VM専用仮想メモリとの連携による VMマイグレーションの高速化

村岡 裕二[1]　柏木 崇広[1]　光来 健一[1]

概要：近年，大容量メモリを持つ仮想マシン（VM）が提供されるようになってきている．VM はホストのメンテナンス等の際にマイグレーションされるが，大容量メモリを持つ VM の移送先として十分な空きメモリを持つホストを確保しておくのはコスト面での負担が大きい，そこで，移送先ホストの仮想メモリを活用してマイグレーションを行うことが考えられるが，従来の仮想メモリはマイグレーションとの相性が悪く，マイグレーション性能が大幅に低下する．本稿では，VM 専用の仮想メモリと連携することにより VM マイグレーションの高速化を実現するシステム VMemDirect を提案する．VMemDirect は移送先ホストの NVMe 上に VM 専用スワップ領域を作成し，アクセスすることが予測されるメモリデータを物理メモリに，それ以外のメモリデータを VM 専用スワップ領域に直接転送する．マイグレーション後は VM 専用の仮想メモリを用いて物理メモリと VM 専用スワップ領域の間でページングを行う．我々は VMemDirect を KVM に実装し，マイグレーション性能およびマイグレーション後の VM の性能を測定した．

1. はじめに

IaaS 型クラウドでは大容量メモリを持つ仮想マシン（VM）が提供されるようになっており，ビッグデータの解析等に利用されている．VM が稼働しているホストをメンテナンスする際には，VM を停止させずに別のホストにマイグレーションすることで VM の実行を継続することができる．マイグレーションでは，移送先ホストにネットワーク経由で VM のメモリなどの状態を転送し，移送先ホストで VM を再開する．マイグレーション中に VM のメモリの内容が更新されると当該メモリのデータを移送先ホストに再送する．このように，マイグレーションを行うには，移送先ホストに VM のメモリを格納することのできる空きメモリが必要となる．大容量メモリを持つ VM の場合，移送先として十分な空きメモリを持つホストを確保し続けることはコスト面で大きな負担となる．

そこで，移送先ホストで仮想メモリを用いることにより，マイグレーションを行うことが考えられる．仮想メモリを用いると，VM がディスク上に存在するメモリデータを必要とした時には，当該データを物理メモリ上に転送（ページイン）し，代わりに，物理メモリ上の不要なメモリデータをディスクに転送（ページアウト）することができる．しかし，従来の仮想メモリはマイグレーションとの相性が悪く，マイグレーションの性能が低下する．マイグレーション中に大量のページングが発生するためである．また，マイグレーション後に VM が必要とするメモリデータがページアウトされていることも多く，VM の性能が大幅に低下する．

このような問題に対して，VM のメモリを分割して複数の小さなホストに転送する分割マイグレーション [1] [2] が提案されている．分割マイグレーションでは，VM 本体とアクセスされることが予測されるメモリデータを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データはサブホストに転送する．マイグレーション中にページングが発生しないため，マイグレーション性能を向上させることができる．また，VM が必要とするメモリデータの多くはメインホストに転送されるため，マイグレーション直後の性能も改善される．しかし，ホスト間でのリモートページングの性能を向上させるには高価な高速ネットワークが必要になる．また，ネットワークやサブホストに障害が発生すると VM の実行を継続できなくなる．

本稿では，VM 専用の仮想メモリを用意して分割マイグレーションの技術を適用することにより，VM マイグレーションの高速化を実現するシステム VMemDirect を提案する．VMemDirect では安価になってきている高速な NVMe 上に VM 専用のスワップ領域を作成し，仮想メモリと VM マイグレーションを密に連携させる．メモリデータを物理メモリまたは NVMe に直接転送することにより，マイグ

[1] 九州工業大学
Kyushu Institute of Technology

レーション中にページングを発生させないようにして性能低下を防ぐことができる．また，VM が必要とするメモリデータを予測して物理メモリに転送することにより，マイグレーション後の性能も向上させることができる．

VMemDirect を KVM に実装し，VM 専用の仮想メモリを用いたマイグレーションを実現した．VM 専用スワップ領域はスパースファイルを用いて作成し，ディスク容量を効率よく利用する．移送元ホストでメモリアクセス履歴に基づいてメモリ格納先を決定し，再送時も同じ格納先にメモリデータを直接転送する．マイグレーション後は Linux の userfaultfd 機構を利用してページングを行う．実験により，VMemDirect は従来の仮想メモリを使用した場合よりも，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確認できた．

以下，2 章では従来の仮想メモリを用いたマイグレーションと分割マイグレーションの問題点について述べる．3 章では VM 専用の仮想メモリと連携することにより VM マイグレーションの高速化を実現する VMemDirect について述べ，4 章でそのシステムの実装について述べる．5 章では VMemDirect を用いて行った実験について述べる．6 章で関連研究について触れ，7 章で本稿をまとめる．

2. 大容量メモリを持つ VM のマイグレーション

VM を提供する IaaS 型クラウドの普及に伴い，大容量のメモリを持つ VM が提供されるようになってきている．例として，Amazon EC2 では 4TB のメモリを持つ VM が提供されている．マイグレーションを行うためには，移送先ホストに VM のメモリよりも大きな空きメモリが必要となる．しかし大容量メモリを持つ VM では，マイグレーションのための十分な空きメモリを持つホストを移送先として確保し続けることは，コスト面で大きな負担となる．マイグレーションが行えない場合には，ホストのメンテナンスの際に VM を停止させなければならなくなる．

2.1 仮想メモリを用いたマイグレーション

移送先ホストに十分な空きメモリがない場合でも，図 1 のように移送先ホストの仮想メモリを用いることで，VM マイグレーションを行うことが可能である．仮想メモリを用いることで VM のメモリの一部がディスク上のスワップ領域に透過的に格納され，マイグレーションに必要なメモリ量を確保することができる．VM がスワップ領域のメモリデータを必要とした時は，そのメモリデータを物理メモリ上に転送してページインを行う．代わりに，物理メモリ上のメモリデータをスワップ領域に転送してページアウトを行う．ディスクの読み書きはメモリと比べて低速であるため，ページングの発生により VM の性能は低下するが，最近の高速な NVMe を用いることにより，ページングのオーバヘッドを削減することができる．

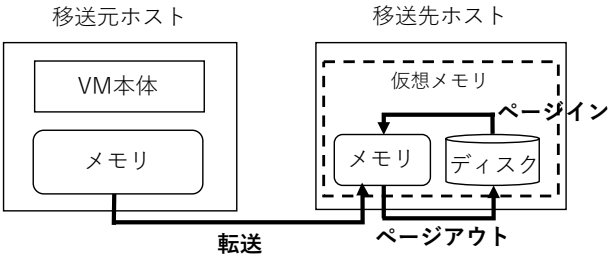

図 1 仮想メモリを用いたマイグレーション

しかし，従来の仮想メモリは VM マイグレーションとの相性が悪い．マイグレーションではまず VM のメモリ全体を転送するが，移送先ホストの物理メモリに空きがなくなると，それ以後は物理メモリからスワップ領域へのページアウトが発生する．これにより大量のページアウトが発生し，マイグレーション時間の増加や高負荷の原因となる．また，VM のメモリの再送時にもページングが発生する．再送により上書きされるメモリが移送先ホストのスワップ領域に存在する場合，物理メモリ上にそのメモリをページインしてから更新する必要がある．同時にページアウトも必要となる．大容量メモリを持つ VM では最初の VM 全体のメモリ転送に時間がかかり，その間に更新されるメモリ量も増大するため，このページングの影響も大きくなる．

マイグレーションの最終段階の VM 停止時に再送によるページングが多発すると，VM のダウンタイムを増大させる．また，KVM のように仮想化ソフトウェアのメモリが VM のメモリと同じ仮想メモリ機構で管理されている場合，そのメモリはマイグレーション中にページアウトされる可能性が高い．仮想化ソフトウェアのメモリの大部分は VM のメモリ転送中には利用されないためである．そのため，マイグレーションの最終段階で仮想デバイスの復元を行う際にページインが頻発する．マイグレーション後は，頻繁に更新されるメモリデータは再送により物理メモリ上にあることが多いが，読み込みのみが頻繁に行われるメモリデータは再送されないためスワップ領域にある可能性が高い．そのため，VM 再開後に大量のページングが発生する可能性がある．したがって，従来の仮想メモリを用いてマイグレーションを行うと，ページングにより性能が大幅に低下する．

2.2 分割マイグレーション

分割マイグレーション [1] [2] は，図 2 のように VM のメモリを分割して複数の小さなホストに転送する．これにより，十分な空きメモリを持つ大きなホストがなくてもマイグレーションを行うことができる．マイグレーション時には，CPU やデバイスの状態など VM の核となる情報を移送先メインホストに転送する．また，今後のアクセスが予測されるメモリデータは可能な限りメインホストに転送す

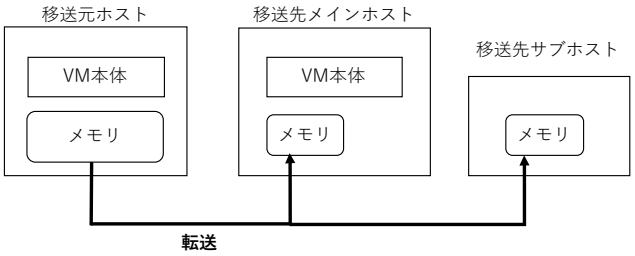

図 2　分割マイグレーション

る．メインホストに入りきらなかった残りのメモリデータはサブホストに転送する．マイグレーション後はメインホスト上で VM を稼働させる．マイグレーション後に VM がサブホスト上に存在するメモリデータを必要とした時は，サブホストからメインホストにそのメモリデータをページインし，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データをサブホストにページアウトする．

分割マイグレーションではマイグレーション中にページングが発生しないため，マイグレーション性能の低下を抑えることができる．また，VM が必要とするメモリはメインホストに転送されるため，マイグレーション後のページング頻度も抑えられる．しかし，リモートページングの際にネットワーク転送を行うオーバヘッドが大きく，性能を向上させるためには高価な高速ネットワークが必要となる．また，リモートページングによりネットワーク帯域を消費し，サービスの実行への影響も及ぶ．ネットワークやサブホストに障害が発生した場合は VM の実行を継続できなくなるという問題もある．

3. VMemDirect

本稿では，VM 専用の仮想メモリを用意して分割マイグレーションの技術を適用することにより，VM マイグレーションの高速化を実現するシステム VMemDirect を提案する．VMemDirect では移送先ホストの NVMe 上に VM 専用のスワップ領域を作成し，仮想メモリとマイグレーションを密に連携させる．最近の NVMe は高速かつ安価になってきており，高価な高速ネットワークとサブホストを用意するよりもコストを下げることができる．また，マイグレーション後の VM の実行はネットワークや周囲のホストの障害の影響を受けない．

3.1 直接メモリ転送

VMemDirect は VM のメモリを移送先ホストに転送する際に，従来の仮想メモリのページングに任せるのではなく，物理メモリまたは NVMe 上の VM 専用スワップ領域のいずれかにメモリデータを直接転送する．移送元ホストではマイグレーションを開始する際にメモリデータの格納先を決定し，メモリを再送する際も同じ格納先に転送する．

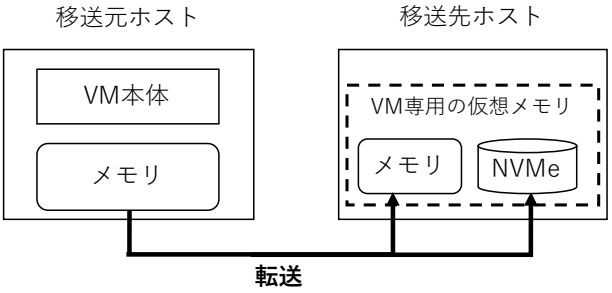

図 3　VMemDirect によるマイグレーション

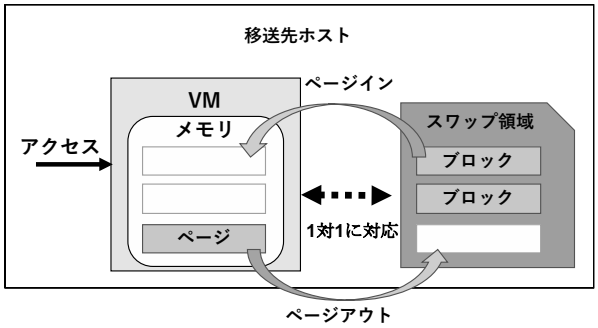

図 4　VM 専用の仮想メモリ

これにより，図 3 のようにマイグレーション中にページングが発生しないようにすることができる．物理メモリ上のデータはページアウトされることはなく，スワップ領域にあるメモリデータはページインされることはない．メモリの再送時には，物理メモリまたはスワップ領域のデータを直接上書きすることで更新する．

メモリデータの格納先は VM のメモリアクセス履歴に基づいて決定する．VM がアクセスすることが予測されるメモリデータは物理メモリに転送し，それ以外の物理メモリに入りきらないメモリデータはスワップ領域に転送する．これにより，更新によって再送されるメモリデータは移送先ホストの物理メモリ上に格納される可能性が高くなり，再送時に NVMe 上のスワップ領域に書き込むことによるオーバヘッドを削減することができる．マイグレーション後は頻繁に更新されるメモリデータだけでなく頻繁に読み込みが行われるメモリデータも物理メモリに格納されている可能性が高いため，VM 再開後のページングの発生を抑制することができる．

3.2 VM 専用の仮想メモリ

VMemDirect では仮想化ソフトウェアが VM ごとに仮想メモリを提供する．この仮想メモリは VM のメモリだけをページングの対象とするため，仮想化ソフトウェアのメモリがページアウトされてしまうことによる性能低下を防ぐことができる．そのために，VM 単位で VM のメモリと同じサイズのスワップ領域を NVMe 上に作成する．図 4

のように，VM のメモリの各領域はスワップ領域のブロックに 1 対 1 に対応づけられる．VM のメモリデータが物理メモリ上にない場合だけスワップ領域の対応するブロックにデータが格納され，それ以外のブロックは実データを持たない．

VM が物理メモリ上に存在しないメモリ領域にアクセスすると，仮想化ソフトウェアがそれを検出してページングを行う．まず，スワップ領域の対応するブロックを読み込み，VM の対応するメモリページにそのデータを書き込むことでページインを行う．同時に，スワップ領域の対応するブロックは無効にする．次に，VM のメモリアクセス履歴に基づいて，今後アクセスされないことが予測されるメモリ領域を選ぶ．そのメモリデータをスワップ領域に書き込み，VM のメモリ領域を解放することによりページアウトを行う．

4. 実装

VMemDirect を QEMU-KVM 2.4.1 および Linux 4.11 に実装した．

4.1 VM 専用スワップファイル

VMemDirect では，VM 専用スワップ領域を図 5 のように，VM のメモリと同じサイズを持つスパースファイルと呼ばれる特殊なファイルとして作成する．スパースファイルはホールと呼ばれる実データを持たないブロックを持つことができる．空のブロックであることを表すメタデータをディスクに書き込むことで，ホールの領域については実際のディスク容量を消費しないファイルを作成することができる．VM のメモリ領域とスパースファイルのブロックを 1 対 1 に対応づけて管理し，VM のメモリデータは物理メモリとスパースファイルのいずれかにだけ保持する．VM のメモリ領域が物理メモリ上に存在する場合には対応するスパースファイルのブロックをホールにすることでディスクの使用量を削減する．また，ページイン時にスパースファイルからメモリデータを取得した後，そのブロックをスパースファイルから削除してホールにすることでディスク容量を効率よく使用する．

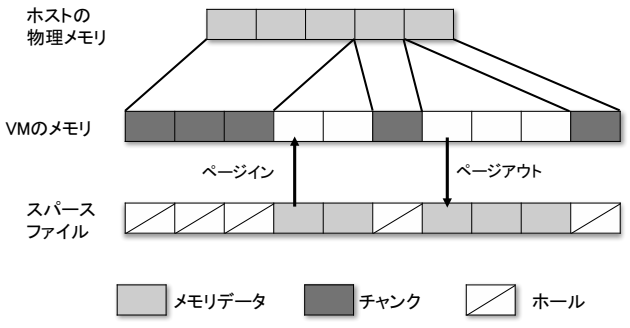

図 5 VM 専用スワップファイル

VMemDirect はスワップファイルの読み書きの際にページキャッシュが作られないように，ダイレクト I/O を用いてスワップファイルにアクセスする．物理メモリの空きメモリの大部分は VM のメモリデータを格納するために使われるため，ページキャッシュが作られると従来の OS の仮想メモリによって VM のメモリがページアウトされる可能性があるためである．また，ディスク帯域を最大限に活用するために，スワップファイルの読み書きはある程度大きなサイズ単位で行う．4KB のメモリページ単位では大幅に性能が低下するため，現在の実装では 256 ページのチャンク単位としている．

4.2 直接メモリ転送

マイグレーションを開始する際に，VMemDirect ではメモリアクセス履歴に基づいて VM のメモリデータの格納先を決定する．メモリの再送時に同一の格納先に転送されるようにするために，移送元ホストでは VM のメモリページ毎に格納先を管理する．そのために格納先ビットマップを作成し，移送先の物理メモリに転送する場合は 0 を，VM 専用スワップファイルに転送する場合は 1 をセットする．メモリ転送時は格納先ビットマップを参照して格納先の情報をメモリデータの付加情報として送信する．移送先でメモリデータを受信すると付加情報に従って物理メモリまたは VM 専用スワップファイルの対応するオフセットに直接格納する．メモリ転送中に VM のメモリが更新された場合は，格納先ビットマップを参照して同一の格納先に転送し，物理メモリまたは VM 専用スワップファイルを直接上書きすることで更新する．また，VM のメモリアクセス履歴の情報も送信し，移送先でメモリアクセス履歴を引き継ぐことができるようにする．

VMemDirect は VM のメモリデータをできるだけチャンク単位で VM 専用スワップファイルに書き込む．そのために，移送先ホストでは受信したページ単位のメモリデータを一時的に物理メモリに保存する．VM の最初のメモリ転送の間はメモリデータが順番に受信できるため，チャンクサイズ分のメモリデータを受信したらスワップファイルへの一括書き込みを行う．メモリ再送時には更新されたページだけが転送されるため，ページ単位での書き込みを行う．

4.3 VM 専用ページング

VMemDirect でのページングは Linux の userfaultfd 機構を用いて行う．マイグレーション完了時に QEMU-KVM が VM のページを userfaultfd 機構に登録する．図 6 のように VM が物理メモリに存在しないページへアクセスするとページフォールトが発生し，QEMU-KVM にイベントが通知される．次に，ページフォールトが発生した VM のメモリアドレスに対応するブロックを VM 専用スワップファイルから読み込む．そのメモリデータを userfaultfd

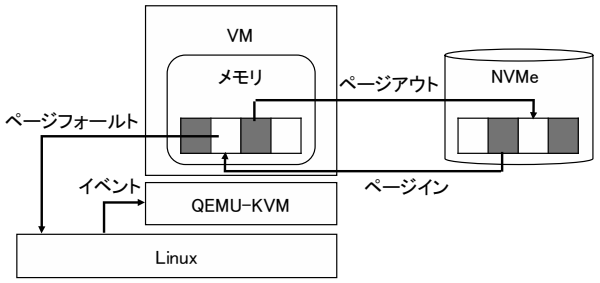

図 6 VMemDirect のページング機構

機構を用いて QEMU-KVM の対応するメモリページに書き込む．このメモリデータを VM 専用スワップファイルから削除してホールにすることでページイン処理が完了する．同時に，メモリアクセス履歴に基づいて今後使われる可能性が低いと予測される VM のメモリを物理メモリから選択する．VM 専用スワップファイルの対応するブロックに選択したデータを書き込み，VM のメモリページのマッピングを削除することでページアウトを行う．メモリデータの取得とマッピングの削除をアトミックに行うために，S-memV [1] [2] で開発された userfaultfd の拡張を用いた．

ページアウトするページを選択する時に物理メモリ上にあるページを選択できるようにするため，移送元ホストと同様にビットマップを作成してメモリデータの格納先の管理を行う．マイグレーション中のメモリ受信時に格納先が物理メモリの場合にはビットマップに 0 を，スワップファイルの場合には 1 をセットする．ページインが発生するとそのページに対応するビットを 0 に，ページアウトが発生するとそのページに対応するビットを 1 に更新する．

4.4 メモリアクセス履歴

VMemDirect では VM のメモリアクセス履歴を管理するために，VM の拡張ページテーブル（EPT）をたどることで各ページのメモリアクセスに関する情報を取得する．S-memV で開発された機構を用いて，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し，メモリアクセス履歴を更新する．この時，VM のメモリサイズに応じたビットマップを確保し，ioctl の引数として KVM に渡す．KVM は VM のすべてのページに対して EPT をたどり，ページテーブルエントリ（PTE）を取得する．ページへのアクセスがあると PTE のアクセスビットが 1 にセットされるため，渡されたビットマップの対応するビットにアクセスビットの値をセットする．次の期間のアクセスを記録できるようにするために，PTE のアクセスビットは 0 にクリアする．

メモリアクセス履歴は LRU 近似アルゴリズムであるエージングアルゴリズムを用いて作成する．図 7 のように各ページに 8 ビットを割り当ててメモリアクセス履歴を管理し，8 ビットの最上位ビットに PTE から取得したアクセスビットを記録する．また，定期的にメモリアクセス履歴を

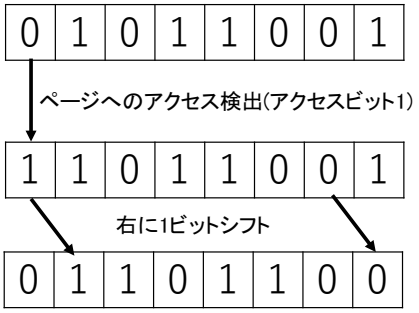

図 7 エージングアルゴリズムを用いたメモリアクセス履歴の管理

右に 1 ビットシフトし，再び最上位ビットにアクセスビットを記録する．これにより最近アクセスされたページほどメモリアクセス履歴の値が大きくなるため，今後使われる可能性の高いページを予測することが可能となる．

VMemDirect ではマイグレーション開始時にメモリアクセス履歴を用いて VM のメモリデータをチャンク単位に分割する．格納先を決定する時にそれぞれのチャンクに含まれるページの中でメモリアクセス履歴の値が最大のものを探し，その値が大きいチャンクから順に移送先の物理メモリに割り当てる．チャンク内のページの最大のメモリアクセス履歴を用いるのは，最も最近アクセスされたページを重視するためである．

5. 実験

VMemDirect の有用性を示すために，マイグレーション性能とマイグレーション後の VM の性能を調べる実験を行った．比較として移送先ホストに十分なメモリがある場合と従来の仮想メモリを用いた場合についても調べた．仮想メモリが用いるスワップ領域として Samsung NVMe SSD 960 EVO を使用した．移送元ホストと移送先ホストには，Intel Xeon E3-1226 v3 の CPU，16GB のメモリを搭載したマシンを 2 台用い，10 ギガビットイーサネットで接続した．仮想メモリを用いる場合には移送先ホストの空きメモリが 1GB になるように調整した．ホスト OS には Linux 4.11 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，メモリは 2GB 割り当てた．ページ内のデータがすべて 0 の時の最適化は無効にし，VM のメモリ全体が使われている状態にした．

5.1 マイグレーション性能

VMemDirect のマイグレーション性能を調べるために，マイグレーション時間とダウンタイムを測定した．VMemDirect では，VM 専用スワップ領域にページ単位でアクセスする場合とチャンク単位でアクセスする場合について測定した．チャンクサイズは 1MB（256 ページ）とした．マイグレーション時間は図 8 のようになり，メモリが十分にあ

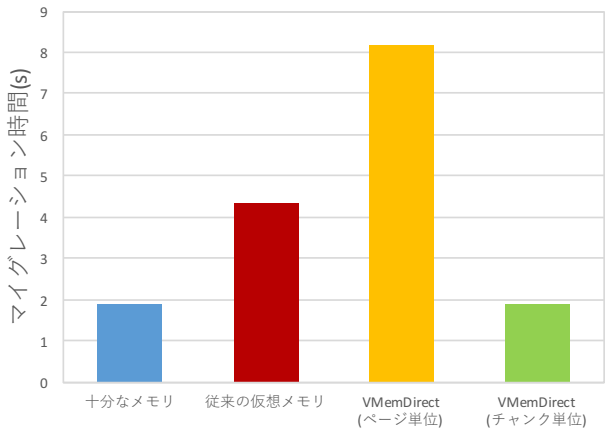

図 8　マイグレーション時間

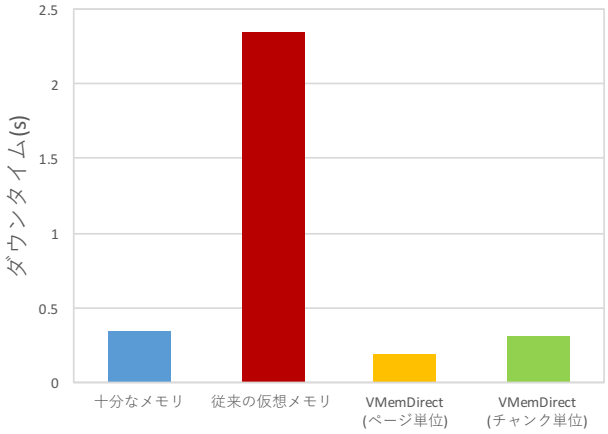

図 9　ダウンタイム

る場合と比較して，従来の仮想メモリを用いた場合は 2.3 倍に増加した．一方，VMemDirect においてページ単位で VM 専用スワップファイルにアクセスした場合はメモリが十分にある場合の 4.3 倍のマイグレーション時間となった．VMemDirect の性能が従来の仮想メモリより低下しているのは，ページ単位でのアクセスでは NVMe の性能を活かせていないことが挙げられる．また，VMemDirect ではダイレクト I/O を用いてメモリデータの書き込みを同期的に行っているのに対し，従来の仮想メモリではページアウト処理を非同期に行えているためだと考えられる．これに対し，VMemDirect においてチャンク単位で VM 専用スワップ領域にアクセスした場合は，メモリが十分にある場合よりマイグレーション時間が 9 ミリ秒の増加となり，ほぼ同程度の性能となった．

ダウンタイムは図 9 のようになり，メモリが十分にある場合と比べて，従来の仮想メモリを用いた場合は 6.3 倍に増加した．これは仮想デバイスの復元が主な原因であり，マイグレーション中にページアウトされた仮想化ソフトウェアのメモリをページインしているためである．VMemDirect においてページ単位で VM 専用スワップ領域にアクセスした場合は，メモリが十分にある場合と比べてダウンタイムが 46%減少し，チャンク単位でアクセスした場合は 9%減少した．VMemDirect では VM 専用の仮想メモリを提供することで仮想化ソフトウェアのメモリがページアウトされることを防ぐことができたため，ダウンタイムが増加しなかったと考えられる．逆にダウンタイムが減少しているのは，QEMU-KVM によるダウンタイムの見積もりが正確に行えなくなったためである．QEMU-KVM は残りのメモリを 0.3 秒で転送できると判断した時に VM を停止させる．見積もり時に VM 専用スワップ領域へ転送したメモリデータが多かった場合にそれを基に見積もりを行うと，実際には物理メモリに転送されるメモリデータが多かった場合に見積もりよりも早く転送が完了する．

5.2　スワップ領域の性能の影響

仮想メモリが用いるスワップ領域として，HDD，SSD，NVMe を使用した場合のマイグレーション性能を調べた．VMemDirect はメモリをチャンク単位で分割した場合について測定した．HDD には WD5000AAKX，SSD には Crucial MX300 を SATA 3 で接続して使用した．実験結果を図 10 と図 11 に示す．マイグレーション時間は従来の仮想メモリと比較して VMemDirect ではそれぞれ 79%，20%，56%短縮することができた．HDD を用いた場合，ページングによるオーバヘッドの影響が大きく，VMemDirect によるページング抑制の効果が大きく現れた．SSD を用いた場合にあまり性能が改善できない原因は調査中である．ダウンタイムは従来の仮想メモリと比べてそれぞれ 95%，87%，86%短縮することができた．VMemDirect において，より高速なスワップ領域を用いたほうがダウンタイムが長くなったのは，遅いスワップ領域を用いた場合のほうが QEMU-KVM による見積もりがより不正確になったためと考えられる．従来の仮想メモリにおいて，NVMe を用いたほうが SSD を用いた場合よりダウンタイムが長い原因は調査中である．

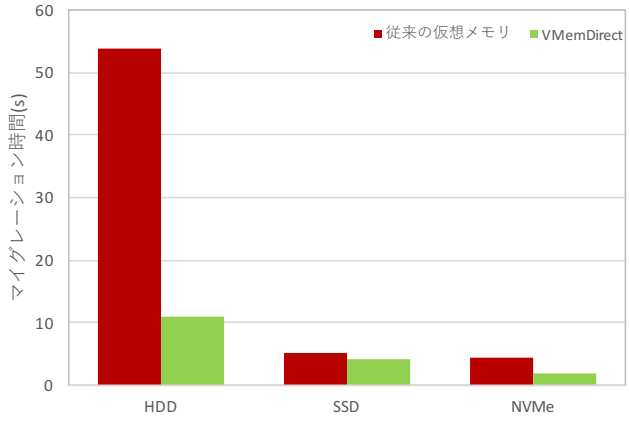

図 10　スワップ領域の性能のマイグレーション時間への影響

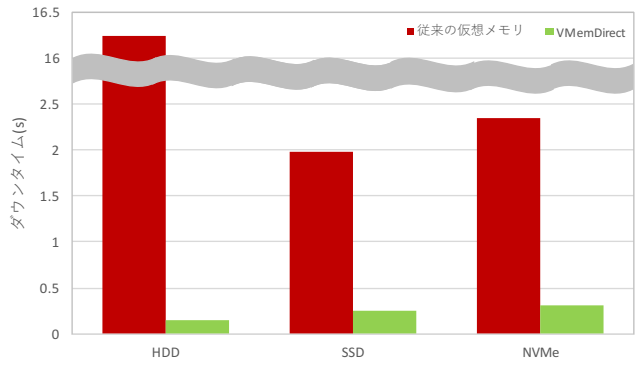

図 11 スワップ領域の性能のダウンタイムへの影響

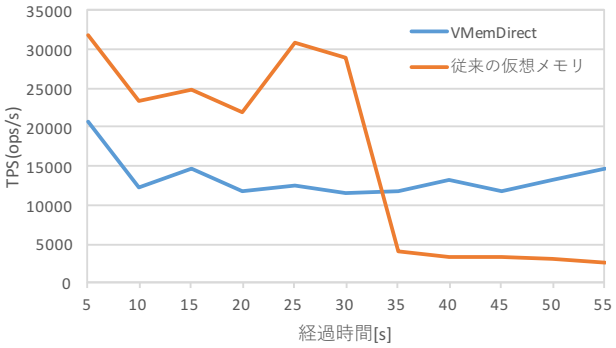

図 12 マイグレーション後の memcached の性能

5.3 マイグレーション後の VM の性能

ページングの性能を調べるために，マイグレーション後に VM 内で memcached [12] を動作させ，memaslap ベンチマークを用いて測定を行った．memslap の set と get の比率を 0.6 対 0.4 に設定し，memcached が確保するメモリ量を 1GB とした．実験結果を図 12 に示す．従来の仮想メモリでは memaslap 開始から 30 秒後までは高い性能を保ったが，35 秒後に性能が急激に悪化した．これは memcached に格納されたデータが物理メモリの空き容量を超え，ページングが頻発するようになったためである．

一方，VMemDirect は従来の仮想メモリを用いた場合の性能が落ち込む前と比べて，平均で 52%の性能となった．これは VMemDirect が userfaulfd 機構を用いてプロセスレベルでページングを行なっているためだと考えられる．また，実装上の問題により VMemDirect はページングが頻発し始めたと思われる 60 秒後に停止し，実行を継続できなくなった．

6. 関連研究

vMotion はスワップ領域を移送先と共有するかどうかで異なるマイグレーション手法を用いる [6]．Unshared-Swap vMotion は移送先で異なるスワップ領域を用意し，マイグレーション時にはスワップ領域に格納されているメモリデータも移送先に転送する．一方，Shared-Swap vMotion はスワップファイルをネットワーク上の共有ストレージに格納し，マイグレーション時にはスワップ領域のメモリデータは転送しない．マイグレーション中のページングのために移送先では一時的なスワップ領域を用意し，マイグレーション後に 1 つのスワップ領域に統合する．後者の手法ではマイグレーションを高速化することができるが，常にネットワーク越しのページングが必要となる．

Agile ライブマイグレーション [5] では，cgroup ごとにスワップファイルを作成するカーネルパッチ [4] を用いて，Shared-Swap vMotion と同様に VM ごとのスワップ領域をネットワーク上に配置する．そのスワップ領域を用いて，アクセスされているメモリ領域であるワーキングセット以外は常にページアウトされた状態にする．マイグレーション時にはスワップ領域のメモリデータは転送せず，ワーキングセットメモリと CPU などの状態のみを転送する．マイグレーション後にスワップ領域をそのまま用いることによりマイグレーション性能を向上させることができる．しかし，ネットワークのスワップ領域にワーキングセット以外のメモリデータをすべて格納するため，ページングのオーバヘッドが増大する．

FlashVM [7] は SSD をページングに利用する仮想メモリである．FlashVM はディスク帯域を有効活用するために，HDD の場合よりも多くのページをまとめてページアウトする．SSD のランダム読み込みが高速であることを利用して，より有用なページをプリフェッチしてページフォールトを減らす．また，ページアウトされたページのデータを書き出すレートをきめ細かく調整することでページフォールトの遅延を減らす．

VSwapper [8] は仮想メモリを用いた VM の性能を改善する．この論文ではディスクからデータを読み込んだページがそのままページアウトされたり，ページアウトされたページ全体を書き換えたりする場合に仮想メモリの性能が低下することが示されている．VSwapper ではディスク I/O を監視して，変更されていないページはページアウト時にスワップ領域に書き込まないようにする．また，ページアウトされたページへの書き込みをバッファに一時保存し，ページ全体に書き込まれた場合にはスワップ領域から読み込まないようにする．このような最適化により最大で 10 倍の性能向上を達成している．この手法は VMemDirect でマイグレーションした後の VM にも適用することができる．

ExpEther を用いてイーサネット経由で SSD を接続し，スワップ領域として用いるシステムが提案されている [9]．ExpEther により，コンピュータの内部バスの PCI-Express（PCIe）をイーサネットで延長し，コンピュータの物理的な配置に縛られず SSD を拡張可能となる．ローカルメモリとイーサネット上の SSD 間で DMA を用いて高速なデータ通信を行い，OS によるページングを用いることでコン

ピュータのメモリ拡張が可能となる．

HPBD [10] は，Infiniband を用いる高性能なネットワークブロックデバイスである．HPBD をスワップ領域として用いてリモートページングを行うことで，プロセスが RDMA 経由でリモートサーバのメモリを高速かつ透過的に利用することができる．Infiniswap [11] も RDMA を用いたリモートページングシステムである．HPBD とは違い，RAM ディスクを用いずにリモートメモリを提供することで CPU 負荷を低下させている．また，動的にリモートメモリを管理し，リモートメモリに書き込んだデータを非同期にディスクにも書き込むことで耐障害性を向上させている．

7. まとめ

本稿では，VM 専用の仮想メモリと連携することにより VM マイグレーションの高速化を実現するシステム VMemDirect を提案した．VMemDirect は NVMe 上に VM 専用スワップ領域を作成し，VM がアクセスすることが予測されるメモリデータを移送先ホストの物理メモリに，それ以外を VM 専用スワップ領域に直接転送する．マイグレーション後は VM 専用の仮想メモリを用いて物理メモリと VM 専用スワップ領域の間でページングを行う．実験結果より，マイグレーションの性能低下を抑えられることを確認した．

今後の課題は，様々なアプリケーションを動作させた際のマイグレーション性能とマイグレーション後のアプリケーション性能を調べることである．また，S-memV [1] [2] の分割マイグレーションおよびリモートページングと性能を比較することで，NVMe と高速ネットワークを用いることによる利害得失を明らかにしたい．現在の実装では QEMU-KVM が VM のページングを行っているが，OS レベルで VM 単位のページングを行うことで性能が改善できると考えられる．さらに，潤沢な物理メモリを持ったホストへの統合マイグレーション [3] を効率よく行えるようにすることも計画している．

参考文献

[1] M. Suetake, H. Kizu, and K. Kourai Split Migration of Large Memory Virtual Machines In Proceedings of the 7th ACM SIGOPS Asia-Pacific Workshop of Systems (APSys 2016), 8 pages, 2016.
[2] M. Suetake, T. Kashiwagi, H. Kizu, and K. Kourai S-memV: Split Migration of Large-memory Virtual Machines in IaaS Clouds In Proceedings of the 2018 IEEE International Conference on Cloud Computing (CLOUD 2018), pp.285-293, 2018.
[3] 柏木 崇広, 末竹 将人, 光来 健一. IPmigrate: 複数ホストに分割された VM のマイグレーション手法. SWoPP2017, 2017.
[4] Y. Zhao. Per-cgroup swap file. http://lwn.net/articles/592923.
[5] U. Deshpande, D. Chan, T. Guh, J. Edouard, K. Gopalan, and N. Bila. Agile Live Migration of Virtual Machines. In IEEE International Parallel and Distributed Processing Symposium, 2016.
[6] I. Banerjee, P. Moltmann, K. Tati, and R. Venkatasubramanian. VMware ESX Memory Resource Management: Swap. VMware Technical Journal, Vol.3, No.1, 2014.
[7] M. Saxena and Michael M. Swift. FlashVM: Virtual Memory Management on Flash. In Proceedings of USENIX Annual Technical Conference, 2010.
[8] N. Amit, D. Tsafrir, and A. Schuster. VSWAPPER: A Memory Swapper for Virtualized Environments. In ACM Proceedings of International Conference on Architectural Support for Programming Languages and Operating Systems(ASPLOS'14), pages 349-366, 2014.
[9] J. Suzuki, T. Baba, Y. Hidaka, J. Higuchi, N. Kami, S. Uchida, M. Takahashi, T. Sugawara, and T. Yoshikawa. Adaptive Memory System over Ethernet. In Proceedings of USENIX Workshop on Hot Topics in Storage and File Systems 2010.
[10] S. Liang, R. Noronha, and Dhabaleswar K. Panda. Swapping to Remote Memory over InfiniBand: An Approach using a High Performance Network Block Device. In Proceedings of IEEE Cluster Computing, 2005.
[11] Yiwen Zhang Mosharaf Chowdhury Juncheng Gu, Youngmoon Lee and Kang G. Shin. Efficient Memory Disaggregation with InfInIswap. In Proceedings of the 14th USENIX Symposium on Networked Sys- tems Design and Implementation (NSDI'17), 2017.
[12] B. Fitzpatrick. memcached - A Distributed Memory Object Caching System. http://memcached.org/.