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　５ 年　 ６ 月 １４ 日現在

機関番号：１２６０５
研究種目：基盤研究(C)（一般）
研究期間：2020～2022
課題番号：２０Ｋ１１７４２

研究課題名（和文）インメモリDB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 In-memory DBs

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）　　3,300,000円

研究成果の概要（和文）：本応募課題では，インメモリ DB 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．従来，独立性の高かったインメモリ DB 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を達成する．結果として，VM 再配置によるメリットである負荷分散や消費電力削減といった恩恵をインメモリ DB が稼働する VM にもたらす．研究を通して，提案方式をソフトウェアとして実現し，シンプルなワークロードおよび実践的なワークロードを用いて，定量的に方式の有効性を示した．

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な特色，独創的な点は 1).インメモリDB 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．本方式によって，AIサービスのキーであるインメモリDB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，データセンタの効率的な管理に導く．また，VM モニタとインメモリDB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研究領域を活性化していく．

研究成果の概要（英文）：This project aims at shortening the time for live-migrating virtual machines (VMs) with even large memory DBMSes. Live migration of VMs is a technique that moves active VMs between different physical hosts without losing any running states. Although it is desirable for administrators that the live migration is completed as quickly as possible, the pre-copy-based live migration, widely used in modern hypervisors, does not satisfy this demand on the current trend that VMs on which running applications are performance-critical such as DBMSes have quite large memory. To quickly produce the running state of the migrating VMs on the destination, our approach performs regular memory transfers while simultaneously constructing the DBMS's buffer-pool by fetching the data items from the shared storage. The experimental results show that the migration time of our prototype is shorter under workloads, including sysbench and TPC-C, than the default pre-copy and post-copy schemes, respectively.

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　DBMS　インメモリDB

様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９（共通）

１．研究開始当初の背景

ビッグデータやそれらの機械学習をベースとしたAIサービスが一般的となった今日において，その源泉となるデータを管理するデータベース管理システム(DBMS)は IT インフラを支える必須のソフトウェアコンポーネントとなっている．特にインメモリ DBMS(インメモリ DB)と呼ばれる DBMS が大きな役割を担っており，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ことで，従来のディスクアクセスを主体とする DBMS よりも高いスループットと低いレイテンシの達成を実現している．たとえば，memcached や Redis とよばれるインメモリ DB は，全データアイテムをメモリ上に保持して管理することでディスクアクセスを避けて高い性能を達成している．また，MyRocks や伝統的な関係データベースである MySQL においても，Log-structured merged-tree といったディスクアクセスを最小限に抑えるデータ構造を採用している．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，ビッグデータ処理が閉じた環境だけではなく，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．

現行のクラウド環境のシステムソフトウェアスタックは大規模メモリ/インメモリ処理が主流となる以前に設計されており，インメモリ DB をクラウド環境上で効率的かつ安定的な稼働させることは難しい．大規模なメモリを搭載する仮想マシン(VM)を提供するクラウド環境は存在するものの，システムソフトウェアレベルで提供される仮想マシン(VM)移送や複製作成などの保守技術はメモリ使用量に応じてその処理時間が長くなる．そのため，インメモリ DB に既存技術を適用するとインメモリ DB の性能劣化が著しく，サービスの継続が困難もしくは不可能となる．そこで，クラウド環境上でインメモリ DB を安定的に稼働させるにはシステムソフトウェアの姿はどうあるべきか，という学術的問いに答える必要がある．

２．研究の目的

本応募課題では，インメモリ DB が稼動している 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を伴うため，従来想定してきたの数 GB クラスの VM 移送よりも遥かに時間を長く要する．そのため，上記のメリットを得ることが難しい．応募課題ではインメモリ DB 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ながらクラウド環境上での AI サービスの実現を狙う．

実現する移送方式では，これまで独立性の高かったインメモリ DB 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，インメモリ DB のメモリは多くの領域がストアしたデータアイテムに支配されており，また，インメモリ DB のアクセスに局所性があることに着目する．データアイテムの中でも頻繁にアクセスされるもののみを抽出してそれのみを移送元から転送し，転送されないデータアイテムはストレージなどから再び構築する．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という高レベルな情報は把握できない．提案する移送方式では，このギャップを埋めるべく，インメモリ DB 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

初年度は，提案方式の詳細設計およびプロトタイプの実装を行った．Linux 環境では，QEMU と呼ばれるソフトウェアモジュールが VM および VM 移送を実行している．これらに対して 1)データアイテム以外のメモリ領域を転送する機構, 2) 移送開始および完了を VM に通知する機構を組み込む. 1)は移送元，2)は移送元と先とで動作する．ここで，QEMU は memcached の資源管理情報を共有していないため，データアイテムがどのメモリページに配備されているかなどを把握できない．そこで，memcached と Linux とを連動させ，3)データアイテムのメモリページの物理アドレスを QEMU に通知する機構，4)頻繁にアクセスされるデータアイテムを抽出する機構，を用意し，これらを連動させる．移送先で引き継ぐデータアイテム量と移送時間，memcached の性能にはトレードオフがあり，移送先でのデータアイテム量を調整

可能にしておく．なお，本応募課題で扱うソフトウェアスタックに関しては応募者がこれまでに利用/拡張経験があり，十分な開発ノウハウが蓄積されており，円滑かつ詳細に設計することができる．

次年度以降には，前年度に作成したプロトタイプを用いて，人工的なワークロードを用意してプロトタイプの詳細な挙動を解析した．それが終了したら，実践的なワークロードを稼働させたときの有効性を定量的に評価した．提案方式は VM レベルの機構を伴う方式なので，Amazon EC2 といった環境を利用することは困難なため，実験にはクラウド環境を模した環境を構築して実験する．読み込みだけのワークロードから始まり，一部のテーブルだけを更新，ランダムに更新など，こちらが挙動を完全に把握できる単純なワークロードを稼動させながらプロトタイプを用いて移送する．トランザクションを含むワークロードに対しても評価実験を行えた．現在のソフトウェアは様々なコンポーネントから形成されているため，これらとプロトタイプが競合することなく動作するかを確認する．こうしたテストは，実践的なワークロードを与えた際の挙動分析が円滑に行うために実施し，その後，YCSB や Twitter のリクエストといった複雑なワークロードを稼働させ，提案方式の実践面の評価を行った．

４．研究成果

本研究の成果は次の３つであり，応募改題における学術的な特色，独創的な点となる． １つ目は提案方式の実現性を明らかにした点である．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．具体的には VM モニタ，オペレーティングシステム(OS) として，Linux および QEMU を，インメモリ DB として MySQL を拡張して提案方式を実現した．ソフトウェアの新技術としては，1.DBMS-memory Enlightenment，2.DBMS-storage Enlightenment，そして 3.アダプティブ DB ブロック転送である．DBMS-memory Enlightenment は VM モニタが DBMS の使用するバッファプールのメモリ ページ特定を可能にする．バッファプールをどのように使用しているかは DBMS プロセスが管理しているためプロセス自身がバッファプールに関する情報を収集し VM モニタへ提供することで実現する．ここで問題となるのが DBMS プロセス が管理するメモリページのアドレス空間とハイパーバイザが管理するメモリページのアドレス空間が 異なることである．ハイパーバイザが管理しているメモリページは mPFN レベルである．ゲスト VM 内の OS が管理するメモリページはゲストの物理ページフレーム（gPFN）レベルであり，ユーザプロセスである DBMS は仮想メモリアドレス（VFN）レベルとなっている．DBMS プロセスが VFN を VM モニタへ提供しても対応関係を知ることができないため，まず gPFN レベルまで変換する必要がある．VFN と gPFN の対応関係はゲスト VM の OS が提供しているため，ゲスト OS に VFN から gPFN へ変換するためのシステムコールを追加する．DBMS-memory Enlightenment と同様に DB ブロックとバッファプールの関 係についての知識や DB ブロックのメタデータを含む情報をハイパーバイザが認識できるように DBMS-storage Enlightenment を導入する．DBMS-storage Enlightenment はハイパーバイザがバッファプールのメモリページに保存されているデータとストレージ上のデータの対応関係を得るための機構である．移送を開始する際に DBMS-memory Enlightenment でメモリアドレスを取得すると同時にそのメモリアドレスに保存されている DB ブロックの ID も含めてハイパーバイザへ通知する．次に移送元で通常のライブ移送処理によってメモリページが転送されないように追加の処理を行う．移送先では DB ブロックの ID を元に共有ストレージから適切な DB ブロックをフェッチし，メモリアドレスを元に VM のメモリ領域にコピーすることでバッファプールの復元を行う．アダプティブ DB ブロック転送はバッファプール構築を共有ストレージからの復元と移送元からのメモリページ転送の両方を活用する機構である．目標は DBMS を含む VM のライブ移送時間を短縮することであるためバッファプールの構築に共有ストレージからの復元だけに頼らず，移送元からのメモリページ転送を活用するほうが得策である． 提案手法でのバッファプールの構築は共有ストレージからの復元を行うことで移送時間の短縮を図るが, VM のメモリのほとんどがバッファプールであった場合には移送元からの転送が先に完了しその転送能力が余剰となる．この余剰となった転送能力を活用するためにバッファプール構築をどの割合で分担するかが重要である．移送元からの転送能力や共有ストレージからの復元能力を事前に把握することは困難である．移送元から他のメモリページの転送が先に完了した場合には，まだ未構築のバッファプールのメモリアドレスリストの一部を移送元へ通知することでその領域を移送元からの転送に切り替えることで実現する．この移送元からのバッファプール転送はバッファプール構築が完了するまで繰り返される．

２つ目は Real-world なソフトウェアスタックに対する有効性である．実際のクラウド環境で利用されているソフトウェアを用いる．上述した Linux や QEMU，memcached は Google Compute Engine といった実際のクラウド環境で採用実績のあるソフトウェアである．これらの上で提案方式のプロトタイピングを実現した．プロトタイピングする上では，ビットマップの扱いがポイントとなる．既存の移送元の転送済みかどうかを判別するビットマップだけでは対応できないことが問題であるため，バッファプールの構築対象のメモリページかどうかを判別する構築対象ビットマップを追加する．移送開始時に DBMS から取得したメモリアドレスについて転送済ビットマップと追加で構築対象ビットマップに対象のメモリページであるかもマークする．通常は転送済みと判断されるため移送元 から転送されることはないが，ページフォールトが発生し要求された場合には構築対象ビットマップも一緒に確認することで移送元から転送されていないかどうかを判断することが可能となる．よって，Post-copy 方式においてもバッファプールを移

送先で共有ストレージから構築することが可能となり，ライブ移送の移送時間短縮を実現する．

3 つ目は実践的なークロードに対する有効性である．クラウド環境を模した環境を構築して，プロトタイプを様々なワークロードを用いて定量的に評価した．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，YCSB 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した．実験結果から，Sysbench 及び TPC-C を含むワークロードを実行している状況下では VM のライブ移送の移送時間がデフォルトよりも Pre-copy 及び Post-copy 方式において最大で 1.71 倍短縮されることが明らかとなった．作成したプロトタイプはライブ移送が用いられる代表的 な 3 つのシチュエーション（VM 退避，バッチ移送，輻輳中の移送）でも有効であることが確認された．

５．主な発表論文等

〔雑誌論文〕　計0件

〔学会発表〕　計1件（うち招待講演　0件／うち国際学会　0件）

１．発表者名 池田 慧,山田 浩史
２．発表標題 In-memory KVS と連動する仮想マシンライブ移送
３．学会等名 IPSJ 第158回システムソフトウェアとオペレーティング・システム研究会
４．発表年 2023年

〔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

７．科研費を使用して開催した国際研究集会

〔国際研究集会〕　計0件

８．本研究に関連して実施した国際共同研究の実施状況

共同研究相手国	相手方研究機関