBMS のメモリは多くの領域がバッファプールで支配されており，また，バッファプールはディスク上の DB ファイルから構築可能であることに着目する．移送先の VM モニタでディスク上の DB ファイルからバッファプールを構築し，それと同時にその他のメモリ領域を移送元から転送することで，VM の構築を並列化する．これにより従来の VM モニタだけに頼った移送方式での性能限界を打ち破る．しかしながら，通常，VM モニタは計算機資源の低レベルな情報しか扱えず，VM に割り当てたメモリページがどのような用途で利用されているか，またディスク上にどのファイルがあるかといった情報は把握できない．提案する移送方式では，こうしたギャップを埋めるべく，DBMS が能動的に VM モニタにメモリ情報およびファイル情報を提供することで，VM モニタ上で高レベルな情報を扱えるようにする．

３．研究の方法

課題期間内には，DBMS が稼動する VM 移送方式をソフトウェアとして実現し，実際にアプリケーションを稼働させて定量的にその有効性を明らかにする．具体的には 1. 提案方式の実現性，2. Real-world なソフトウェアスタックに対する有効性，3. 実践的なワークロードに対する有効性を示す．提案方式の実現性は，既存のオープンソースソフトウェアを拡張する形で提案方式を設計/実装することで達成する．Real-world なソフトウェアスタックに対する有効性は，広く用いられているオープンソースソフトウェアを拡張することで示す．具体的には，VM モニタとして Amazon EC2 といった実際のクラウド環境で採用実績のある Xen を，DBMS として著名な Relational DBMS のひとつである MySQL を，OS として Linux を採用し，これらの上で提案方式のプロトタイピングを行なう．実践的なワークロードに対する有効性は，様々なワークロードをプロトタイプに適用し定量的に評価することで示す．読み込みや一部テーブルの更新といった人工的なワークロードだけではなく，TPC-C などの実環境を模したワークロードを与え，その際の移送時間や資源使用率の変化を計測し，提案方式の利害得失を定量的に明らかにする．

４．研究成果

提案手法を Xen 4.4.2, Linux 3.18.20, MySQL 5.6.26 に対して設計，実装を行いプロトタイプの実現に成功した．具体的には次のとおりである．テクニカルなチャレンジは 1.Semantic Gaps を埋めるために各レイヤを繋ぐこと，2.DB キャッシュのページ転送量を削減するために移送アルゴリズムの変更を行うという点である．DBMS から移送元の VMM に対して DB で使用しているメモリ領域を伝えている（Enlightened DB Memory）．これによって移送プロセスは DB キャッシュのページ転送を省略できるようになる．VMM は渡された情報を PFN リストとして保存し，移送プロセスによって使用される．尚ディスク上のデータと DB キャッシュのデータが一致している読み込みの命令によって得られた PFN リスト（R-PFN リスト）と，ディスク上のデータと DB キャッシュのデータが一致しているとは限らない書き込みの命令によって得られた PFN リスト（W-PFN リスト）は区別して保存する．移送プロセスは移送が開始されたら保存された PFN リストを使用し，DB キャッシュのページ転送を省略できるようになる．

また，VMM は VM 内で稼働している DBMS のデータがどのようにディスク上に配置されているかを知ることができない．そのため本研究では，DBMS のみが知っている DB データのディスク上のレイアウトを移送先の VMM に伝えている（Enlightened DB Storage）．これによって移送プロセスは転送が省略された DB キャッシュに当たるページを知ることができる．しかし VMM は DBMS のストレージのレイアウトはわからないため，DB キャッシュ復元機構に DB キャッシュの復元を依頼する．移送先ホストの移送プロセスを実行する特権 VM 上の DB キャッシュ復元機構は，あらかじめ共有ストレージから DB キャッシュを復元するために共有メモリを作る．
移送プロセスは移送元の VMM から送られてきた PFN リストの情報を元に，DB キャッシュ復元機構に対して DB キャッシュの復元を依頼する．DB キャッシュ復元機構は，移送 VM の稼働させている DBMS のダンプファイルから DB キャッシュを復元する．そのときに共有メモリに DB キャッシュのデータを書き込む．これによって移送プロセスは共有メモリから DB キャッシュに当たるページを読み込むことが可能になり，コピー回数を減らすことができる．

移送元ホストは，1batch 分のページを転送する前にどのページを転送しないかを移送先ホストへ伝えてから，DB キャッシュのページ転送を省略する．まずどのページが DB キャッシュのページで，転送を省略したのかが移送先ホストにもわかるように移送先ホストへ転送しない DB キャッシュのページを伝える．移送プロセスは PFN リストを管理していないため，VMM にハイパーコールで PFN リストを要求する．移送プロセスはページ転送の前に PFN リストを移送先ホストへ転送する．次にページ転送量を削減するために DB キャッシュのページ転送を省略する．
DB キャッシュのページ転送をするときは，PFN リストを参照し，PFN リストの PFN と一致したページを DB キャッシュのページとして転送を省略する.PFN リストのサイズはページサイズの 4KB よりも小さいため，ネットワークを使用した転送量を削減できる．以上の流れを 1 回目のイテレーション中，batch 分の転送ごとに繰り返す．2 回目以降のイテレーションは，VM 内のページに変更があった場合にのみ発生するため，PFN リストを転送する必要がなく，また変更があったページは全て転送する必要がある．

移送先ホストでは，batch 分のページを読み取る前に移送元ホストから送られた PFN リストを読み取り，転送が省略された DB キャッシュを共有ストレージから復元する． まず転送された PFN リストを読み取る．その PFN リストの情報から DB キャッシュ復元機構に DB キャッシュの復元を依頼する．DB キャッシュ復元機構はあらかじめ用意されている共有メモリに DB キャッシュを保存する．先に DB キャッシュを復元しておくことで，DB キャッシュのページを VM にあてはめるときの DB キャッシュの復元がボトルネックになることを防いでいる．VM にページを当てはめていく段階では，DB キャッシュのページについては共有メモリから読み込みあてはめている．もし DB キャッシュ復元機構が DB キャッシュの復元が出来ていない場合，同期を取って DB キャッシュの復元を待つ．それ以外のページは通常のページ転送によってあてはめている．

プロトタイプの有効性を検証するために定量的な実験も実施した．ワークロードを実行している MySQL を稼働させている VM の移送を行った．ワークロードは，MySQL に DB キャッシュだけ復元させた Idle，Sysbench RO，Sysbench RW，TPC-C で，データは全て DB キャッシュに収まるものとする．提案手法は Xen のデフォルトの移送方式に比べ，総移送時間とページ転送量が小さくなっていることが確認できた．VM が 8GB のときに総移送時間は Idle，Sysbench RO，Sysbench RW，TPC-C それぞれのワークロードにおいて，51%，57%，60%，53%小さくすることができた．
ページの転送量はそれぞれ 76%，78%，71%，68%削減することができた．書き込みワークロードのときはページの書き換えが速く，Xen のデフォルトの移送方式はイテレーションの最大回数まで実行してから Stop-and-copy に移行してしまった．提案手法はフェイズ移行の指標を追加したことにより，無駄なページ転送と DB キャッシュの転送を省略し，総移送時間とページ転送量

を削減することができた.

提案方式のオーバヘッドを確認するためにワークロードが Sysbench RO と Sysbench RW のときの MySQL のスループットも計測した. Sysbench RO の場合, Xen のデフォルトの移送方式とスループットに差はなかった. これは一度読み込んだページはハイパーコールを利用しないためである. Sysbench RW の場合, Xen のデフォルトの移送方式よりもスループットが小さくなった. これは移送中もページの書き換えが起こり, ハイパーコールを呼んでしまっているからである. しかし移送開始後に書き換えが起こったページは再度転送するため, 移送中は書き換えが起きてもハイパーコールを利用しないことで改善が見込める.

５．主な発表論文等

〔雑誌論文〕　計2件（うち査読付論文　1件／うち国際共著　0件／うちオープンアクセス　0件）

１．著者名 清水祐太郎，山田浩史	４．巻 2018-OS-144
２．論文標題 ページ共有を利用したMulti-Variant監視効率化手法	５．発行年 2018年
３．雑誌名 研究報告システムソフトウェアとオペレーティング・システム（OS）	６．最初と最後の頁 1 - 13
掲載論文のDOI（デジタルオブジェクト識別子） なし	査読の有無 無
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

１．著者名 Ken Terada, Hiroshi Yamada	４．巻 E101.D
２．論文標題 Shortening Downtime of Reboot-Based Kernel Updates Using Dwarf	５．発行年 2018年
３．雑誌名 IEICE Transactions on Information and Systems	６．最初と最後の頁 2991-3004
掲載論文のDOI（デジタルオブジェクト識別子） https://doi.org/10.1587/transinf.2017EDP7397	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

〔学会発表〕　計0件

〔図書〕　計0件

〔産業財産権〕

〔その他〕

－

６．研究組織

	氏名 （ローマ字氏名） （研究者番号）	所属研究機関・部局・職 （機関番号）	備考