VMマイグレーションにおける効率的な移送先変更手法

緒方 彬人[1]　光来 健一[1]

概要：仮想マシン (VM) はマイグレーションによって別のホストに移動させることができるが，VM のメモリサイズに比例した時間がかかる．そのため，大容量メモリを持つ VM では，移送先ホストの負荷増大やネットワーク混雑など，マイグレーション開始時から状況が変化する可能性が高くなる．このような場合，移送先をより適したホストに変更することが望ましいが，移送先変更の際にはマイグレーションを一旦キャンセルする必要がある．それまでに転送された VM の状態はすべて破棄されるため，再度すべての状態を転送し直す必要が生じ，再び状況が変化するリスクがある．
本稿では，マイグレーションをキャンセルせずに移送先ホストを柔軟に変更できるシステム DCmigrate を提案する．DCmigrate はマイグレーションを部分的にキャンセルすることで，移送元ホストから未転送の VM の状態のみを新しい移送先ホストにシームレスに転送する．それに加えて，古い移送先ホストは移送元から受信済みの VM の状態を保持し続け，それを新しい移送先ホストに転送する．2 つのホストから VM の状態を並列に転送し，それぞれのホストからの転送量を最適化することにより，マイグレーションを効率化する．DCmigrate を libvirt と QEMU に実装し，移送先変更後のマイグレーション時間が短縮できることを確認した．

1. はじめに

IaaS 型クラウドだけでなく，PaaS 型クラウドや SaaS 型クラウドにおいても多くの場合，内部で仮想マシン (VM) を利用してサービスを提供している．近年，クラウドで利用される VM のメモリの大容量化が進んでいる．例えば，Amazon EC2 では最大 32TB のメモリを持つ VM が提供されている [1]．VM はマイグレーション技術を用いることで，ホストのメンテナンスや負荷分散の際に別のホストに移動させることができる．マイグレーションにかかる時間は VM のメモリサイズに比例するため，大容量メモリを持つ VM のマイグレーションにはより長い時間がかかることになる．

時間のかかるマイグレーション中には，移送先ホストの負荷が高くなったり，移送元ホストと移送先ホストの間のネットワークが混雑したりするなど，開始時と状況が変わることがある．このような状況変化はマイグレーション時間が長いほど発生しやすいため，大容量メモリを持つ VM の場合，状況変化が発生する可能性が高くなる．このような場合，マイグレーション時間が長くなったり，マイグレーション後に移送先で意図した性能を発揮できないといった不都合が生じるため，移送先を変更することが望ましい．しかし，移送先変更の際にはマイグレーションを一旦キャンセルする必要があり，それまでに転送された VM の状態はすべて破棄される．その結果，再度すべての状態を転送し直す必要が生じ，再び状況が変化するリスクがある．

そこで本稿では，VM マイグレーションをキャンセルせずに移送先ホストを柔軟に変更できるシステム DCmigrate を提案する．DCmigrate は移送元ホストで移送先を新しいホストに切り替え，未転送の VM の状態だけを新しい移送先ホストに転送する．それに加えて，変更前の古い移送先ホストは移送元ホストから受信済みの VM の状態を新しい移送先ホストに転送する．これにより，移送元ホストと古い移送先ホストの両方から新しい移送先ホストへ VM の状態を並列に転送することができる．その結果，従来手法と比べて，移送先ホストを変更した後のマイグレーション時間を短くすることができる．そのため，移送先変更後のマイグレーション中に再び，状況が変わって，さらに移送先を変更する必要が生じる可能性を減らすことができる．

DCmigrate は移送先を変更する際に，実行中のマイグレーションを部分的にキャンセルする．それにより，移送元ホストは新しい移送先に対して VM の状態転送をシームレスに継続する．同様に，古い移送先ホストは受信済みの VM の状態を破棄せずに保持し続け，新しい移送先に転送できるようにする．新しい移送先ホストはこれら 2 つのホストから VM の状態を並列に受信し，VM が稼働中の移

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

送元ホストからの状態を優先することで VM の状態が最新になるように統合する．また，移送元ホストはワークスティールを行うことにより，古い移送先からまだ転送されていない VM の状態の一部を代わりに転送する．これにより，マイグレーション時間が最小化されるように転送量を調整する．

DCmigrate を libvirt[7] と QEMU[2] を拡張することで実装し，マイグレーション中の移送先変更を実現した．実験の結果，移送元と移送先のネットワークが両方とも混雑した場合には，従来手法と比べて，移送先変更後のマイグレーション時間を短縮できることが分かった．一方，移送先のネットワークだけが混雑した場合にはマイグレーション時間は短縮されなかったが，移送元のネットワークの帯域使用率を抑えることはできることも分かった．

以下，2 章では VM マイグレーション中に状況が変化した際の問題点について述べる．3 章では実行中のマイグレーションの移送先ホストを変更可能にする DCmigrate を提案する．4 章では DCmigrate の実装について述べる．5 章では DCmigrate を用いて移送先ホストを変更した実験について述べる．6 章で関連研究に触れ，7 章で本稿をまとめる．

2. VM マイグレーション中の状況の変化

近年，クラウドで利用される VM のメモリの大容量化が進んでいる．VM はマイグレーション技術を用いることにより，VM が稼働しているホストから別のホストへ移動させることができる．この技術を活用することで，ホストのメンテナンスを行う際でも，稼働中の VM を停止させることなく作業を行うことが可能となる．また，ホストが高負荷になった場合には，VM を他のホストに移動させることで負荷分散を行い，システム全体の安定性を高めることができる．一般に，マイグレーションにかかる時間は VM の持つメモリのサイズに比例する．そのため，大容量メモリを持つ VM では非常に長い時間がかかってしまう．

VM マイグレーションは移送元ホストから移送先ホストに VM の状態を転送することによって行われる．マイグレーションを開始すると，まず，移送先ホストにおいて空の VM を作成し，VM の状態が送られてくるのを待つ．移送元ホストがメモリデータを移送先ホストに転送すると，移送先ホストはそのデータを VM のメモリに書き込む．マイグレーション中に移送元ホストの VM によって更新されたメモリについては，そのデータを移送先ホストに再送する．転送すべきメモリデータが閾値を下回ったら移送元の VM を停止させ，残りのメモリデータおよび仮想デバイスの状態を転送する．移送先ホストは受信したすべての VM の状態を VM に反映させ，VM を再開する．

マイグレーションにおいては，負荷が低くネットワークが空いているホストが移送先として選択されるのが一般的である．しかし，マイグレーション中に移送先ホストの負荷が高くなったり，ネットワークが混雑するようになったりして，マイグレーション開始時から状況が変化することがある．このような状況になった場合，マイグレーション時間が非常に長くなってしまう上，マイグレーション後の VM も十分な性能を発揮できなくなる可能性がある．マイグレーション中の状況の変化はマイグレーション時間が長いほど起こりやすくなるため，大容量メモリを持つ VM では状況が変化する可能性が高くなる．

マイグレーションに影響を及ぼす状況変化が発生した場合には，より適した移送先ホストがあるのであれば移送先を変更することが望ましい．移送先として，より空いているネットワークに接続されたホストがある場合，移送先を変更することでマイグレーションを高速に完了させることができる．また，マイグレーション後の VM のネットワーク性能も向上する．より負荷の低いホストがある場合，移送先を変更することでマイグレーション後にリソースを十分に確保でき，VM の処理性能を向上させることができる．

しかし，マイグレーション中に移送先を変更するには，実行中のマイグレーションを一旦キャンセルする必要がある．マイグレーションをキャンセルするとそれまでに移送先に転送した VM の状態はすべて破棄されてしまうため，新しい移送先に対して一からマイグレーションをやり直すことになる．再度，VM のすべての状態を転送し直す必要があるため，移送先変更後のマイグレーションでも VM のメモリサイズに比例したマイグレーション時間がかかる．その結果，再度，状況が変化してしまうリスクがある．そのような場合には，再び移送先変更を試みるか，より長い時間がかかることを許容してマイグレーションを継続しなければならなくなる．

3. DCmigrate

本稿では，実行中の VM マイグレーションをキャンセルせずに移送先を柔軟に変更することにより，効率のよいマイグレーションを実現する DCmigrate を提案する．DCmigrate は移送先を変更する際に図 1 のように，移送元ホストで移送先を新しいホストに切り替え，未転送の VM の状態のみを新しい移送先ホストにシームレスに転送する．それに加えて，移送先を変更する前の古い移送先ホストは移送元から受信済みの VM の状態を破棄せずに保持し続け，新しい移送先に転送する．これにより，移送元と古い移送先の 2 つのホストから VM の状態を並列に転送することができる．

DCmigrate を用いて VM の状態を並列に転送することで，実行中のマイグレーションをキャンセルして一からやり直す従来手法と比べて，移送先ホストを変更した後のマイグレーション時間を短縮することができる．これにより，移送先変更後のマイグレーション中に状況が変化して，

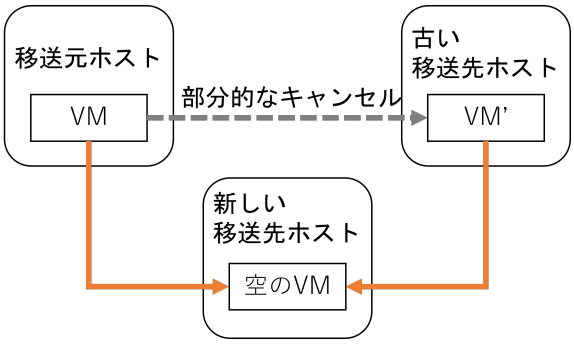


図 1: DCmigrate による移送先変更の流れ

再び移送先の変更が必要になるリスクを減らすことができる．また，移送元ホストをメンテナンスする際には作業により早く取り掛かることができるようになり，負荷分散を行う際には性能をより早く改善することができるようになる．それに加えて，移送元ホストから転送しなければならない VM の状態を減らすことができるため，移送元ホストの負荷を減らすとともにネットワーク負荷を分散させることもできる．

DCmigrate は移送先を変更する際に，実行中のマイグレーションを部分的にキャンセルする．移送元ホストでは，古い移送先へのマイグレーションジョブを終了させるが，VM の状態の転送は継続する．そして，新しい移送先へのマイグレーションジョブを開始した後で，VM の状態の転送先を切り替える．一方，古い移送先ホストでは，移送先としてのマイグレーションジョブを終了するが，マイグレーション先として作成された VM は削除しない．これにより，VM が受信済みの状態を保持し続けることができる．その後，移送元としてのマイグレーションジョブを開始し，新しい移送先に VM の状態を転送する．

新しい移送先では，移送元と古い移送先から並列に受信した VM の状態を統合し，移送元の最新の状態をもつ VM を再構築する．ライブマイグレーションの場合，マイグレーション中に移送元で実行されている VM が状態を更新する．更新された状態は新しい移送先に再送されるため，新しい移送先は移送元と古い移送先の両方から重複して状態を受信する可能性がある．このような場合には，移送元の状態の方が新しいため，移送元から受信した VM の状態を優先する．すでに古い移送先から状態を受信していた場合には移送元からの状態で上書きする．まだ受信していなかった場合には，その後で古い移送先から受信した状態は破棄する．

DCmigrate は移送元と古い移送先からの転送量を調整することで，移送先変更後のマイグレーションを最適化する．古い移送先が保持している VM の状態はもともと移送元から転送されたものであるため，移送元も同じか，より新しい状態を保持している．この特性を利用して，移送元から VM の状態の転送が完了した際に，古い移送先に未転送の状態が残っていればその一部を譲り受ける．そして，移送元が保持している状態を古い移送先の代わりに新しい移送先に転送する．このワークスティールを繰り返すことにより，移送元と古い移送先はそれぞれの転送速度に合わせて，マイグレーション時間が最短になるように並列に状態転送を行うことができる．

4. 実装

DCmigrate を libvirt[7] と QEMU[2] を拡張することによって実装した．

4.1 libvirt の拡張

libvirt を用いるシステムでは，各ホストで VM を管理するためのサーバである libvirtd が動作する．VM のマイグレーションを実行する際には，virsh などの libvirt クライアントが libvirtd に対してリモート手続き呼び出し (RPC) を実行する．libvirtd は仮想化ソフトウェアの QEMU に対して QMP と呼ばれるプロトコルを用いてコマンドを送信し，マイグレーション処理を行う．移送先の変更が発生する場合，移送元ホストと移送先ホストに加えて，新しい移送先を含めた 3 つのホストに対して RPC を実行する必要がある．

libvirt は以下の 5 つのフェーズからなるプロトコルを用いてマイグレーションを実行する．

Begin 移送元ホストで実行され，移送先ホストとの接続の確立や認証情報の検証，マイグレーションを行う VM の情報の収集を行う．

Prepare 移送先ホストで実行され，QEMU の立ち上げを行った後，移送元で収集された情報を受け取る．この情報を基に移送先で空の VM を作成する．

Perform 移送元ホストで実行されるマイグレーションのメインフェーズであり，VM の状態の転送を行う．

Finish 移送先ホストで実行され，移送元からすべての VM の状態を受け取るまで受信を続ける．完了後に移送先で VM の動作を開始する．

Confirm 移送元ホストで実行され，マイグレーションが完了したことを確認する．移送元の QEMU を終了させ，不要となったリソースの解放等を行う．

DCmigrate はこのマイグレーションプロトコルを拡張し，Perform フェーズと同時に古い移送先ホストで実行される **Forward** フェーズを導入することで，移送先変更を伴うマイグレーションを実現する．

4.1.1 Begin フェーズでの部分的なキャンセル

DCmigrate における Begin フェーズでは，図 2 のように移送元の libvirtd において実行中のマイグレーションを部分的にキャンセルする．通常のキャンセルとは異なり，実行中のマイグレーションジョブは終了させるが，QEMU に対してキャンセルのためのコマンドを送信しない．これに

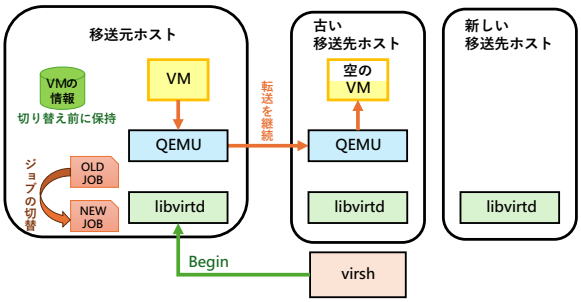


図 2: Begin フェーズでの部分的なキャンセル

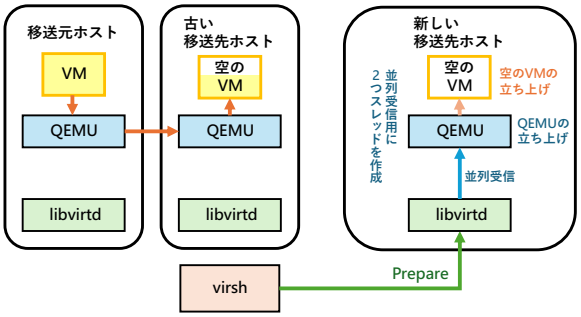


図 3: Prepare フェーズでの並列受信の準備

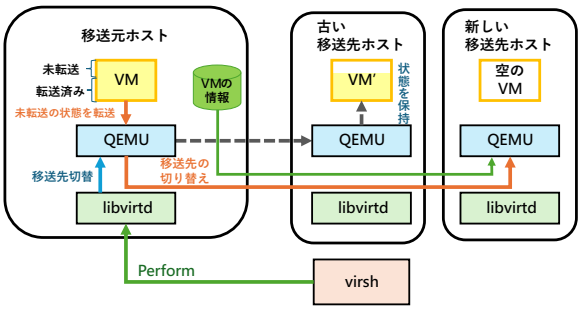


図 4: Perform フェーズでの移送先切り替え

より，QEMU には VM の状態転送を中断することなく継続させる．その後は，通常のマイグレーション時と同様の処理を行う．

4.1.2 Prepare フェーズでの並列受信の準備

DCmigrate における Prepare フェーズでは，図 3 のように新しい移送先の libvirtd が通常のマイグレーション時と同様に QEMU を立ち上げる．QEMU が空の VM を作成すると，QEMU に対して並列待受コマンドを送信する．このコマンドにより，QEMU は移送元と古い移送先の両方から VM の状態を受信する準備を行う．

4.1.3 Perform フェーズでの移送先切り替え

DCmigrate における Perform フェーズでは，図 4 のように移送元の libvirtd が QEMU に対して移送先切替コマンドを送信し，移送先を新しいホストへと切り替える．QEMU は未転送の VM の状態転送を継続し，libvirtd は転送完了を待つ．

4.1.4 Forward フェーズでの受信済み状態の転送

DCmigrate で新たに追加された Forward フェーズは Perform フェーズと並列に実行される．このフェーズでは，図 5 のように古い移送先の libvirtd において実行中のマイグレーションを部分的にキャンセルする．通常のキャンセルとは異なり，実行中のマイグレーションジョブは終了させるが，QEMU は終了させない．その後，QEMU に対して状態転送コマンドを送信し，受信済みの VM の状態を新しい移送先に転送させる．

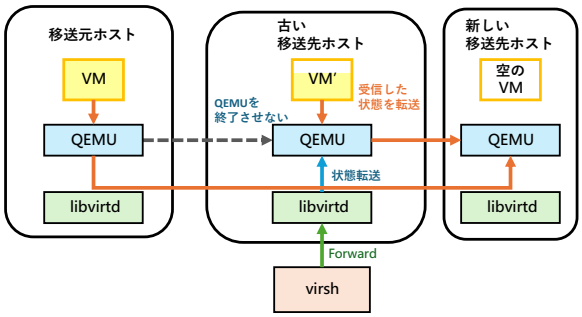


図 5: Forward フェーズでの受信済み状態の転送

4.2 QEMU での実装

QEMU は libvirtd から送信されるコマンドに応じてマイグレーション処理を行う．この節では，移送元，古い移送先，新しい移送先それぞれのホストにおける QEMU の拡張について説明する．

4.2.1 移送元ホストでの処理

Perform フェーズに移送元の QEMU が libvirtd から移送先切替コマンドを受け取ると，libvirtd から提供された情報を基に，現在の移送先の QEMU との間のネットワーク接続を新しい移送先の QEMU との間のネットワーク接続に切り替える．通常のマイグレーションの開始時には，移送先でマイグレーション先となる空の VM を作成するために，VM に関する様々な情報を移送先の QEMU に転送する．移送先変更時には新しい移送先の QEMU に同じ情報を転送し直す必要があるが，VM に関する情報を再度，収集するのは無駄である．そのため，移送元の QEMU は通常のマイグレーション開始時に転送した VM 情報を保存しておき，移送先変更時にはそれを再送するだけで済ませる．

その後，新しい移送先に対して VM の状態の転送を再開するが，この転送処理は元のマイグレーション処理を引き継ぐ形で行う．そのため，移送元の QEMU から新しい移送先の QEMU に転送するのは，古い移送先にまだ転送されていない状態となる．VM のメモリページをすべて転送し終わると，マイグレーション中に変更されたページの再送を行う．そして，転送すべき残りのページ数が閾値を下回るとマイグレーションの最終段階に入り，VM の実行を停止して VM の残りの状態を転送する．転送が完了すると新しい移送先で VM が実行を開始し，Confirm フェーズに移送元の QEMU は libvirtd によって終了させられ，VM

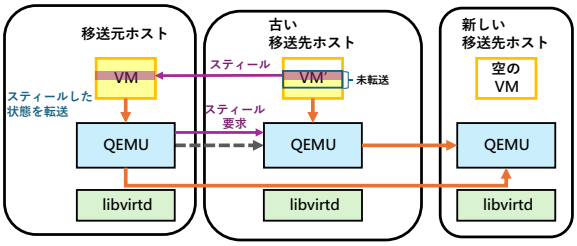


図 6: ワークスティールによる転送量調整

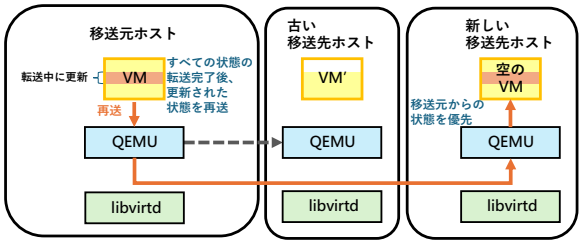


図 7: 重複受信したメモリデータの統合

が削除される．

マイグレーションの最終段階に入る前に，移送元の QEMU は図 6 のように古い移送先の QEMU との間で確立しておいたネットワーク接続を用いてワークスティール要求を送信する．古い移送先の QEMU から VM のメモリページの転送が完了していなければ，未転送のページの内の一部の情報を受け取る．移送元の QEMU は古い移送先の QEMU と同じか，より新しいメモリデータを保持しているため，メモリデータそのものを古い移送先から移送元へ転送する必要はない．移送元の QEMU は譲り受けたページをダーティビットマップに登録することにより，ページの再送の仕組みを用いて新しい移送先に転送する．この処理は古い移送先に未転送のページがなくなるまで繰り返し行う．

4.2.2 古い移送先ホストでの処理

Forward フェーズに古い移送先の QEMU が libvirtd から状態転送コマンドを受け取ると，移送元の QEMU との間のネットワーク接続を維持したまま，新しい移送先の QEMU との間のネットワーク接続を確立する．その後，移送元から受信済みのメモリデータを新しい移送先に転送する．メモリデータを受信済みのページを管理するために，古い移送先の QEMU はメモリデータの受信時に受信ビットマップへの記録を行う．メモリデータを転送する際には，この受信ビットマップのビットを先頭から順番に調べ，ビットが 1 の場合には対応するページのメモリデータを新しい移送先に転送する．

古い移送先の QEMU は移送元からワークスティール要求を受信すると，未転送のページがある場合にはそれを 2 つのグループに分割し，一方のグループに含まれるページの情報を移送元に返す．移送元に返したページは移送元の QEMU から転送されるため，古い移送先の QEMU はもう一方のグループに含まれるページだけを転送する．未転送のページは移送元と古い移送先の転送速度を基に，転送速度が速い方により多くのページが含まれるように分割する．これにより，未転送のページを 2 つのホストからほぼ同時に転送し終われるようにする．

4.2.3 新しい移送先ホストでの処理

Prepare フェーズに新しい移送先の QEMU が libvird から並列待受コマンドを受け取ると，VM の状態を 2 つのホストから並列に受信できるようにするために 2 つのスレッドを作成する．一方のスレッドは従来のマイグレーションと同様に移送元からのネットワーク接続を待ち受け，接続を確立した後，メモリデータを受信して VM のメモリに書き込む．メモリデータの受信が完了すると，VM のその他の状態も受信する．もう一方のスレッドは古い移送先からのネットワーク接続を待ち受け，接続を確立した後，メモリデータを受信して VM のメモリに書き込む．

メモリデータの重複受信を検出するために，移送先の QEMU は図 7 のように受信ビットマップを用いてページが受信済みかどうかを管理する．メモリデータを受信した時にそのページに対応するビットが 0 であれば 1 にセットし，VM のメモリに書き込む．古い移送先からメモリデータを受信した時に対応するビットが 1 であれば，すでに移送元から受信済みであることが分かる．この場合，移送元の VM が更新したメモリデータを受信したことを意味するため，そのメモリデータを優先し，古い移送先から受信したメモリデータは破棄する．一方，移送元からメモリデータを受信した時に対応するビットが 1 であれば，メモリが更新されたことによる再送であることが分かる．この場合，受信したメモリデータを VM のメモリに書き込む．

5. 実験

VM マイグレーションの実行中に DCmigrate を用いて移送先を変更し，その後のマイグレーションにかかる時間および，移送元におけるネットワークの平均帯域使用率を測定した．この実験では，ネットワークの混雑が発生した場合を想定して，移送元と古い移送先のネットワーク帯域を制限した．移送先を変更するタイミングを変えながら測定を行うために，古い移送先に一定のメモリデータを転送した時点でマイグレーションを一時停止し，移送先変更後に再開するようにした．また，データがすべて 0 のページを圧縮転送する最適化ができるだけ適用されないように，VM の起動時にすべてのページに 0 以外の値を書き込んだ．比較として，移送先変更時にマイグレーションをキャンセルして一から実行し直す従来手法を用いた場合と，移送先を変更せずにマイグレーションを継続した場合についても測定を行った．

この実験には表 1 の 3 台のホストを用いた．また．ホスト OS として Linux 5.15.0，管理サーバとして libvirt 8.9.0，

表 1: 実験に使用したホスト

項目	移送元ホスト	古い移送先ホスト	新しい移送先ホスト
CPU	Intel Xeon W-2245	Intel Core i7-10700	
メモリ	64GB	128GB	
NIC	10GbE		

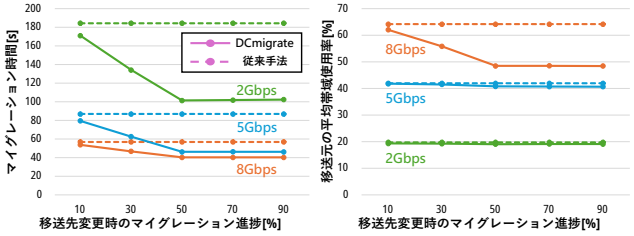


図 8: 同程度に混雑した場合のマイグレーション性能

仮想化ソフトウェアとして QEMU 7.2.0 を動作させた．マイグレーションを行う VM には仮想 CPU を 8 コア，メモリを 40GB 割り当て，ゲスト OS として Linux 5.15.0 を動作させた．

5.1 移送元と移送先が同程度に混雑した場合

マイグレーション中に移送元と移送先のネットワークが同程度に混雑した場合を想定して，移送元と古い移送先のネットワーク帯域を 8Gbps，5Gbps，2Gbps に制限した．新しい移送先のネットワーク帯域は 10Gbps のままとした．実験の結果を図 8 に示す．移送元と古い移送先のネットワーク帯域が 8Gbps に狭まった場合，従来手法と比べてマイグレーション時間を最大で 36%短縮することができた．これは古い移送先から並列に VM の状態を転送できたためである．移送先を変更する前のマイグレーションの進捗度が 50%を超えると，本来は古い移送先からの状態転送にかかる時間が支配的になり，マイグレーション時間は増加する．しかし，マイグレーション時間が増加しなかったことから，ワークスティールによる転送量の最適化が機能していることが確認できた．また，ネットワーク帯域が 5Gbps 以下に狭まった場合には，マイグレーション時間を最大で 47%短縮することができた．この場合には，移送元と古い移送先のネットワーク帯域の合計が新しい移送先の帯域以下になり，それぞれのホストのネットワーク帯域を活かし切れたためである．

マイグレーション中の移送元の平均帯域使用率はネットワーク帯域が 8Gbps に狭まった場合にのみ，従来手法よりも削減できることが分かった．これは，新しい移送先のネットワーク帯域がボトルネックとなり，移送元も古い移送先も 5Gbps 程度でしか状態を転送できなかったためである．

次に，ネットワークが混雑しても移送先を変更せず，そのままマイグレーションを継続した場合とも比較を行った．その際のマイグレーション時間を図 9 に示す．実験の結果，ネットワーク帯域が 5Gbps 以下に狭まった場合には，移送先変更前のマイグレーションの進捗度が 50%以下であればマイグレーション時間は変わらないことが分かった．これは，移送先を変更しても移送元が残りの状態を転送する時間は変わらず，その間に古い移送先が受信済みの状態を転送できたためである．一方，ネットワーク帯域が 8Gbps に狭まった場合は移送先を変更しない方が常にマイグレーション時間は短くなった．この場合は，古い移送先と並列に転送を行うことにより，移送元からは 5Gbps でしか転送できなかったためである．ただし，移送先を変更しなければマイグレーション後の VM のネットワーク性能は悪くなる．

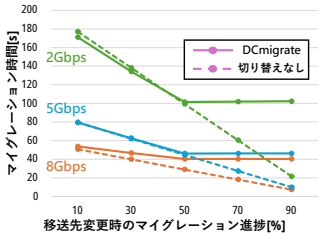


図 9: 同程度に混雑した場合の移送先変更なしとの比較

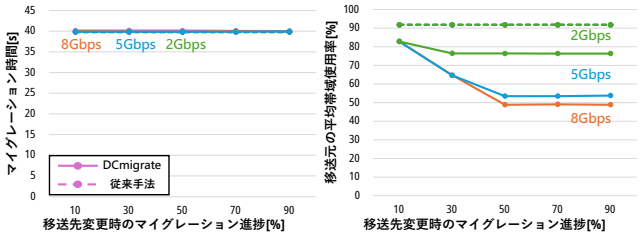


図 10: 移送先のみが混雑した場合のマイグレーション性能

5.2 移送先のみが混雑した場合

マイグレーション中に移送先のネットワークのみが混雑した場合を想定して，古い移送先のネットワーク帯域を 8Gbps，5Gbps，2Gbps に制限した．移送元と新しい移送先のネットワーク帯域は 10Gbps のままとした．実験の結果を図 10 に示す．古い移送先のネットワーク帯域によらず，従来手法と比べてマイグレーション時間を短縮することはできなかった．これは，移送元と新しい移送先のネットワーク帯域が同じ 10Gbps であったため，古い移送先から並列に VM の状態を転送しても高速化できなかったためである．ただし，この場合，DCmigrate は並列転送やワークスティールを行っているが，それらのオーバヘッドによる性能低下はほぼないことが分かった．

一方，マイグレーション中の移送元の平均帯域使用率は削減できることが分かった．古い移送先のネットワーク帯域が広いほど，この削減幅は大きくなった．これは，古い

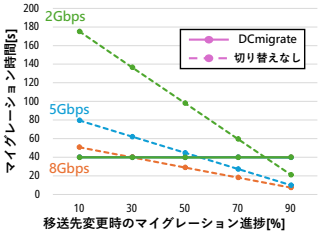


図 11: 移送先のみが混雑した場合の移送先変更なしとの比較

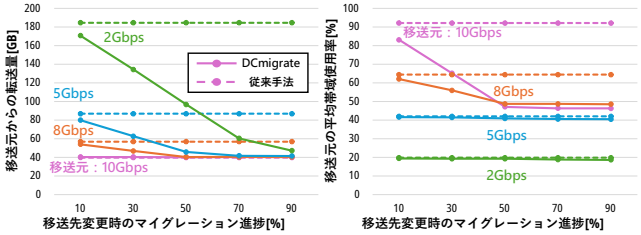


図 12: より適した移送先が見つかった場合のマイグレーション性能

移送先から転送できる VM の状態が増加すると，移送元から転送を行う必要のある状態をより削減できたためである．

移送先変更を行わずにマイグレーションを継続した場合の実験の結果を図 11 に示す．移送先を変更するタイミングが早ければマイグレーション時間を短縮できることが分かった．また，古い移送先のネットワーク帯域が狭くなるほど，移送先変更のタイミングが遅くなってもマイグレーション時間を短縮できることが分かった．これは，移送先を変更しない場合のマイグレーションにより長い時間がかかるためである．

5.3 より適した移送先ホストが見つかった場合

ネットワークは混雑していないものの，負荷が高くなった移送先ホストをより適したホストに変更する場合を想定して，ネットワーク帯域を制限せずに移送先変更後のマイグレーション性能を測定した．その際に，移送元のネットワークが混雑していた場合を想定して，移送元のネットワーク帯域を 8Gbps，5Gbps，2Gbps に制限した場合についても測定した．古い移送先と新しい移送先のネットワーク帯域は 10Gbps のままとした．

実験結果は図 12 のようになった．移送元のネットワーク帯域が 10Gbps の場合には，従来手法と比べてマイグレーション時間は短縮できなかった．これは，新しい移送先のネットワーク帯域がボトルネックとなり，移送元と古い移送先を合わせて 10Gbps までしか転送できなかったためである．一方，移送元のネットワーク帯域が狭い場合には，マイグレーション時間を最大で 74%削減することができた．移送元のネットワーク帯域が狭くなるほど，マイグレーション時間がより短縮できた．これは，ネットワーク帯域が 10Gbps の古い移送先から VM の状態を高速に転送することができたためである．

一方，移送元のネットワーク帯域が 8Gbps 以上の場合には平均帯域使用率を削減できることが分かった．これは，古い移送先から 10Gbps で状態を転送することにより，移送元で利用可能なネットワーク帯域が 5Gbps 程度になったためである。

移送先を変更せずにマイグレーションを継続した場合の実験結果を図 13 に示す．移送元のネットワーク帯域が，

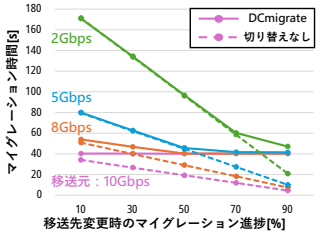


図 13: より適した移送先が見つかった場合の移送先変更なしとの比較

5Gbps 以下の場合にはタイミング次第でマイグレーション時間の増加なしで移送先を変更できることが分かった．これは，移送元からの状態の転送中に古い移送先からの転送を完了できるためである．一方で，移送元が 8Gbps 以上の場合には常にマイグレーション時間は長くなることが分かった．この場合，古い移送先から状態を転送することで，移送元からは 5Gbps 程度でしか転送できなくなったためである．

5.4 移送元と移送先の混雑度が異なる場合

移送元と古い移送先の両方のネットワークが混雑しており，それぞれの混雑度が異なる場合を想定して，ネットワーク帯域を 8Gbps，5Gbps，2Gbps のいずれかに制限した．移送元のネットワーク帯域を 8Gbps，5Gbps，2Gbps にした場合の結果をそれぞれ図 14，15，16 に示す．両方のホストのネットワーク帯域を制限した場合には，従来手法と比べて常にマイグレーション時間を短縮できることが分かった．移送元のネットワーク帯域がより狭く，古い移送先のネットワーク帯域がより広い場合にマイグレーション時間をより短縮できた．DCmigrate は移送先変更前に転送した状態を古い移送先から転送するため，古い移送先の帯域が相対的に広い場合，移送元からすべての状態を転送し直す従来手法よりも高速になるためである．

平均帯域使用率については，移送元のネットワーク帯域を 8Gbps，移送先のネットワーク帯域を 5Gbps に制限した場合のみ，削減できることが分かった．これは，移送元と古い移送先のネットワーク帯域の合計が 10Gbps 以上になり，かつ，移送元のネットワーク帯域が 5Gbps 以上の場合にのみ，移送元が利用する帯域が狭まるためである．

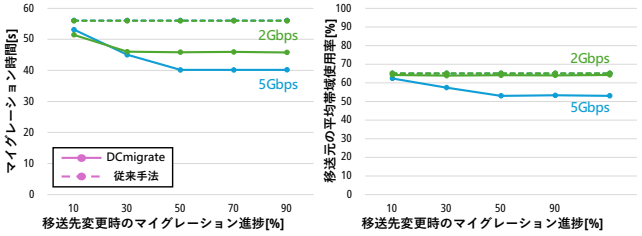


図 14: 混雑度が異なる場合のマイグレーション性能 (移送元が 8Gbps の場合)

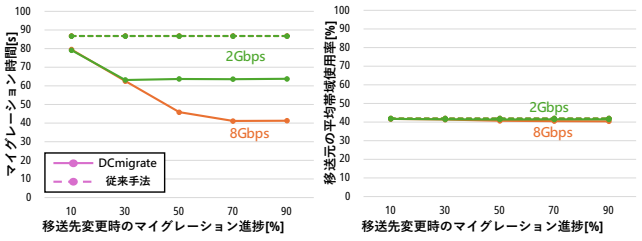


図 15: 混雑度が異なる場合のマイグレーション性能 (移送元が 5Gbps の場合)

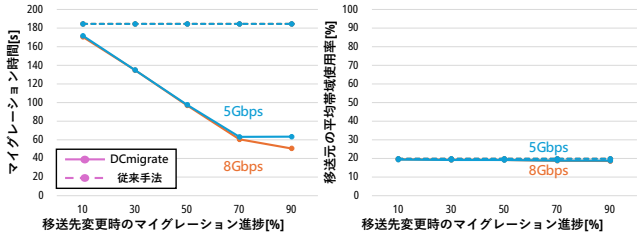
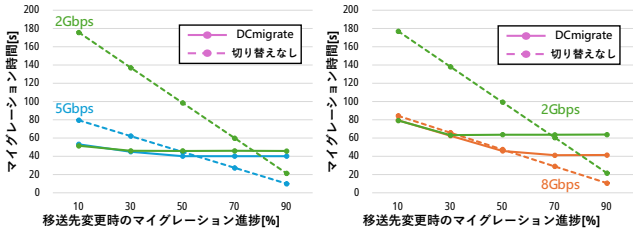


図 16: 混雑度が異なる場合のマイグレーション性能 (移送元が 2Gbps の場合)

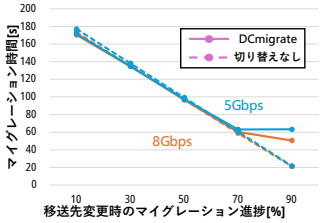
移送先変更を行わず，マイグレーションを継続した場合の実験結果を図 17 に示す．古い移送先のネットワーク帯域が移送元より狭い場合にのみ，移送先を変更するタイミング次第でマイグレーションを高速化できることが分かった．この場合，古い移送先のネットワーク帯域がボトルネックとなるため，十分な帯域をもつ移送先に変更することでマイグレーションを高速化できる．逆に，移送元のネットワーク帯域の方が狭い場合には，移送先変更のタイミングが早ければ移送先を変更しない場合とマイグレーション時間は変わらないことが分かった．これは，古い移送先からの転送時間が移送元からの転送時間より短くなるためである．

6. 関連研究

Acinonyx [6] は VM マイグレーション中にネットワーク経路を切り替えることでマイグレーション時間を短縮することができる．そのために，複数のネットワーク経路を持つデータセンタにおいて，ホスト間で発生する大量のデータフローを Software Defined Networking (SDN) を用いて動的にスケジューリングする．しかし，移送先ホストを変



(a) 移送元が 8Gbps の場合　(b) 移送元が 5Gbps の場合



(c) 移送元が 2Gbps の場合

図 17: 混雑度が異なる場合の移送先変更なしとの比較

更することはできないため，移送先ホストの負荷が高い場合や混雑したネットワークを経由しないと移送先ホストに到達できない場合には改善が見込めない．DCmigrate と併用することにより，移送先ホストの変更に加えてネットワーク経路の明示的な変更も行えるとより効果的である．

VM マイグレーションの途中で状況が変わった場合，分割マイグレーション [8] に切り替えることができれば，残りの VM の状態を転送するホストを変更することができる．分割マイグレーションは VM のメモリを分割して複数の移送先ホストに転送する手法である．これにより，より適したホストに VM の状態を転送することが可能になる．しかし，分割マイグレーション後はホスト間でリモートページングを行ってメモリデータを転送する必要があるため，VM の性能が低下する．また，最初から分割マイグレーションを行う場合とは異なり，アクセスされることが予測されるメモリを VM が実行されるメインホストに転送することができないため，さらに VM の性能が低下する可能性が高い．

そこで，これら 2 つの移送先ホストから 1 つのホストに VM の状態を転送する統合マイグレーション [5] を行うことも考えられる．これにより，リモートページングを行わずに VM を実行できるようになり，VM の性能を回復させることができる．DCmigrate は分割マイグレーションへの切り替えと統合マイグレーションを同時に行う手法と考えることもできる．ただし，DCmigrate ではマイグレーションが完了するまで VM は移送元ホストで動作し続けるため，マイグレーション中にリモートページングによって VM の性能が低下することはない．

プレコピーマイグレーションをキャンセルした時には移送元ホストの VM が実行を継続することができるため，別の移送先ホストを選択してマイグレーションをやり直すこ

とができる．一方，ポストコピーマイグレーションをキャンセルすると VM の状態が移送元ホストと移送先ホストにまたがっているため，VM の実行を継続できなくなる．キャンセラブルなポストコピー VM 移送 [9] では，移送先ホストで稼働中の VM が更新したメモリを移送元ホストに転送してメモリの同期をとる．これにより，マイグレーションをキャンセルした時に移送元ホストで VM の実行を継続することができる．DCmigrate をポストコピーマイグレーションに適用すれば，移送先を変更する際にマイグレーションをキャンセルできるようにするために VM のメモリを同期する必要はなくなる．

Scatter-Gather ライブマイグレーション [4] は，移送元ホストから複数の中間ホストに VM のメモリデータを転送し，並行してそれを移送先ホストに転送するマイグレーション手法である．中間ホストを用いることにより，移送先のネットワークの混雑状況やメモリの利用状況に依存せずにマイグレーションを開始することができる．移送元からそれぞれの中間ホストへの転送量をネットワークの混雑状況に応じて変化させることで転送の最適化も可能である．複数の中間ホストから移送先ホストへ VM のメモリデータを転送する点は，DCmigrate の移送先変更後のマイグレーションと類似している．しかし，Scatter-Gather マイグレーションはポストコピーマイグレーションと同様に，CPU の実行状態だけを移送先ホストに転送して VM の動作を先に開始するため，マイグレーション中に移送先を変更することはできない．

Agile ライブマイグレーション [3] は，VM の使われていないページを事前にネットワーク上のスワップデバイスにページアウトさせておき，マイグレーション実行時には使われているページのみを転送する．これにより，マイグレーション中の VM の性能を保ったまま，マイグレーション時間を削減することができる．スワップデバイスにページアウトする処理が完了するまでは移送先を変更可能であるため，ページアウト中に状況が変化した場合にも対応することができる．また，ネットワーク帯域を考慮してスワップデバイスにページアウトする量を調整することで，トータルのマイグレーション時間を最適化することも可能である．しかし，マイグレーション後にスワップデバイス上のページが必要になった場合には VM の性能が低下する．

7. まとめ

本稿では，VM マイグレーションをキャンセルせずに移送先ホストを柔軟に変更できるシステム DCmigrate を提案した．DCmigrate は移送元ホストで移送先を新しいホストに切り替え，未転送の VM の状態のみを新しい移送先ホストに転送する．それに加えて，古い移送先ホストは移送元から受信済みの VM の状態を新しい移送先に転送する．2 つのホストから VM の状態を並列に転送し，それぞれのホストからの転送量を最適化することにより，移送先変更後のマイグレーション時間を短縮する．実験の結果，ネットワークが混雑した場合に，従来手法と比較してマイグレーション時間を短縮できることを確認した．

今後の課題は，従来のマイグレーションにおいて適用されている VM の状態の圧縮転送等の最適化を古い移送先からの転送にも適用することである．また，ワークスティール後に移送元の転送速度が低下しても，古い移送先からの再スティールは行われない．そのため，状況の変化により柔軟に対応できるワークスティール処理を実装する必要がある．

謝辞 本研究の一部は，JST，CREST，JPMJCR21M4 の支援を受けたものである．また，本研究成果の一部は，国立研究開発法人情報通信研究機構（NICT）の委託研究（JPJ012368C05501）により得られたものである．

参考文献

[1] Amazon Web Services, Inc. Amazon EC2 High Memory Instances. https://aws.amazon.com/ec2/instance-types/u7i/.
[2] F. Bellard. Qemu. https://www.qemu.org/.
[3] U. Deshpande, D. Chan, T.-Y. Guh, J. Edouard, K. Gopalan, and N. Bila. Agile live migration of virtual machines. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 1061–1070, 2016.
[4] U. Deshpande, Y. You, D. Chan, N. Bila, and K. Gopalan. Fast server deprovisioning through scatter-gather live migration of virtual machines. pp. 376–383, 2014.
[5] T. Kashiwagi and K. Kourai. Flexible and Efficient Partial Migration of Split-memory VMs. pp. 248–257, 2020.
[6] A. Nadjaran Toosi and R. Buyya. Acinonyx: Dynamic Flow Scheduling for Virtual Machine Migration in SDN-Enabled Clouds. pp. 886–894, 2018.
[7] Red Hat, Inc. libvirt: The virtualization API. https://libvirt.org/.
[8] M. Suetake, T. Kashiwagi, H. Kizu, and K. Kourai. S-memV: Split Migration of Large-Memory Virtual Machines in IaaS Clouds. pp. 285–293, 2018.
[9] 小川遥加, 山田浩史, 十文字優斗, 阿部芳久. キャンセラブルなポストコピー VM 移送. pp. 1–12, 2017.