大容量メモリを備えた仮想計算機の移送の実現および高速化

代表研究者　　光　来　健　一　　九州工業大学大学院情報工学研究院・准教授

1 はじめに

クラウドコンピューティングのサービス形態の一つである IaaS 型クラウドでは、仮想マシン（VM）をサービスとしてユーザに提供する。IaaS 型クラウドの普及にともない、1 台のサーバに多くの VM を統合するだけでなく、大容量メモリを持つ VM も提供されるようになってきた。例えば、Amazon EC2 の X1 インスタンスでは 2TB のメモリが提供されている。このような大容量メモリを持つ VM は Apache Spark[1]や Facebook Presto[2]などを用いたビッグデータ解析に用いられる。できるだけメモリ上にデータを保持することで、ビックデータをより高速に解析することができる。また、メモリ上に大量のデータを保持する SAP HANA[3]や Microsoft SQL Server[4]などの高速なインメモリ・データベースを用いることもできる。

VM はホストをメンテナンスする際などにマイグレーションにより他のホストに移動されることがあるが、大容量メモリを持つ VM のマイグレーションには二つの問題がある。一つ目の問題はマイグレーション時間である。マイグレーション時間は基本的に VM のメモリサイズに比例して長くなる。この問題は、VM のメモリを並列に送る手法[5]や、高速なネットワークを用いる[6]ことで解決することができる。もう一つの問題は、VM のマイグレーションには移送先ホストに十分な空きメモリが必要になるということである。マイグレーションのために大容量メモリを備えた空きホストを常に確保しておくのは、可能だとしてもコストの大幅な増大を招く。VM のマイグレーションが行えなければ、ホストのメンテナンス中は VM を停止させなければならなくなり、VM のメモリ上の大量のデータを失うことになる。

一方で、クラウド上には小さな空きメモリを持ったホストが多く存在している。これらの空きメモリの合計は大容量メモリを持つ VM のメモリサイズを上回る場合が多い。このような断片的なメモリを一つに統合するためにリモートページング[7][8][9]を用いた仮想メモリが提案されてきた。従来の仮想メモリは物理メモリに入りきらないメモリをディスクにページアウトすることで仮想的に大容量メモリを利用することを可能にする。リモートページングはローカルディスクの代わりにネットワーク上のホストの空きメモリを利用する。しかし、仮想メモリはマイグレーションとの相性が悪く、マイグレーション中やマイグレーション後に大量のページングを発生させる場合がある。その結果、マイグレーション性能や VM の実行性能が大幅に低下し、最悪の場合、スラッシングのせいで VM のマイグレーションが終了しないことにもなる。

そこで、 本研究では、上記の問題を解決するために大容量メモリを持つ VM を複数のホストに分割してマイグレーションすることを可能とするシステム S-memV を提案する。S-memV では、マイグレーション時の VM の移送先のホストは 1 台とは限らず、1 台のメインホストと 1 台以上のサブホストからなる。S-memV は CPU やデバイスの状態のような VM の核となる情報を移送先のメインホストに送る。また、VM がマイグレーション後にアクセスすると予測されるメモリもできる限りメインホストに送る。一方、メインホストのローカルメモリに入りきらないメモリはサブホストのいずれかに送る。S-memV では移送先のメインホストまたはサブホストに直接メモリを送るため、マイグレーション中にはページングが発生しない。マイグレーション後はメインホストで VM を動作させ、VM がサブホストにあるメモリを必要とした場合には、メインホストとサブホストの間でリモートページングを行う。S-memV では、VM のメモリを分割する際に参照の時間的局所性を考慮するため、マイグレーション後の VM の性能低下を抑えることができる。

我々は S-memV を KVM に実装し、VM の分割マイグレーションと統合マイグレーションを実現した。サブホスト上ではメモリサーバが動作し、メインホストで動作する VM のメモリの一部を管理する。拡張ページテーブルを用いて VM のメモリ参照情報を取得する機構を開発し、参照の時間的局所性を利用する LRU 近似アルゴリズムを実装した。ページングには Linux 4.3 で導入された userfaultfd 機構を用い、ページアウトのための拡張を行った。実験結果より、S-memV は仮想メモリを用いた従来手法よりもマイグレーション時間とダウンタイムを大幅に短縮できることが分かった。また、分割マイグレーション後の VM の性能低下を抑えられることも分かった。

2 大容量メモリを持つ VM のマイグレーション

VM マイグレーションは稼働している VM を停止させることなく別のホストへ移動させる技術である。マイグレーションを活用することで、VM が提供しているサービスを停止させることなくホストのメンテナンスを行うことができる。マイグレーションを行う際には、移送先ホストに新しい VM を作成し、移送元ホストで動いている VM のメモリの内容をネットワーク経由で移送先ホストの VM のメモリにコピーする。この間、移送元ホスト上で VM のメモリの内容は更新され続けているため、再度、更新されたメモリを転送する。これを繰り返して転送するメモリ量が十分小さくなったら移送元ホストの VM を停止し、VM の CPU やデバイスの状態および更新されたメモリを転送してマイグレーションを完了する。

近年、大容量メモリを持つ VM が使用されるようになってきた。例えば、Amazon EC2 では 244GB のメモリを持つ 8xlarge インスタンスが提供されている。最近では、新しく 2TB のメモリを持つ x1.32xlarge インスタンスも提供されるようになった。このような大容量メモリを持つ VM はビッグデータの解析[1][2]やインメモリ・データベース[3][4]などのように、大量のデータを高速に扱う場合に用いられている。しかし、このような大容量メモリを持つ VM は、マイグレーション時に移送先を見つけることが困難になるという問題がある。移送先として大きな空きメモリ容量を持ったホストを確保し続けておくことは、コスト面からも難しいことが多い。十分なメモリ容量を持つホストが多数の小さな VM を動かすために使われていた場合、まず、それらの VM をマイグレーションして必要な空きメモリ容量を確保する必要がある。

従来、移送先ホストに十分な空きメモリ容量がない場合には仮想メモリが用いられてきた。仮想メモリは物理メモリに入りきらないメモリをディスク上のスワップ領域に待避することで、物理メモリよりも大きなメモリを扱うことができる技術である。しかし、仮想メモリは大容量メモリを持つ VM のマイグレーションとは相性が悪い。マイグレーションの初期段階では VM のメモリが先頭から順番に転送されるため、移送先ホストの物理メモリに入りきらなくなった場合、VM のメモリはアクセスパターンに関わらず LRU に基づいて無条件にアドレスの小さいものから順にページアウトされてしまう。その結果、マイグレーション後に必要なメモリがページアウトされていると VM の性能低下を引き起こす。

メモリの再送時にもページングは頻繁に発生する。移送元ホストで書き換えられたために再送されたメモリが移送先ホストの物理メモリ上にない場合、そのメモリはディスクからページインされた後で上書きされる。大容量メモリを持つ VM のマイグレーションでは最初のメモリ全体の転送に時間がかかるため、その間に書き換えられるメモリ量も増大する。マイグレーション完了時には、頻繁に書き換えられるページは物理メモリ上にあるが、読み込みしか行われないページは再送されないためページアウトされている可能性が高くなる。そのため、マイグレーション後にはディスクとの間で頻繁にページングが起こる。メモリに比べてディスクは非常に遅いため、ページングを繰り返すと性能が著しく低下する。HDD の代わりに SSD を用いることで性能低下を抑えられるが、SSD でも依然としてメモリよりも 1〜2 桁低速である。

ディスクを用いたページングのオーバヘッドを削減するためにリモートページング[7][8][9]が提案されてきた。リモートページングはディスクの代わりにネットワーク上の別のホスト上のメモリとの間でページングを行う。ネットワークが十分高速であれば、リモートページングはディスクを用いたページングよりも高速である。しかし、リモートページングもマイグレーションとは相性が悪い。マイグレーションにより VM のメモリ全体が移送先ホストに転送されている間に、移送先ホストは図 1 のように、ページアウトされたメモリをスワップ用ホストに転送しなければならない。そのため、移送先ホストとスワップ用ホスト間のネットワーク帯域を消費してしまう。加えて、リモートページングによって移送先ホストのシステム負荷が増加し、マイグレーション性能にも影響を及ぼす。

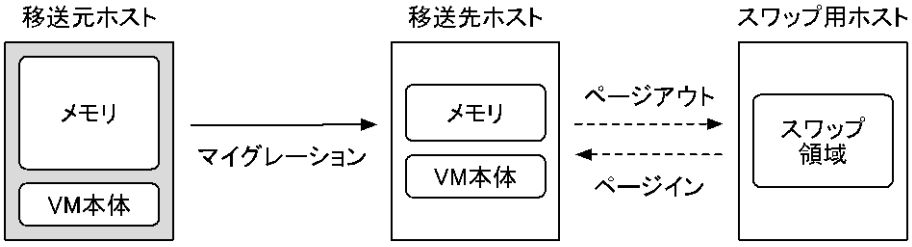

図 1 リモートページングを利用したマイグレーション

3 S-memV

本研究では、大容量メモリを持つ VM を分割して複数のホストにマイグレーションすることを可能とする S-memV を提案する。

3-1 分割マイグレーション

S-memV では、1 つのホストから複数のホストへのマイグレーションが可能である。移送先の複数のホストは図 2 のように 1 台のメインホストと 1 台以上のサブホストからなる。VM を動かす上で必要となる CPU やデバイスの状態などの核となる情報はメインホストに送る。また、VM がアクセスすると予測されるメモリはできる限りメインホストに送り、移送先ホストの VM がオーバヘッドなしでアクセスできるようにする。一方、メインホストに入りきらない VM のメモリはサブホストのいずれかに転送する。この際に、空きメモリを考慮して最適なメインホストとサブホスト群を選択できるようにするために、全ホストの空きメモリを管理するサーバを用意する。移送元ホストはこのサーバにアクセスすることにより、マイグレーション先のホスト群を決定する。

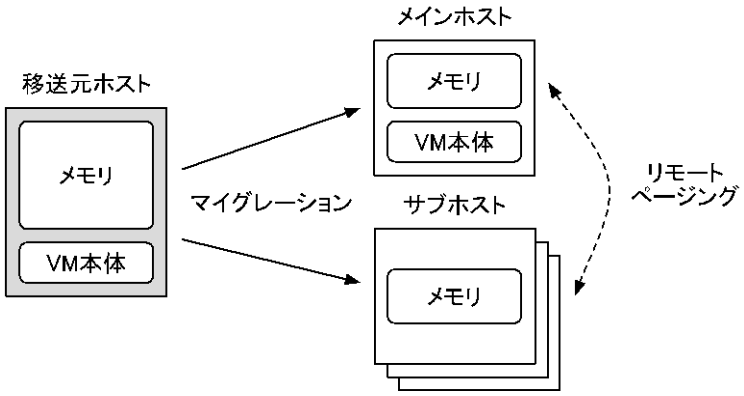

図 2 分割マイグレーション

マイグレーション後はメインホスト上で VM を動作させる。アクセスされたメモリがメインホスト上にない場合は、サブホスト上にある要求されたメモリをメインホストに転送し、ページイン処理を行う。同時に、メインホスト上の今後アクセスされないことが予測されるメモリのページアウト処理を行い、ページインを行ったサブホストに転送する。この挙動はリモートページングと同様である。しかし、S-memV はマイグレーション中にページングを発生させない点が異なる。メインホストに入りきらないメモリはメインホスト経由でサブホストにページアウトされるのではなく、直接、サブホストに転送される。そのため、マイグレーションの最初のメモリ転送時も再送時も、メインホストとサブホスト間で無駄なネットワーク転送が行われることはない。また、VM がアクセスすることが予測されるメモリはメインホストに転送されているため、マイグレーション後のページング頻度は低くなることが期待できる。

3-2 複数ホストにまたがる VM のマイグレーション

複数のホストに分割マイグレーションを行った VM は、ホストのメンテナンス終了後や十分な空きメモリ容量を持ったホストが用意できた時などに、図 3 のように再び 1 台のホストで動作するようにマイグレーションすることができる。その際に、移送元のメインホストとサブホスト上の VM のメモリを並列に転送することにより、高速にマイグレーションを行うことができる。マイグレーション中にページインまたはページアウトされたメモリについては、転送漏れや重複がないように転送する。

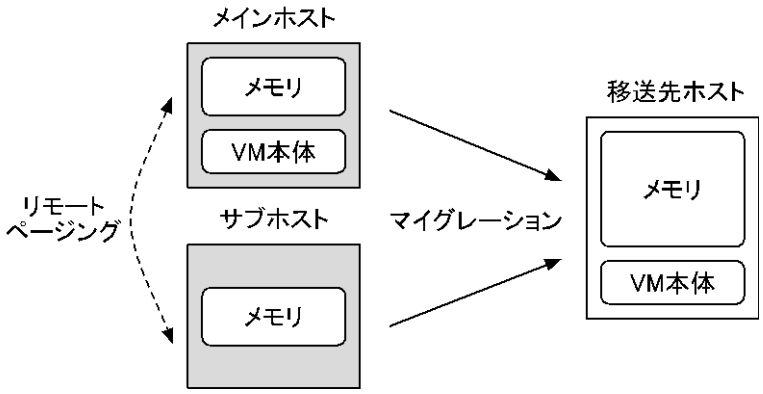

図 3 統合マイグレーション

VM が動作している複数のホスト全体または一部をメンテナンスできるようにするために、複数のホストにまたがる VM の全体または一部を別のホスト群にマイグレーションすることもできる。メインホスト上にある VM の本体をマイグレーションする際には、移送先のメインホストに VM の核となる情報や VM がアクセスすると予測されるメモリを転送し、入りきらないメモリは既存のサブホストまたは新しく確保したサブホストに転送する。サブホスト上にある VM の一部をマイグレーションする際には、移送先のサブホストに VM のメモリの一部を転送する。マイグレーション中にページングが発生した場合には、必要に応じて追加でメモリを転送したり、転送済みのメモリを無効化したりする。

4 実装

S-memV を QEMU-KVM 2.4.1 および Linux 4.3 に実装した。現在のところ、プレコピー・マイグレーションを用いた分割マイグレーションをサポートしている。S-memV のシステム構成を図 4 に示す。移送元ホストと移送先メインホストでは S-memV を実装した QEMU-KVM が動作し、マイグレーション前後で VM を動かす。移送先サブホストでは VM のメモリの一部を管理するメモリサーバを動作させる。

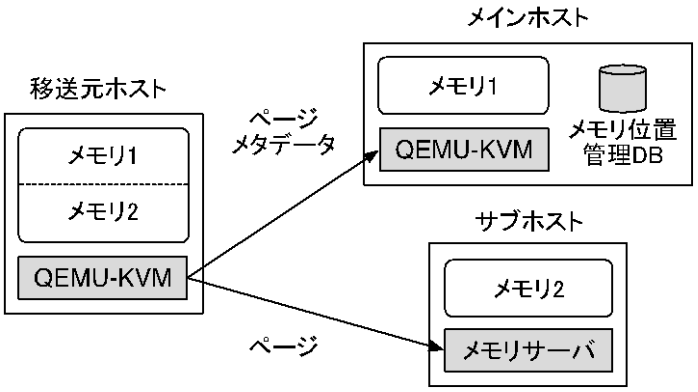

図 4 S-memV のシステム構成

4-1 分割マイグレーション

VM のメモリを分割してマイグレーションできるようにするために、QEMU-KVM のマイグレーション機構の拡張を行った。S-memV ではマイグレーションを開始した時に、移送先のメインホストで起動した QEMU-KVM だけでなく、サブホストで動作しているメモリサーバにもネットワーク接続を行う。次に、移送先メインホストとサブホストの空きメモリに応じて VM のメモリを分割する。メモリの分割は LRU に基づいて行い、マイグレーション後にアクセスされる可能性が高いメモリページから順にメインホストに、残りをサブホストに割

り当てる。現在の実装ではマイグレーション開始時までのメモリ参照履歴に基づいて VM のメモリを分割するため、マイグレーション中のメモリアクセスは考慮していない。LRU に基づくメモリ分割については 4-4 節で詳しく述べる。

VM のメモリが分割できたら、それに応じてメモリページをメインホストまたはサブホストに転送する。メインホストには従来と同様にメモリブロックのオフセットやメモリページの内容を送るのに加え、移送元でのメモリ参照履歴の情報を送る。この付加情報により、移送先でも VM のメモリ参照履歴を引き継ぐことができる。一方、サブホストには VM の物理メモリアドレスとメモリページの内容を送信する。メモリブロックのオフセットではなく物理メモリアドレスを送るのは、サブホストではメモリブロックを管理していないためである。サブホストのメモリサーバでの処理については 4-2 節で述べる。サブホストに送ったページについては、メインホストにもサブホストの IP アドレスとメモリアドレスの情報を送る。メインホストでは各メモリページがどのホストにあるかをメモリ位置管理データベースによって管理し、マイグレーション後のリモートページングに利用する。

マイグレーションの終了時には、メインホストからサブホスト上のメモリサーバにネットワーク接続を行い、サブホストとの間でリモートページングが行えるようにする。VM がメインホスト上にないページにアクセスした時にリモートページングを行えるようにするために、Linux 4.3 で導入された userfaultfd 機構を用いる。リモートページングの実装については 4-5 節で詳しく述べる。

4-2 メモリサーバ

メモリサーバは VM のメモリの一部を管理するサーバであり、サブホスト上で動作する。メモリサーバはページ番号をインデックスとする配列（ページ配列）を用いて VM のメモリを 4KB のページ単位で管理する。メモリサーバが VM の物理メモリアドレスとメモリページの内容からなるページアウト要求を受信した時には、4KB のメモリを確保して送られてきたメモリページの内容をコピーし、メモリアドレスに基づいてページ配列に登録する。一方、VM の物理メモリアドレスからなるページイン要求を受信した時には、メモリアドレスを基にページ配列を探索し、VM のメモリページが見つかればそのデータを要求元に送信する。同時にそのデータをページ配列から削除する。

4-3 メモリ参照履歴の管理

S-memV では VM のメモリ参照履歴を管理するために、図 5 のように VM の拡張ページテーブル（EPT）をたどって各ページのメモリ参照に関する情報を取得する。そのために、QEMU-KVM は定期的に Linux カーネル内の KVM に対して ioctl システムコールを発行する。現在の実装では、1 秒おきに KVM を呼び出している。その際に VM のメモリサイズに応じたビットマップを確保し、それを ioctl の引数として KVM に渡す。KVM は VM のすべてのメモリページについて EPT をたどり、ページテーブルエントリ（PTE）を取得する。PTE のアクセスビットは対応するページにアクセスした時に 1 にセットされるため、アクセスビットの値を渡されたビットマップに記録する。その後、次の期間のアクセスを記録できるようにするために、PTE のアクセスビットをクリアする。

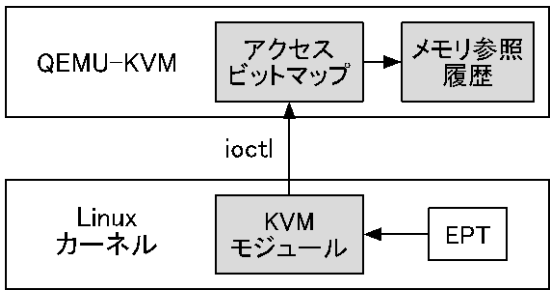

図 5 メモリ参照履歴の取得

S-memV は LRU 近似アルゴリズムとして、クロックアルゴリズムとエージングアルゴリズムをサポートしている。クロックアルゴリズムでは各ページについて 1 ビットのメモリ参照履歴が必要になるため、定期的に EPT をたどって取得したアクセスビットの値をメモリ参照ビットマップに蓄積していく。具体的には、図 6(a)のように取得したアクセスビットが 1 の場合にはメモリ参照ビットマップのビットを 1 にし、0 の場合には

メモリ参照ビットマップの更新を行わない。このように定期的にメモリ参照ビットマップを更新するのは、リモートページングの際にメモリ参照情報を取得するオーバヘッドを減らしつつ、できるだけ最新のメモリ参照情報を利用できるようにするためである。メモリ参照ビットマップはクロックアルゴリズムを実行することによっても更新される。クロックアルゴリズムの実装については 4-5 節で述べる。

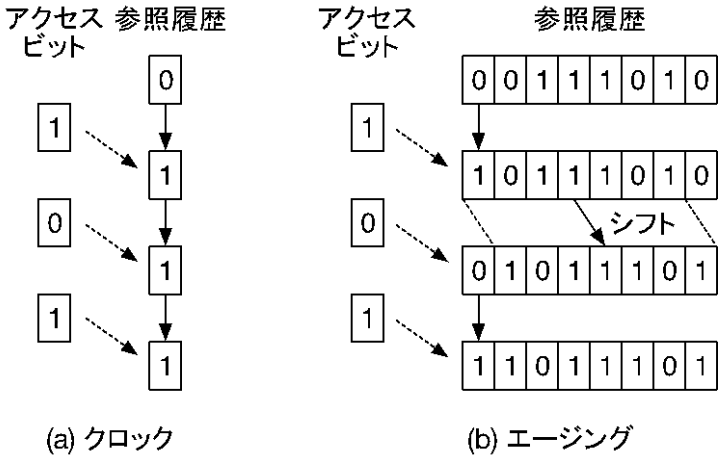

図 6 メモリ参照履歴の更新

一方､エージングアルゴリズムでは各ページに対してある程度の過去までのメモリ参照履歴が必要になる。現在の実装では図 6(b)のように各ページに 8 ビットを割り当てて参照履歴を管理している。定期的に、EPT をたどって取得したアクセスビットの値はしばらくの間、この 8 ビットの最上位ビットに蓄積していく。これは 1 秒間のメモリ参照を 1 ビットに対応させると 8 秒間の履歴しか利用できないためである。この参照履歴は定期的に 1 ビット右にシフトし、再び最上位ビットにアクセスビットを蓄積していく。このようにすることで、より最近にアクセスされたページほどメモリ参照履歴の値が大きくなる。エージングアルゴリズムの詳細についても 4-5 節で述べる。

4-4 LRU に基づくメモリ分割

S-memV では、マイグレーションを行う際にメモリ参照履歴に基づいて VM のメモリを分割する。このメモリ分割は連続するメモリページをチャンクとよばれるかたまりに分け、チャンク単位で行う。これはリモートページングがチャンク単位で行われるためである。チャンクのサイズは 1 以上の 2 のべき乗とする。メモリ分割の方法は用いる LRU 近似アルゴリズムによって異なる。

クロックアルゴリズムを用いる場合、チャンクごとにメモリ参照ビットマップの対応するビットの値の合計を計算し、その値が大きいチャンクに含まれるメモリページから順にメインホストに割り当てる。ビットの合計値を用いるのは、アクセスされたページ数が多いチャンクを優先するためである。チャンクサイズが 1 の場合には、メモリ参照ビットマップのビットが 1 のページをメインホストに割り当てる。ビットが 1 のすべてのページを割り当ててもメインホストに空きがあれば、ビットが 0 のページもメインホストに割り当てる。メインホストの空きがなくなったら残りのページはサブホストに割り当てる。

エージングアルゴリズムを用いる場合、それぞれのチャンクのページの中でメモリ参照履歴の 8 ビット値が最大のものを見つけ､その値がより大きいチャンクに含まれるページから順にメインホストに割り当てる。メモリ参照履歴の最大値を用いるのは、チャンクの中で一番最近アクセスされたページを重視するためである。

4-5 リモートページング

マイグレーション後のリモートページングは Linux の userfaultfd 機構を用いて実装した。リモートページングのシステム構成を図 7 に示す。S-memV はまず、userfaultfd 機構に VM のメモリを登録する。VM のメモリは匿名メモリマッピングを用いて確保されており、マイグレーション時に受信したデータを書き込んだページだけに実メモリが割り当てられる。それ以外のページについては実メモリは割り当てられていない。Transparent HugePages が有効になっていると 2MB 単位で実メモリが割り当てられてしまい、チャンク単位でのページングが行えないため、この機能は無効にした。VM がまだ実メモリが割り当てられていないページ

にアクセスすると、ページフォールトが発生し、userfaultfd 機構によって QEMU-KVM にイベントが通知される。

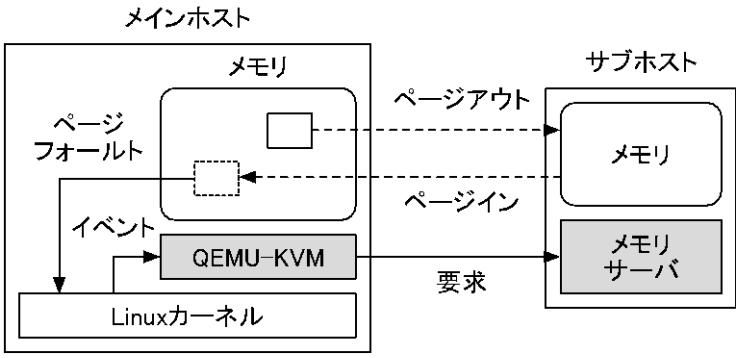

図 7 リモートページングのシステム構成

イベントを受け取った QEMU-KVM はページフォールトが発生した QEMU-KVM 上のメモリアドレスを VM の物理メモリアドレスに変換し、サブホストのメモリサーバにページイン要求を送る。その際に、まずそのメモリアドレスを含むページについてページイン要求を送り、続けてそのページを含むチャンクの残りのすべてのページについてページイン要求を送る。このような順番で要求を送ることで、ページフォールトが発生したページを最初に処理して VM の実行を再開できるようにしている。メモリサーバがその要求を受け取ると、渡されたメモリアドレスに対応するページのデータを QEMU-KVM に返送する。ページのデータを受け取った QEMU-KVM は userfaultfd 機構を用いてデータを該当メモリに書き込むことでページイン処理が完了する。同時に、ページの所在を管理するメモリ位置データベースを更新して、ページインしたメモリをメインホストで管理する。

ページインにともなってメモリ量のバランスを取るために、メインホストからサブホストにメモリをページアウトする。まず、S-memV は 4-3 節で述べた VM のメモリ参照履歴に基づいて今後アクセスされないことが予測されるメモリチャンクを選択する。次に、ページアウトするページの内容を取得し、そのページへの実メモリの割り当てを QEMU-KVM から削除する。この操作をアトミックに行うために userfaultfd 機構を拡張し、対応するページテーブルエントリをクリアしてそのページの内容を返せるようにした。次に、選択されたチャンクに含まれるページに対してメモリサーバにページアウト要求を送る。それに加えて、メモリ位置データベースを更新してページアウトしたページをメインホストで管理しないようにする。

ページアウトするチャンクは LRU に基づいて決定する。クロックアルゴリズムを用いる場合には、チャンクごとにメモリ参照ビットマップの対応するビットの合計値を求めて、その合計値が最小のチャンクを選ぶ。これにより、アクセスされていないページが多いチャンクを優先して選ぶことができる。エージングアルゴリズムを用いる場合には、チャンクに含まれるページのメモリ参照履歴の中から最大値を見つけ、その値が最小のチャンクを選ぶ。このようにすることで、チャンクの中で一番最近アクセスされたページを重視することができる。

5 実験

S-memV の有効性を示すために、マイグレーション性能やマイグレーション後の VM の性能などを調べる実験を行った。比較のために、移送先ホストに十分なメモリがある場合と、仮想メモリとして SSD または HDD を用いた場合についても調べた。実験には、Intel Xeon E3-1270v3 3.5GHz の CPU、16GB のメモリを搭載したマシン 2 台をそれぞれ移送元ホストおよび移送先メインホストとした。S-memV を用いる場合は移送先メインホストの空きメモリを 6GB と仮定し、仮想メモリを用いる場合には移送先ホストの物理メモリの空きが 6GB になるように調整した。移送先サブホストには、Intel Xeon E3-1270v2 3.5GHz の CPU、12GB のメモリを搭載したマシンを用いた。仮想メモリを用いる場合には、移送先ホストで 1TB の SATA III HDD または Crucial MX300 SSD にスワップ領域を設定した。これらのマシンは 10 ギガビットイーサネット（10GbE）で接続した。ホスト OS には Linux 4.3.0 を用い、仮想化ソフトウェアには QEMU-KVM 2.4.1 を使用した。VM には仮想 CPU

を 1 つ、メモリを 12GB 割り当てた。

5-1 マイグレーション性能

S-memV のマイグレーション性能を調べるために、まず、VM 内でアプリケーションを動作させない場合のマイグレーション時間とダウンタイムを測定した。マイグレーション時間は図 8(a)のようになり、メモリが十分にある場合と比べて、S-memV ではマイグレーション時間が 5%増加した。それに対して、仮想メモリを用いた場合には SSD をスワップ領域にした場合でも 2.2 倍の時間がかかった。HDD をスワップ領域にすると 11.7 倍の時間がかかった。ダウンタイムは図 8(b)のようになり、S-memV では十分なメモリがある場合より 7 ミリ秒だけ長くなった。しかし、仮想メモリを用いた場合には SSD で 3.4 秒、HDD では 30 秒も長くなった。これは激しいページングによるものである。

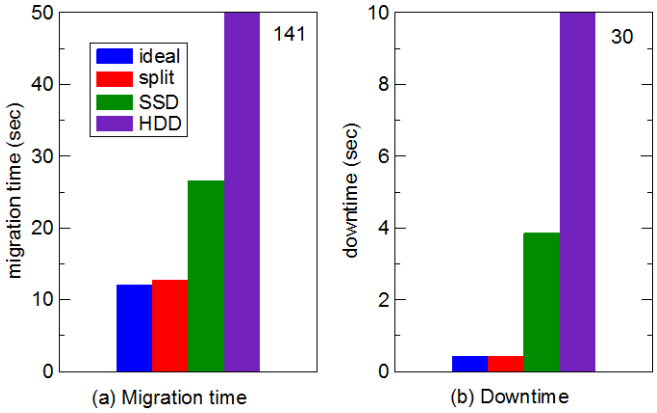

図 8 マイグレーション性能

次に、VM 内で 6GB のメモリを書き換え続けるプログラムを動作させ、大量のメモリが再送される状態でマイグレーションを行った。実験結果を図 9 に示す。メモリが十分にある場合でも、上の実験と比べてマイグレーション時間が 31 秒増加した。それと比べて、S-memV では 23 秒の増加にとどまった。仮想メモリを用いた場合には、SSD をスワップ領域にしても 55 秒、HDD では 188 秒もマイグレーション時間が増加した。一方、ダウンタイムは上の実験とほぼ同じであったが、HDD を用いた場合だけ 26 秒減少した。これは再送が繰り返されることによってよくアクセスされるデータが移送先ホストのメモリ上に残り、マイグレーションの最終段階でのページング頻度が減ったためと考えられる。

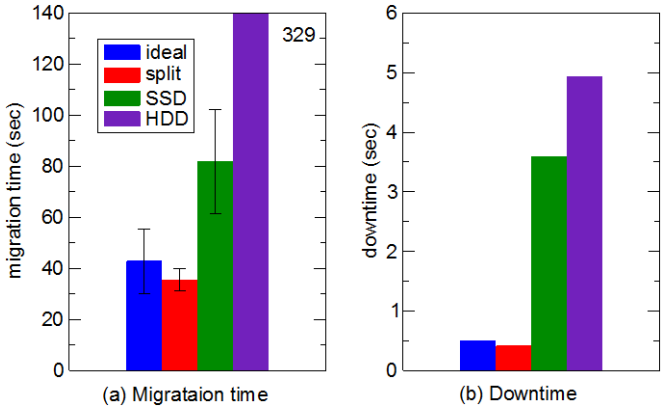

図 9 マイグレーション性能（高負荷時）

さらに、ネットワークがマイグレーション性能に与える影響について調べた。この実験ではギガビットイーサネット（GbE）と 10GbE とで比較を行い、GbE でのマイグレーション時間を考慮して VM に割り当てるメモリは 2GB とした。図 10(a)より、S-memV では 10GbE を用いることで 7 倍高速化できることが分かった。一

方、仮想メモリを用いた場合には SSD をスワップ領域にしても 2.9 倍しか高速化できなかった。また、HDD をスワップ領域にすると GbE を用いたほうが高速であるという結果となった。ダウンタイムについては図 10(b)に示すように、S-memV ではほぼ変化がなかったが、仮想メモリを用いる場合には 10GbE のほうが長くなった。これは 10GbE のほうが転送すべき残りページが多い状態でマイグレーションの最終段階に入るようなアルゴリズムになっているためと考えられる。

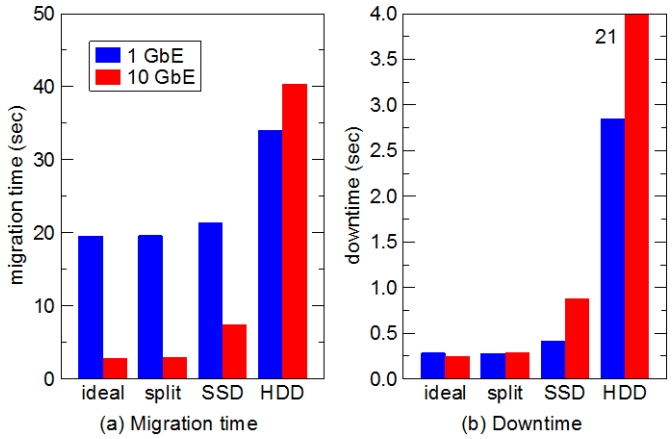

図 10 マイグレーション性能（GbE と 10GbE）

5-2 マイグレーション後の VM の性能

マイグレーション後の GNU sort の性能を測定した。メモリ参照履歴を取得するために、まず、VM 内で 2GB のデータに対して sort コマンドを 60 秒間実行して一時停止させた。次に、VM のマイグレーションを行って sort コマンドを再開し、終了までにかかる時間を測定した。図 11 にソート時間と実行中のページイン回数を示す。メモリが十分にある場合と比べて、S-memV での性能低下は 16%であった。これに対して、仮想メモリを使用した場合の性能低下は SSD をスワップ領域にすると 35%であった。HDD を用いるとソート時間は 7.6 倍にもなった。この差はページイン回数と強く関連していることが分かる。

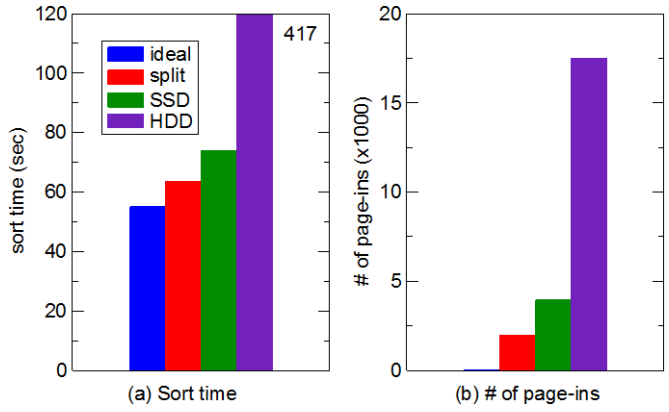

図 11 ソート性能

ここで、メモリ分割とリモートページングのどちらに LRU を用いた効果があったのかを調べるために、マイグレーション後に sort コマンドの実行を開始して性能を測定した。この場合、sort コマンドのメモリ参照を考慮せずにメモリ分割が行われるため、LRU に基づくリモートページングの効果だけを確認することができる。この実験では、S-memV における性能低下は仮想メモリを用いた場合と同程度であった。つまり、LRU に基づくリモートページングの性能は SSD を用いたページング性能と同等であるということである。これより、S-memV がマイグレーション時に行っている LRU に基づくメモリ分割のほうが効果があることが示された。

次に、マイグレーション後の memcached[10]の性能を測定した。memcached はインメモリ・データベースの一種である。memcached に 6GB のメモリを割り当て、memaslap ベンチマーク[11]を使ってそのデータにアクセスした。メモリ参照履歴を取得するためにマイグレーション前に memaslap を実行しておき、マイグレーション後に再び実行した。5 秒おきのスループットを図 12 に示す。メモリが十分にある場合にはスループットは終始安定していた。S-memV ではマイグレーション直後のスループットが低下したが、すぐに性能を回復できていることが分かる。しかし、メモリが十分にある場合と比べるとスループットは 14%低かった。一方、仮想メモリを用いた場合にはマイグレーション後のスループットは大幅に低下し、性能が回復するのに SSD で 45 秒、HDD で 85 秒かかった。

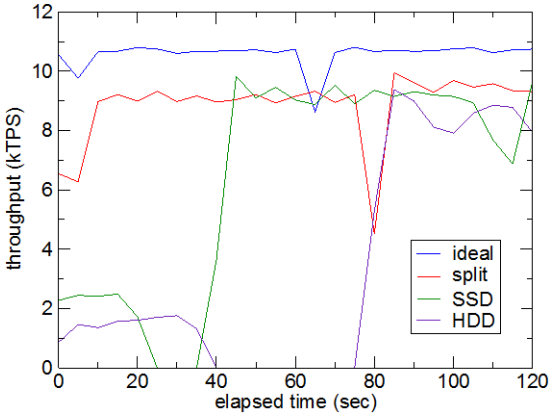

図 12 memcached の性能

6 まとめ

本研究では、大容量メモリを持つ VM を複数のホストに分割してマイグレーションすることを可能とするシステム S-memV を提案した。S-memV は VM の核となる情報と VM がアクセスすると予測されるメモリを移送先のメインホストに転送し、メインホストに入りきらないメモリをサブホストに転送する。マイグレーション後、VM はメインホストで動作し、必要に応じてサブホストとの間でリモートページングを行う。S-memV を KVM に実装し、分割マイグレーションを実現した。実験結果より、マイグレーション性能とマイグレーション後の VM の性能の低下を抑えることができることを確認した。

【参考文献】

[1] Apache Software Foundation. Apache Spark – Lightning-Fast Cluster Computing. http://spark.apache.org/.

[2] Facebook, Inc. Presto: Distributed SQL Query Engine for Big Data. https://prestodb.io/.

[3] SAP SE. SAP HANA https://hana.sap.com/.

[4] Microsoft Corporation. SQL Server 2014. https://www.microsoft.com/en/server-cloud/products/sql-server/.

[5] X. Song, J. Shi, R. Liu, J. Yang, and H. Chen. Parallelizing Live Migration of Virtual Machines. In Proceedings of the 9th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments. page 85-96 (2013).

[6] Mellanox Technologies. Accelerating Virtual Machine Migration over vSphere vMotion and Mellanox End-to-End 40GbE Interconnect Solutions. http://www.melloanox.com/related-docs/solutions/SB_Accelerating_Virtual_Machine_Migration.pdf (2016).

[7] D. Comer and J. Griffioen. A New Design for Distributed System: The Remote Memory Model. In Proceedings of the Summer 1990 USENIX Conference, pages 127-135 (1990).
[8] U. Deshpande, B. Wang, S. Haque, M. Hined, and K. Gopalan. MemX: Virtualization of Cluster-Wide Memory. In Proceedings of International Conference on Parallel Processing (2010).
[9] M. J. Feeley, W. E. Morgan, E. P. Pighin, A. R. Karlin, H. M. Levy, and C. A. Thekkath. Implementing Global Memory Management in a Workstation Cluster. In Proceedings of the 15th ACM Symposium on Operating Systems Principles, pages 201-212 (1995).
[10] B. Fitzpatrick. memcached – A Distributed Memory Object Caching System. http://memcached.org/.
[11] B. Aker. memaslap – Load Testing and Benchmarking a Server. http://docs.libmemcached.org/bin/memaslap.html.

〈発 表 資 料〉

題　　名	掲載誌・学会名等	発表年月
Split Migration of Large Memory Virtual Machines	7th ACM SIGOPS Asia-Pacific Workshop on Systems (APSys 2016)	2016 年 8 月 4 日
S-memV: 大容量メモリを持つ VM の分割マイグレーション	情報処理学会 第 139 回 OS 研究会	2017 年 3 月 2 日