複数ホストにまたがるVMのメモリ使用状況に着目した高速化

田内　聡一朗[1]　光来　健一[1]

概要：近年，クラウドサービスの一つとして，ユーザに仮想マシン（VM）を提供するIaaS型クラウドが普及している．それに伴い，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．このようなVMのマイグレーションを容易にするために，VMのメモリを複数の小さなホストに分割して転送する分割マイグレーションが提案されている．分割マイグレーション後にはリモートページングを行ってVMが必要とするメモリをホスト間で転送する．しかし，従来のリモートページングでは必要とされたメモリの中に使用中のデータがなかったとしても転送を行う必要があった．本稿では，未使用メモリに関連するオーバヘッドを削減することで複数ホストにまたがるVMの高速化を実現するシステム *FCtrans* を提案する．FCtransはVMの起動時から未使用メモリを追跡し，分割マイグレーション後も追跡を続ける．この情報を用いて，マイグレーション時には移送先ホストに未使用メモリを転送しないようにする．そして，分割マイグレーション後には未使用メモリに対してリモートページングを行わないようにし，未使用メモリにアクセスしたVMの実行を即座に再開する．OSが解放したメモリページも未使用メモリとして扱えるように，VMの外からOSのページ管理情報を取得する．FCtransをKVMに実装し，従来手法と性能を比較する実験を行った．

1. はじめに

近年，クラウドサービスの一つとして，ユーザに仮想マシン (VM) を提供するIaaS型クラウドが普及している．それに伴い，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おり，ビッグデータの解析などに利用されている．ホストのメンテナンス等の際には，VMを停止させることなく別のホストへマイグレーションすることで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VMのOSの情報やメモリなどの状態をネットワーク経由で転送し，移送先ホストでVMを再開する．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が有する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の場合には，十分な空きメモリを持つホストを常に確保しておくのはコストの面での負担が大きい．

そこで，VMのメモリを複数の小さなホストに分割して転送する分割マイグレーション [1] が提案されている．分割マイグレーションでは，仮想CPUや仮想デバイスの情報と可能な限りのメモリデータをメインホストに転送し，メインホストへ転送し切れないメモリデータをサブホストへ転送する．マイグレーション後にサブホスト上に存在するメモリデータが要求されると，VMはサブホストからメインホストへ要求されたデータを転送（ページイン）し，代わりに不要なメモリデータをサブホストに転送（ページアウト）することでメインホストのメモリ容量を確保する．この処理はリモートページングと呼ばれる．リモートページングの問題点は，VMがサブホストにあるメモリにアクセスすると必ずページインが行われてVMの性能が低下することである．その一方で，VMのメモリの中には使われていない領域が存在することも多い．例えば，VM内でOSが起動した直後にはVMのメモリ領域の多くは未使用である．しかし，従来のリモートページングは未使用メモリに対してもサブホストからのページインやサブホストへのページアウトを行う．同様に，従来の分割マイグレーションは未使用メモリであっても移送先ホストに転送を行う．

本稿では，未使用メモリに関連するオーバヘッドを削減することで複数ホストにまたがるVMの高速化を実現するシステム *FCtrans* を提案する．FCtransはVMの起動時から未使用メモリを追跡し，分割マイグレーション後も追跡を続ける．追跡した未使用メモリの情報を用いて，マイグレーション時には移送先ホストに未使用メモリを転送しないようにする．そして，分割マイグレーション後にリモートページングを必要とした際には，未使用メモリに対

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

してネットワーク転送を行わないようにする．その結果，VM がサブホストにある未使用メモリを必要とした時には，即座に VM の実行を再開させることができる．また，未使用メモリのデータを転送しないことにより，分割マイグレーションを高速化することもできる．

FCtrans を QEMU-KVM に実装し，リモートページングの高速化を実現した．Linux の userfaultfd 機構を用いることで，VM 起動直後から VM のメモリ使用状況を使用ビットマップで管理する．OS が解放したメモリページも未使用メモリとして扱えるように，LLView[2] を用いて VM の外から OS のページ管理情報を取得する．マイグレーション後に VM がメインホストに存在しないメモリにアクセスした時には，未使用メモリであればページインを行わずに直接物理メモリを割り当てる．メインホストに空きメモリがある場合にはページアウトも行わない．FCtrans を用いて実験を行い，分割マイグレーション後のページイン回数および，マイグレーション時間とダウンタイムが大幅に削減できることがわかった．

以下，2 章では大容量メモリを持つ VM のマイグレーションが抱える問題点について述べる．2 章では複数ホストにまたがる VM の高速化を実現するシステム FCtrans を提案し，4 章でその実装について述べる．5 章では，FCtrans の性能を調べるために行った実験の結果を述べる．6 章で関連研究に触れ，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 マイグレーションは，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用いることで，VM 上で稼働しているサービスの提供を中止す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まず，移送先ホストに VM を作成して，その後，移送元ホストの VM のメモリデータをネットワーク経由で移送先ホストへ転送していく．転送中に更新された VM のメモリは移送先ホストへ再送され，移送元ホスト上と移送先ホスト上で VM のメモリの差分が十分に小さくなったら移送元ホストの VM を停止させる．そして，VM の更新されたメモリのデータと仮想 CPU や仮想デバイスの状態を転送し，移送先ホストで VM の実行を再開する．

近年，大容量メモリを持つ VM が利用されるようになってきている．例えば，Amazon EC2 では 12TB のメモリを持つ VM が提供されており，ビッグデータの解析などに利用されている．VM はホストのメンテナンス等の際に別のホストへマイグレーションされるが，マイグレーションの要件として，移送先ホストは VM のメモリサイズよりも大きな空きメモリ容量を確保しなければならない．そのため，大容量メモリを持つ VM のマイグレーションはより困難になる．メンテナンス時や災害などの緊急時のために十分な空きメモリ容量を持ったホストを常に確保しておくのはコストの面での負担が大きいためである．移送先ホストとして適切なホストが存在しない場合には，VM のマイグレーションを行うことができないため，ホストのメンテナンスの間，ユーザは VM が提供するサービスを利用することができなくなってしまう．

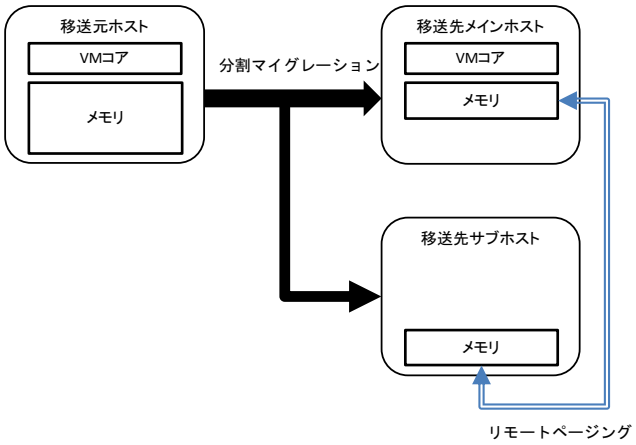

図 1　分割マイグレーション

そこで，図 1 のように大容量メモリを持つ VM を複数のホストに分割して転送する分割マイグレーション [1] が提案されている．マイグレーション後に仮想 CPU や仮想デバイスからなる VM コアを動作させるホストはメインホスト，メインホスト以外のホストはサブホストと呼ばれる．分割マイグレーションは仮想 CPU や仮想デバイスの状態などの VM コアと今後アクセスが予測されるメモリデータを優先してメインホストへ転送し，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群へ転送する．マイグレーション後にアクセスが予測されるメモリは LRU に基づいて決定する．メモリを転送する際には移送元ホストとそれぞれの移送先ホストが直接通信を行うことで，効率よくマイグレーションを行うことができる．

分割マイグレーション後には，VM は複数のホストにまたがって動作し，VM コアは移送先メインホストで実行される．そのため，VM はメインホスト上のメモリに直接アクセスすることができるが，サブホスト上のメモリにはリモートページングを行ってアクセスする．VM がサブホストに存在するメモリを必要とした際には，そのメモリをメインホストにページインする．同時に，メインホスト上の今後アクセスされる可能性が最も低いと予測されるメモリ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アクセスされる可能性が高いメモリが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リモートページングの頻度は低い．

リモートページングの問題点は，マイグレーションから時間がたったり，VM のワークロードが変化したりして

ページインとページアウトが頻繁に行われるようになるとVM の性能が低下することである．VM がサブホストにあるメモリにアクセスするとページインが完了するまで VM の仮想 CPU は停止される．ページアウトは仮想 CPU の再開後に行うことができるが，VM やネットワークの性能に影響を及ぼす．

一方で，VM のメモリの中には使われていない領域が存在することも多い．例えば，VM 内で OS が起動した直後には VM のメモリ領域の多くは未使用である．一度使用したメモリ領域であってもアプリケーションの終了などにともなって OS が解放すると未使用状態となる．従来のリモートページングはサブホスト上の未使用メモリに対して VM がアクセスを行った場合でもサブホストからそのメモリデータを転送してページインを行う．そして，メインホスト上に別の未使用メモリがあれば，アクセスされそうにないメモリとみなしてページアウトを行い，サブホストにそのメモリデータを転送する．このように，未使用のメモリや有用なデータが格納されていないメモリについても，従来のリモートページングでは不要なメモリ転送が行われる．同様に，従来の分割マイグレーションは未使用メモリであっても移送先ホストに不要な転送を行う．

3. FCtrans

本稿では，VM の未使用メモリに着目し，複数ホストにまたがる VM の高速化を実現する FCtrans を提案する．FCtrans は図 2 のように VM の起動時から未使用メモリを追跡し，マイグレーション後も追跡を継続する．VM の作成時にはすべてのメモリページが未使用であり，VM の起動時にメモリにデータが書き込まれるとそのページは使用中となる．分割マイグレーション後には，移送先メインホストにおいて VM のメモリの使用状況を管理する．分割マイグレーションやページアウトによってサブホストに転送されたメモリについても，メインホストで管理を行う．

追跡した情報を用いて，FCtrans は分割マイグレーション時に移送先ホストに未使用メモリを転送しないようにする．そして，分割マイグレーション後にリモートページングが必要になった際には，FCtrans は未使用メモリに対して不要なネットワーク転送を行わないようにする．VM がサブホストにある未使用メモリにアクセスした場合にはメモリデータのネットワーク転送は行わず，メインホストで空いているメモリにアクセスさせる．その結果，VM がネットワーク経由でのページインの完了を持つ必要がなく，即座に VM を再開されられるようになる．また，メインホストにある未使用メモリは LRU に基づいてページアウトの対象となることが多いが，FCtrans は未使用メモリの場合にはメモリデータのネットワーク転送を行わない．

分割マイグレーション時に移送元ホストから移送先ホストへ未使用メモリを転送しないことにより，ネットワーク

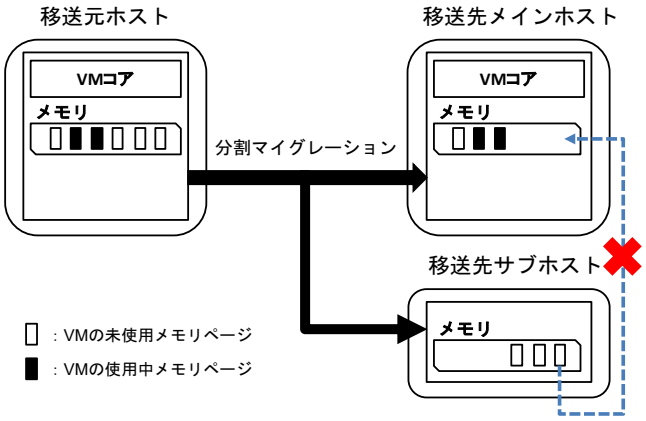

図 2　FCtrans

転送量を削減し，マイグレーションを高速化することもできる．分割マイグレーションではアクセス履歴を用いてメモリの転送先を決定するため，未使用メモリはサブホストに転送されることが多い．そのため，サブホストへのメモリ転送が大きく削減される．この手法は分割マイグレーションだけではなく，通常の 1 対 1 マイグレーションや部分マイグレーション [5] にも適用可能である．

FCtrans は VM が一度も使っていないメモリだけでなく，VM 内の OS が使わなくなったメモリも未使用メモリとして扱う．OS がメモリ領域を確保して一度でもアクセスすると，そのメモリを解放して使わなくなったとしても VM のメモリは使用中のままになる．そこで，FCtrans は OS のメモリ使用状況を監視し，OS が使わなくなったメモリがあれば対応する VM のページを未使用状態に戻す．そのために，LLView [2] を用いて VM の外から OS の情報を取得する．LLView は開発者が容易に VM 内の OS の情報を取得することを可能にするフレームワークである．

4. 実装

分割マイグレーションとリモートページングを QEMU-KVM 2.11.2 に移植し，それに対して FCtrans を実装した．

4.1 VM の使用メモリの追跡

FCtrans はビットマップを用いて，VM のメモリを 4KB のページ単位で使用中であるか未使用であるかを管理する．このビットマップは使用ビットマップと呼ばれる．使用ビットマップは VM に割り当てられているメモリページ数分のビットからなり，図 3 のようにページが使用中であれば対応するビットが 1，未使用であれば 0 となる．全体のメモリ使用状況を管理するために，使用ビットマップは VM 起動前に作成する．VM の起動時にはすべてのページに対応するビットが 0 であり，VM がメモリにアクセスするとそのページに対応するビットは 1 となる．

VM による未使用メモリへのアクセスを検出するために，FCtrans は Linux の userfaultfd 機構を用いる．VM を

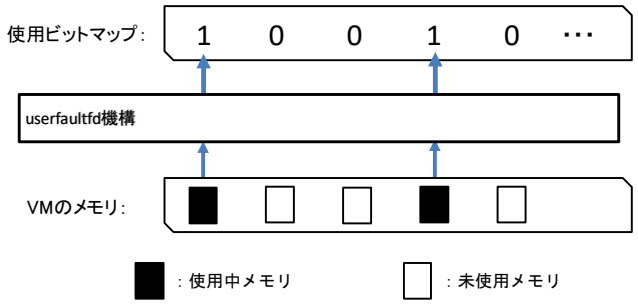

図 3　VM の未使用メモリの管理

作成する際に QEMU-KVM は物理メモリが割り当てられていないメモリ領域を VM のメモリとして確保するため，FCtrans はその領域を userfaultfd 機構に登録する．これにより，VM が未使用メモリに初めてアクセスした時にページフォールトを発生させて QEMU-KVM で処理を行うことができる．ページフォールトの発生時には，FCtrans は userfaultfd 機構を用いて当該ページに物理メモリを割り当て，使用ビットマップの対応するビットを 1 にする．この処理をページ単位で行うとオーバヘッドが大きくなるため，FCtrans はアクセスされたページを含む複数のページ（チャンク）に一括で物理メモリを割り当てる．

4.2　未使用メモリを考慮した分割マイグレーション

FCtrans は分割マイグレーションの際に図 4 のように未使用メモリの転送を行わない．移送先ホストにページ単位でメモリを転送する際に使用ビットマップを調べ，対応するビットが 1 の時，つまり，メモリページが使用中の時にだけメモリデータの転送を行う．ビットが 0 の時にはメモリページが未使用なので送信処理を行わない．移送先サブホストにメモリデータを転送する際には移送先メインホストにそのメモリ情報を転送するが，未使用メモリの場合にはメモリ情報も転送しない．これにより，未使用メモリのデータ転送および，移送元ホストにおける VM のメモリからのデータの読み出し，移送先ホストにおける VM のメモリへのデータの書き込みのオーバヘッドが削減される．

移送先メインホストでは受信したメモリデータを userfaultfd 機構を用いて VM のメモリに書き込む．移送先メインホストで VM を作成する際には移送元ホストと同様に VM のメモリが userfaultfd 機構に登録される．そのため，従来のように VM のメモリに直接書き込むとページフォールトが発生して QEMU-KVM において処理が行われるため，オーバヘッドが大きくなる．そこで，userfaultfd 機構を用いて VM のメモリに間接的にデータを書き込む．userfaultfd 機構は VM に物理メモリを割り当ててからそのメモリに指定されたデータを書き込む．一旦，書き込みを行ったページに対してデータが再送された場合にはページフォールトを発生させることなくデータを直接書き込むことができる．

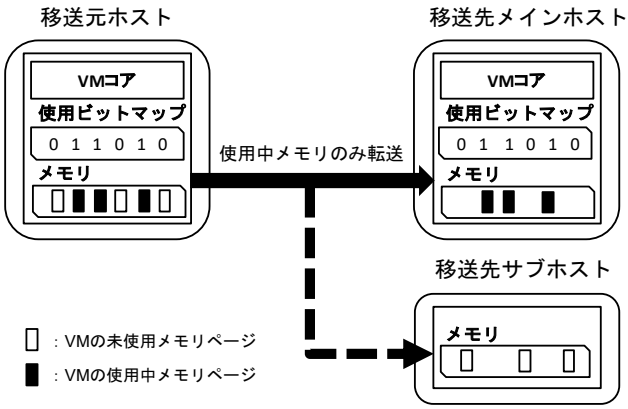

図 4　分割マイグレーションの最適化

移送先メインホストでは受信したメモリ情報に基づいて使用ビットマップの再構築を行う．この再構築は使用ビットマップの情報を直接転送することなく行うことができる．マイグレーション開始時には使用ビットマップのすべてのビットを 0 にしておく．そして，メインホストに格納されるメモリデータまたはサブホストに格納されるメモリの情報を受信した際には，使用ビットマップの対応するビットを 1 にする．これにより，移送元ホストで未使用だったページは移送先ホストでも未使用のままとなる．

4.3　リモートページングの最適化

分割マイグレーション後に，VM がメインホストに存在しないメモリにアクセスするとページフォールトが発生する．このようなメモリはサブホストに存在するメモリまたは未使用メモリのいずれかである．そこで，FCtrans は使用ビットマップを調べて，VM がアクセスしたページに対応するビットが 1，つまり，ページが使用中であれば，従来通りにサブホストからページインを行う．一方，ビットが 0 であり，当該ページが未使用であった場合には，サブホストからのページインは行わない．その代わりに，図 5 のようにメインホストにおいて userfaultfd 機構を用いて空き物理メモリを VM に割り当てる．リモートページングではチャンク単位でページインを行うため，この物理メモリの割り当てもチャンク単位で行う．これにより，未使用メモリのページインに伴うオーバヘッドを削減し，VM の性能を向上させることができる．

FCtrans はページアウトを削減するために，メインホストにおいて VM が利用可能な空きメモリ量の管理を行う．従来のリモートページングではページインとページアウトを対で実行してメインホストの使用物理メモリ量を一定に保っていたが，FCtrans ではページインが行われない場合があるためである．未使用ページにアクセスした際の VM への物理メモリの割り当てにより，メインホストにおいて利用可能な空きメモリがなくなった場合には，従来通りにサブホストへのページアウトを行って空きメモリを確保す

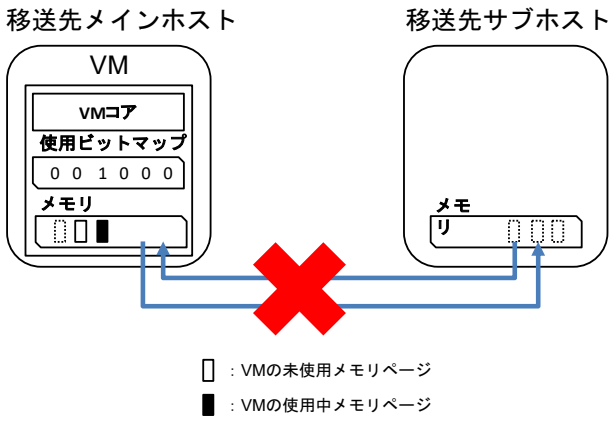

図 5　未使用メモリに対するリモートページング

```
for (pfn = 0; pfn < max_pfn; pfn++) {
  page = pfn_to_page(pfn);
  count = atomic_read(&page->_refcount);
}
```

図 6　メモリページの使用状況を調べるプログラム

る．空きメモリがある場合にはページアウトも行わないため，さらに VM の性能を向上させることができる．

4.4　OS が解放したメモリの反映

VM 内の OS が使わなくなったメモリを未使用メモリとして扱えるように，FCtrans は定期的に VM 内の OS が管理しているページ情報を取得する．取得したページの参照カウントが 0 であれば VM のページを未使用状態に戻す．具体的には，図 7 のように拡張した userfaultfd 機構を用いて VM に割り当てられた物理メモリを解放し，使用ビットマップの対応するビットを 0 にする．その間に同時に OS がメモリの再利用を始めると FCtrans の処理との競合が発生するため，この処理を行っている間は VM のすべての仮想 CPU を一時停止する. 未使用状態に戻されたページに再びアクセスされるとページフォールトが発生し，最初のアクセスと同様に処理される．

VM の外から透過的に OS のページ情報を取得するために，FCtrans は LLView[2] を用いてページの参照カウントを取得する．このプログラムは Linux のヘッダファイルを用いて図 6 のように記述される．すべてのページ番号に対してページ構造体を取得し，その参照カウントを調べる．このプログラムを LLView を用いてコンパイルすると，生成されたビットコードが OS データを取得しようとした時に VM 内のメモリにアクセスするように変換される．

5.　実験

FCtrans を用いてリモートページングと分割マイグレーションの性能向上について調べる実験を行った．また，未使用ページを追跡するためのオーバヘッドを調べた．比較

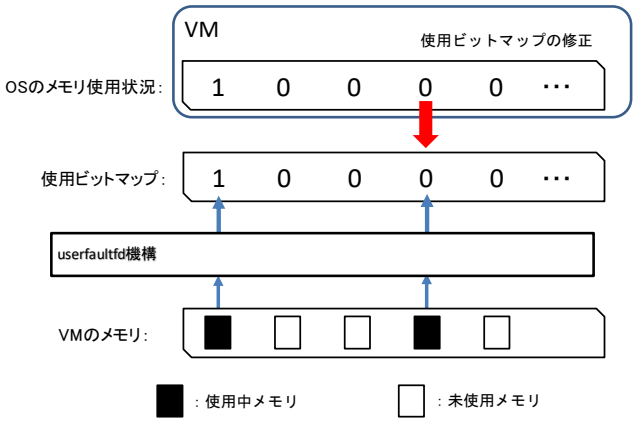

図 7　VM レベルと OS レベルのメモリ情報の統合

として，従来の分割マイグレーションとリモートページングを行うシステムを用いた．移送元ホストには Intel Core i7-7700 の CPU，16GB のメモリを持つマシンを用い，移送先ホストには Intel Xeon E3-1225 v5 の CPU，16GB のメモリを持つマシンを用いた．移送先サブホストは実験によって移送元ホストまたは移送先メインホストと同一とした．これらのマシンの OS として Linux 4.3.0 を動作させ，仮想化ソフトウェアとして QEMU-KVM 2.11.2 を動作させた．また，VM には 1 個の仮想 CPU，1GB のメモリを割り当て，Linux 4.4.0 を動作させた．分割マイグレーション時には 512MB ずつにメモリを分割した．

5.1　分割マイグレーション後のページイン回数

分割マイグレーション後にリモートページングが抑制できているかどうかを調べるためにページインの回数を計測した．VM の中で何も処理を行わない場合と 5 秒ごとに 16MB のメモリを確保して書き込みを行うプログラムを実行した場合について，FCtrans と従来システムと比較を行った．この実験では，移送先メインホストと移送先サブホストが異なるホストになるようにし，サブホストにおいて 5 秒ごとにページインの回数を測定した．

VM 内で何も処理を行わなかった場合の累積ページイン回数を図 8 に示す．5 秒ごとの最大ページイン回数は従来システムで 896 回，FCtrans では 4 回であった．また，1 秒当たりの平均ページイン回数は従来システムで 22 回，FCtrans では 0.03 回であった．実験結果より，FCtrans では未使用メモリへのアクセス時にメインホスト内で物理メモリを割り当てることができたため，従来システムと比較して大幅にページイン回数を削減できることが分かった．

一方，VM 内で 5 秒毎に 16MB のメモリに書き込むプログラムを動作させて負荷をかけた場合の累積ページイン回数を図 9 に示す．5 秒ごとの最大ページイン回数は従来システムで 648 回，FCtrans では 52 回であった．また，1 秒当たりの平均ページイン回数は従来システムで 6.4 回，FCtrans では 0.6 回であった．この場合も従来システムと

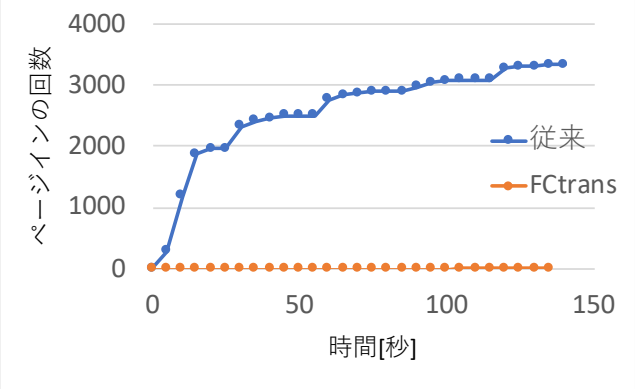

図 8　ページイン回数（負荷なし）

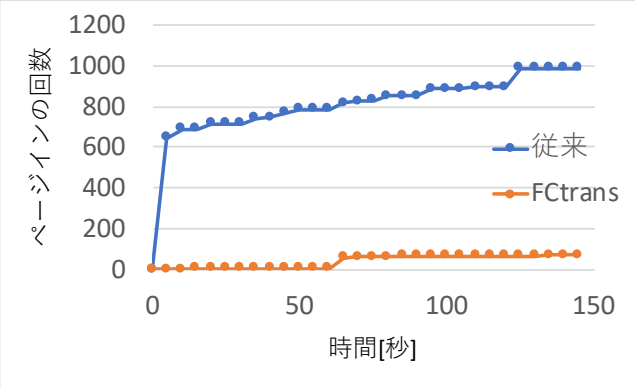

図 9　ページインの回数（負荷時）

比較して FCtrans では大幅なページイン回数の削減を行うことができた．

5.2　マイグレーション性能

FCtrans による分割マイグレーションの性能向上について調べるために，マイグレーション時間を測定した．この実験では移送先メインホストと移送先サブホストを同一とすることで，すべてのメモリデータの転送がネットワークを経由するようにした．実験結果を図 10 に示す．従来の分割マイグレーションと比較して，FCtrans はマイグレーション時間を 71 %短縮できることが分かった．

マイグレーション中に移送元ホストから移送先ホストに転送されたデータ量を図 11 に示す．従来の分割マイグレーションはマイグレーション時にすべてのメモリデータを転送するため，1GB 以上のネットワーク転送が行われた．一方，FCtrans は未使用メモリの転送を行わないため，ネットワーク転送量を 314MB に削減することができた．この削減率と同程度にマイグレーション時間が削減された．

次に，マイグレーション中の VM のダウンタイムを測定した．KVM では，残りのメモリデータを 300 ミリ秒で転送可能と推定された時に VM を停止し，メモリデータと仮想 CPU や仮想デバイスの状態を転送する．実験結果を図 12 に示す．FCtrans では VM の停止後に転送しようとするメモリの中に未使用メモリが含まれたため，実際のメモリ転送時間は 300 ミリ秒より大幅に短くなった．その結果，ダウンタイムは 96 ミリ秒となった．

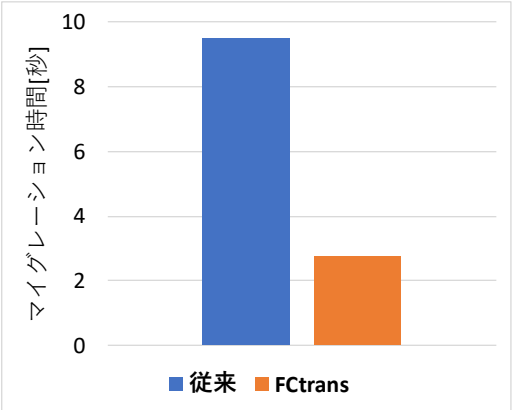

図 10　マイグレーション時間

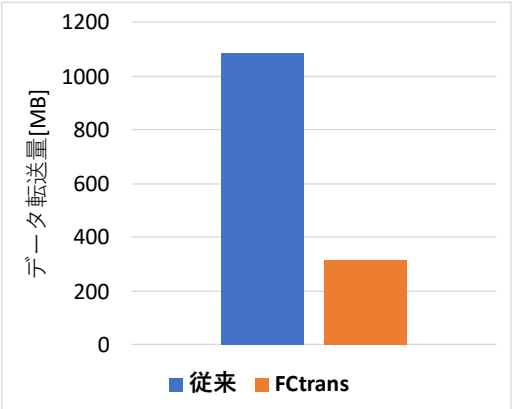

図 11　分割マイグレーション中のデータ転送量

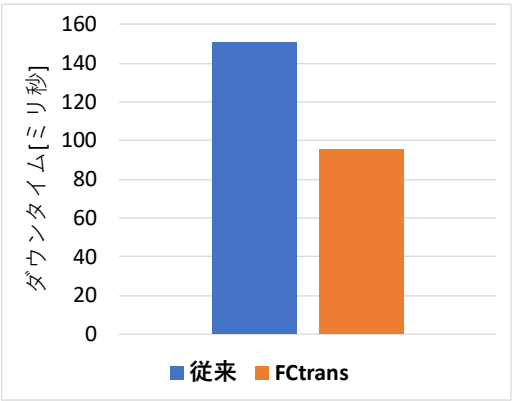

図 12　ダウンタイム

5.3　オーバヘッド

FCtrans において未使用メモリを追跡することによるオーバヘッドを調べた．そのために，カーネル，initrd，systemd に要した処理時間の合計を出力する systemd-analyze コマンドを用いて，VM 内の OS の起動に要した時間を測定した．これは OS の起動時には未使用メモリへの大量の

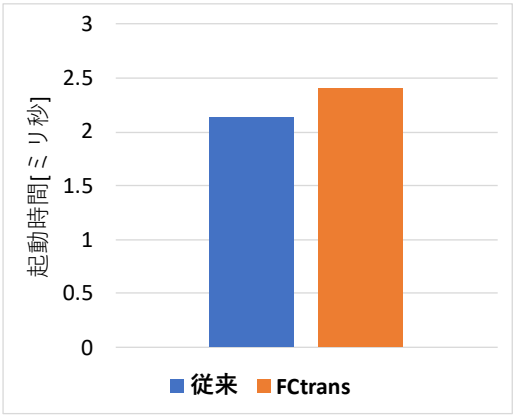

図 13　OS の起動時間

アクセスが行われるためである．VM 内ではデスクトップ環境を使用しないようにした．実験結果を図 13 に示す．FCtrans では OS の起動時間が従来システムに比べて 13 % 長くなることがわかった．FCtrans では VM が最初にメモリページにアクセスした時にページフォールトが発生し，userfaultfd 機構を用いて物理メモリを割り当て，使用ビットマップを更新するためである．従来システムではあらかじめすべてのメモリページに物理メモリを割り当てるため，その後にはページフォールトは発生しない．

6. 関連研究

VSwapper [3] は仮想メモリを用いる場合の VM の性能を改善している．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ディスク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みが行われた場合にはデータをディスク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．この手法はリモートページングにも適用することができると考えられる．

VM マイグレーションにおいて未使用メモリを転送しないようにする様々な最適化手法が提案されている．QEMU では転送する前にページ内のデータをスキャンし，すべてのデータが 0 のゼロページの場合にはそのことを表す 1 バイトのデータだけを転送する．この手法はすべてのメモリをスキャンするオーバヘッドが大きく，スキャン時に未使用メモリにも物理メモリが割り当てられてしまう．KVM における最適化として，メモリページの転送を制御するダーティビットマップを用いて未使用メモリを転送しないようにする手法が提案されている [4]．この手法では VM の起動時からログダーティ機構を用いてメモリへの書き込みを追跡し，未使用メモリの場合にはダーティビットマップのビットを 1 にしないようにする．FCtrans では userfaultfd 機構を用いてメモリへのアクセスを追跡し，独自の使用ビットマップを用いている．

ME2 [6] や SonicMigration [7] では VM 内の OS カーネルを拡張することで，未使用メモリを転送しないようしている．ME2 は VM 内の仮想メモリをスキャンすることで割り当てられていないページを見つけ，マイグレーション時にはそのことを表す 1 バイトのデータだけを転送する．SonicMigration は VM 内の使われていないページのアドレスをハイパーバイザとの間の共有メモリに書き込み，マイグレーション時にはそのページを転送しない．この手法は未使用ページだけでなく，OS が保持しているページキャッシュを転送しないようにするのにも利用されている．これらの手法は VM 内の OS カーネルを変更する必要があるため，適用可能性に制限がある．

VM イントロスペクションを用いて VM の外側で未使用ページを特定し，転送しないようにする手法も提案されている [8],[9]．VM イントロスペクションは VM のメモリを解析することで VM 内の OS データを取得する手法である．VM イントロスペクションに基づくマイグレーションでは，VM のページを転送する際に OS 内でのそのページの利用用途を調べ，未使用であれば転送を行わない．文献 [9] の手法では，ページキャッシュについても転送を行わない．FCtrans における OS が解放したメモリの特定手法はこれらの手法と同じであるが，FCtrans では userfaultfd 機構を用いた VM レベルでの未使用メモリの管理と連携させている．

7. まとめ

本稿では，未使用メモリに関連する不要なネットワーク転送に着目して，複数ホストにまたがる VM の高速化を実現するシステム FCtrans を提案した．FCtrans は VM の未使用メモリを追跡することでメモリ使用状況を管理し，分割マイグレーション時およびリモートページング時に未使用メモリのデータを転送しないようにする．VM がアクセスして一旦，使用中になったメモリページであっても，VM 内の OS が解放した後は未使用状態に戻す．そのために，LLView を用いて VM の外から OS のメモリ管理情報を取得する．FCtrans を Linux の userfaultfd 機構を用いて QEMU-KVM に実装した．リモートページングの性能向上について調べる実験を行い，分割マイグレーション後のページイン回数を 90 %以上削減できることが分かった．また，従来の分割マイグレーションと比較してマイグレーション時間は 71 %，ダウンタイムは 37 %の短縮が可能であることが分かった．一方，未使用メモリを追跡することにより VM 内の OS の起動時間は 6 %長くなることがわかった．

今後の課題は，VM 内の OS が解放したメモリ領域を探索する際に VM を一時停止する時間を削減することである．現在の実装では，探索中は VM を停止させ続けているため，VM のメモリサイズが大きくなるとダウンタイムが増加する．そのため，VM をできるだけ停止させずに探索を行える方法を検討する必要がある．また，探索の実装は VM 内の OS の種類やバージョンに依存するため，複数の VM 内の様々な OS に同時に対応できるようにする必要がある．さらに，未使用メモリを考慮して部分マイグレーション [5] の高速化を行うことも計画している．

参考文献

[1] M. Suetake, T. Kashiwagi，H. Kizu, and K. Kourai: SmemV: Split Migration of Large-Memory Virtual Machines in IaaS Clouds, Proc. IEEE Int. Conf. Cloud Computing, pp.285–293, 2018.
[2] 尾崎雄一, 山本裕明, 光来健一: GPU を用いた OS レベルでの障害検知, 第 142 回 OS 研究会, 2018.
[3] N. Amit, D. Tsafrir, and A. Schuster: VSWAPPER: A Memory Swapper for Virtualized Environments, In ACM International Conference on Architectural Support for Programming Languages and Operating Systems, pp.349–366, 2014.
[4] L. Li and Y. Zhang: Migration Optimization, KVM Forum 2015, 2015
[5] 柏木 崇広, 末竹 将人, 光来 健一: 分割メモリ VM の高速かつ柔軟な部分マイグレーション, 第 30 回コンピュータシステム・シンポジウム (ComSys 2018), 2018
[6] Y. Ma, H. Wang, J. Dong, Y. Li, and S. Cheng: ME2: Efficient Live Migration of Virtual Machine with Memory Exploration and Encoding, In Proc. 2012 IEEE International Conference on Cluster Computing, pp.610-613, 2012.
[7] A. Koto, H. Yamada, K. Ohmura, and K. Kono: Towards Unobtrusive VM Live Migration for Cloud Computing Platforms, In Proc. Asia-Pacific Workshop on Systems, 2012
[8] J. Chiang, H. Li, and T. Chiueh: Introspection-based Memory De-duplication and Migration, In Proc. ACM SIGPLAN/SIGOPS nternational Conference on Virtual Execution Environments, pp.51-62, 2013
[9] C. Wang, Z. Hao, L. Cui, X. Zhang, and X. Yun: Introspection-based Memory Pruning for Live VM Migration, Int. J. Parallel Program, 45(6), pp.1298–1309, 2017