複数ホストにまたがる大容量メモリVMの未使用メモリに着目した高速化

田内　聡一朗[1]　光来　健一[1]

概要：近年，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データをホスト間で転送する．しかし，従来の分割マイグレーションとリモートページングでは使われていないメモリであっ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は分割マイグレーション時には移送先ホストに未使用メモリのデータを転送しないようにする．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．FCtransはVMのマイグレーション開始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．ゲストOSが解放したメモリも未使用メモリとして扱えるように，OSのメモリ管理情報をVMのメモリ管理と統合する．FCtransをKVMに実装し，統合テストベッドのStarBEDを用いて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負担が大きく，運用の柔軟性を低下させる．

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や仮想デバイスの状態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．分割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）する．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空きメモリを確保する．この処理はリモートページングと呼ばれる．

一方，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データの転送を行う．そのため，必ず，VMのメモリサイズに比例した時間がかかる．また，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．VMにとって未使用メモリのデータは不要であるため，ネットワーク経由で転送する必要はない．

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のデータを転送しないようにする．これにより，分割マイグレーションを高速化することができる．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，未使用メモリに対して

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

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を必要とした時には，メインホストの空きメモリを割り当てることにより即座に VM の実行を再開させることができる．メインホストにまだ空きメモリがある場合にはページアウトも行わない．FCtrans はマイグレーション開始時から未使用メモリの追跡を行い，マイグレーション後も追跡を継続する．ゲスト OS のメモリ管理情報を VM のメモリ管理と統合することにより，OS が解放したメモリも未使用メモリとして扱えるようにする．

FCtrans を QEMU-KVM に実装し，分割マイグレーションとリモートページングの高速化を実現した．FCtrans はマイグレーション開始時にホスト OS から VM のメモリ使用状況を一括で取得し，メモリの使用ビットマップを構築する．それ以降は Linux の userfaultfd 機構を用いて，未使用メモリへの初めてのアクセスを検出して使用ビットマップを更新する．また，OS が解放したメモリの情報を LLView[2] を用いて定期的に取得し，使用ビットマップに反映する．この使用ビットマップを用いて，未使用メモリの場合にはネットワーク転送を行わない．統合テストベッドの StarBED 上で実験を行った結果，分割マイグレーション時間を短縮でき，マイグレーション後の VM の性能を向上させられることが分かった．

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．3 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. 複数ホストにまたがる大容量メモリ VM

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断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する．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，移送元ホストと移送先ホストの間で 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，残りのメモリデータと仮想 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 24TB 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．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を確保する必要がある．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く，運用の柔軟性を低下させるためである．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等の間，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．

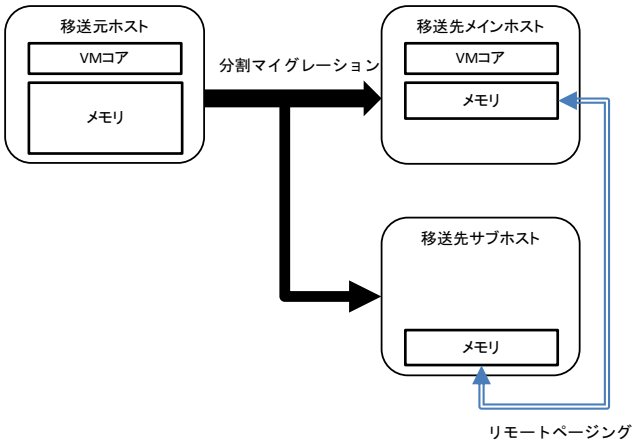

図 1　分割マイグレーション

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と呼ばれ，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 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のメモリアクセスはそれまでのアクセス履歴に基づいて予測される．

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．ただし，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．

このように VM のメモリは必要に応じてネットワーク転送されるが，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ネットワーク転送を行っており，VM のメモリサイズに比例した時間，もしくは再送によってそれ以上の時間がかかる．

また，従来のリモートページングは VM がサブホスト上の未使用メモリに対して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．このようなリモートページングを行うと VM の性能が低下する．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 は停止されるためである．ページアウトは仮想 CPU の再開後に非同期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．

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．移送先ホストでは未使用メモリに物理メモリを割り当てない．これにより，ネットワーク転送量を削減し，分割マイグレーションを高速化することができる．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ホストが決定されるため，使用中のメモリは優先的にメインホストに転送される．そのため，分割マイグレーション後のリモートページングの発生をさらに抑えることができる．この手法は分割マイグレーションだけでなく，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．

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も，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を使い始めようとした場合には，そのメモリデータのネットワーク転送は行わず，メインホストの空きメモリを割り当ててアクセスさせる．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，従来はメインホストで空きメモリを確保するためにページインの際にはページアウトも同時に行われていたが，FCtrans では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を行わない．それにより，ページアウトのオーバヘッドも削減することができる．

このような分割マイグレーションとリモートページングの高速化を可能にするために，FCtrans は VM のマイグレーション開始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．このようにすることで，分割マイグレーションを行うまでは追跡のオーバヘッドが発生しないようにする．統合マイグレーション [5] を行って VM が再び 1 つのホストで動作するようになれば追跡をやめる．VM の作成時にはすべてのメモリ領域が未使用であり，VM の起動後にメモリにアクセスされるとその領域は使用中となる．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，メインホストで一元管理を行う．

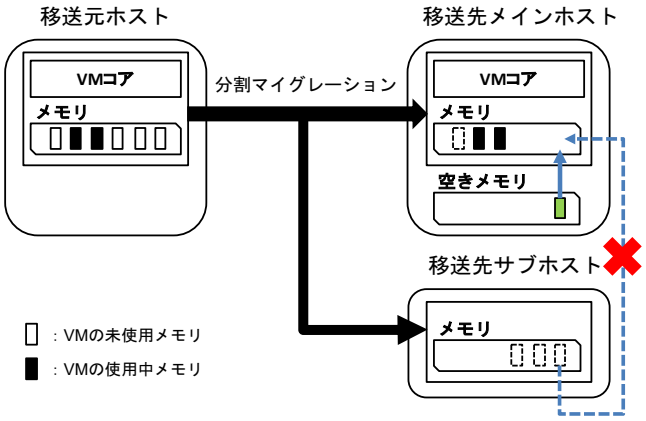

図 2　FCtrans による高速化

FCtrans は定期的にゲスト OS のメモリ使用状況を VM のメモリ管理に反映させる．ゲスト OS がメモリ領域を確保して一度でもアクセスを行うと，OS がそのメモリを解放して使わなくなったとしても，VM から見るとそのメモリは使用中のままになる．そこで，FCtrans は VM の外から VM のメモリを解析することによりゲスト OS のメモリ管理情報を取得する．そして，OS が使わなくなったメモリがあれば対応する VM のメモリを未使用状態とする．同時に，VM に割り当てたメモリを解放して空きメモリを増やす．これにより，ページイン発生時にできるだけページアウトを行わずに済ませられるようにする．

4. 実装

分割マイグレーションとリモートページングが実装された QEMU-KVM 2.11.2 に FCtrans を実装した．ゲスト OS として Linux を対象とした．

4.1 VM の未使用メモリの追跡

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．

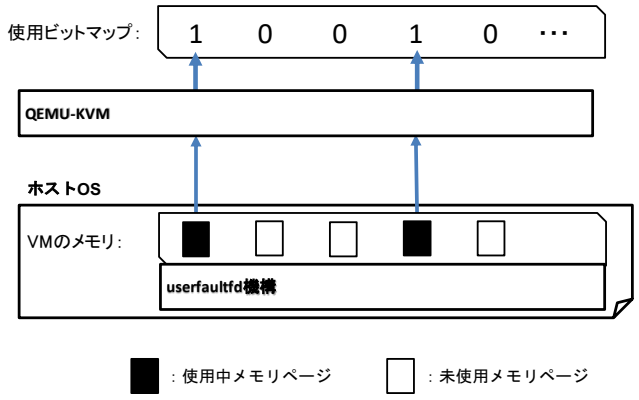

図 3　VM の未使用メモリの追跡

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 はホスト OS の userfaultfd 機構を用いる．FCtrans は QEMU-KVM 上で VM に割り当てられているメモリ領域を userfaultfd 機構に登録する．KVM では VM の作成時には物理メモリが割り当てられていないため，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が発生し，QEMU-KVM に通知される．その際に，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．

FCtrans は分割マイグレーションを行うまでは未使用メモリの追跡を行わない．マイグレーションを行う時に初めて VM のメモリ使用状況の情報が必要になることと，未使用メモリへのアクセスを検出するオーバヘッドが大きいためである．我々の実験によると，ゲスト OS の起動時に未使用メモリの追跡を行うと 13%の性能低下が見られた [12]．FCtrans はマイグレーション開始時に一括で使用ビットマップを構築する．そのために，QEMU-KVM プロセスのページテーブルの情報をホスト OS の proc ファイルシステムの pagemap 疑似ファイルから取得する．QEMU-KVM 内の VM のメモリ領域についてページテーブルのエントリを調べ，ページが存在しなければ未使用メモリとして使用ビットマップの対応するビットを 0 にする．ページが存在する場合にはビットを 1 にする．

4.2　ゲスト OS のメモリ管理情報の統合

ゲスト OS が使わなくなったメモリを VM の未使用メモリとして扱えるように，FCtrans は定期的にゲスト OS のメモリ管理情報を取得して VM のメモリ管理に反映させる．Linux は物理メモリを Buddy System で管理しており，2 のべき乗の単位で連続した物理ページを割り当てる．そこで，FCtrans は VM レベルでは使用中になっているが，ゲスト OS においては空きメモリになっているページがないかを調べる．そのようなページが見つかった場合に

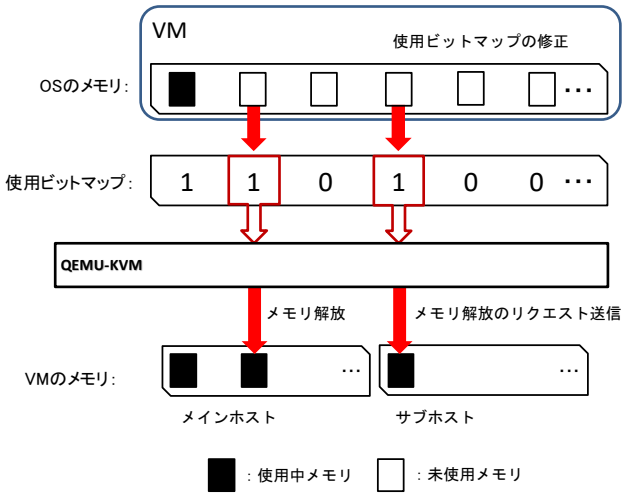

図 4　VM レベルと OS レベルのメモリ管理情報の統合

```
for (pfn = 0; pfn < max_pfn; pfn++) {
    page = pfn_to_page(pfn);

    if (PageBuddy(page)) {
        size = 1 << page_order(page);

        for (i = 0; i < size; i++) {
            page = pfn_to_page(pfn + i);
            /*対応する VM のメモリを未使用状態へ戻す*/
        }

        pfn += size - 1;
    }
}
```

図 5　ゲスト OS のメモリ管理情報の取得

は，そのページを先頭とする空きメモリ領域に対応する VM のメモリを未使用状態に戻す．具体的には，図 4 のように使用ビットマップの対応するビットを 0 にする．そのメモリがメインホスト上にある場合には VM に割り当てられた物理メモリを解放する．サブホスト上にある場合にはサブホストにリクエストを送り，サブホストのメモリを解放する．その間に同時にゲスト 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，初めてのアクセスの場合と同様に処理される．

VM の外から透過的にゲスト OS のメモリ管理情報を取得するために，FCtrans は LLView[2] を用いて Linux のページ構造体の情報を取得する．この処理は Linux のヘッダファイルを用いて図 5 のように記述することができる．ページ番号 0 のページから順番にページ構造体を調べ，PageBuddy 関数の中でページのマップカウントを取得する．マップカウントの値からページが Buddy System によって管理される空きメモリであるかどうかを判定する．

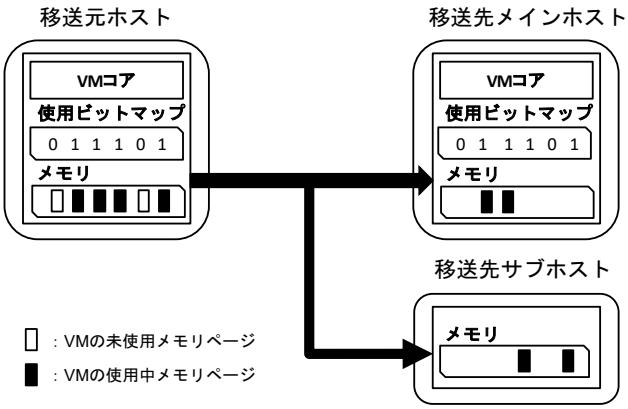

図 6　分割マイグレーションの高速化

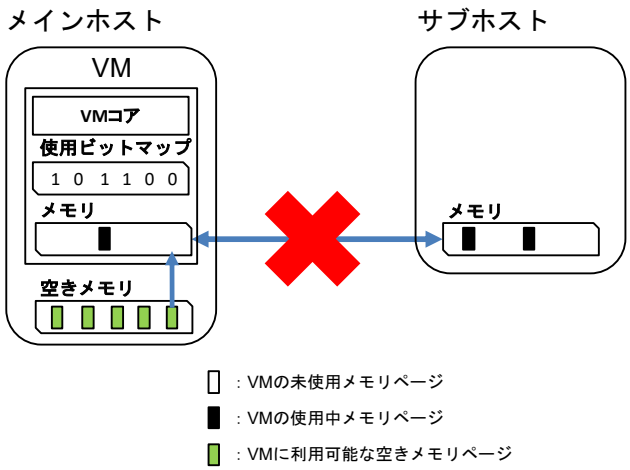

図 7　未使用メモリに対するリモートページング

空きメモリならば page_order 関数を用いてそのページを先頭に連続して管理される領域のページ数を取得する．そして，必要に応じて対応する VM のメモリを未使用状態に戻す処理を行う．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．

4.3　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 6 のように未使用メモリのデータ転送を行わない．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使用中メモリの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の転送も行わない．つまり，未使用メモリはどのホストにも存在しない状態となる．これにより，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，および，ネットワーク転送，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．この再構築は移送元ホストから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移送先メインホストで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．

4.4　未使用メモリを考慮したリモートページング

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．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，つまり，当該ページが未使用であった場合にはページインは行わない．その代わりに，図 7 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．そして，使用ビットマップの対応するビットを 1 にする．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，未使用メモリへのアクセス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 はこのようなページインの最適化を行うために，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で VM が使用する物理メモリ量を一定に保っていたが，FCtrans では使用する物理メモリ量が VM への割り当てより小さ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 VM が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る．空きメモリがある場合にはページアウトを行わないため，システムへの負荷を軽減することができる．

5.　実験

FCtrans を用いて未使用メモリを考慮した分割マイグレーションとリモートページングの性能向上について調べ

る実験を行った．比較として，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．この実験では，NICT の統合テストベッドの StarBED[13] によって提供される 3 台のホストを用い，1 台ずつ移送元ホスト，移送先メインホスト，移送先サブホストとした．これらのホストは Intel Xeon CPU E5-2683 の CPU，324GB のメモリ，10GbE の NIC を搭載している．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．VM には 1 個の仮想 CPU，240GB のメモリを割り当て，Linux 4.14 を動作させた．分割マイグレーション時には VM のメモリを 120GB ずつ 2 台のホストに分割した．

5.1 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，VM の使用中のメモリ量を変化させて分割マイグレーション中のネットワーク転送量を測定した．この実験は，500MB のメモリが使用中であったゲスト OS の起動直後と，100GB を使用中の場合，200GB を使用中の場合について行った．実験結果を図 8 に示す．従来の分割マイグレーションはすべてのメモリデータを転送するため，常に 251GB 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，OS 起動直後の場合は 4.8GB，100GB 使用中の場合は 110GB，200GB 使用中の場合は 215GB にネットワーク転送量を削減することができた．

分割マイグレーションにかかった時間を図 9 に示す．従来の分割マイグレーションと比較して，FCtrans は OS 起動直後の場合には 98%，100GB 使用中の場合には 72%，200GB 使用中の場合でも 36%マイグレーション時間を削減できた．これらの結果から，FCtrans ではネットワーク転送量に応じてマイグレーション時間が短縮されることが分かった．一方，従来の分割マイグレーション時間はネットワーク転送量に対して長い時間がかかっていることも分かった．この原因については現在，調査中である．

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が 300 ミリ秒で転送可能と予測された時に VM を一時停止し，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 10 に示す．FCtrans では常に同程度のダウンタイムであったのに対し，従来の分割マイグレーションでは VM のメモリサイズが増加すると大幅にダウンタイムが長くなった．これは，VM の一時停止後に行ったメモリ転送に予測よりも時間がかかったためと考えられる．その原因は上記のマイグレーション時間が長い原因と同じだと考えられる．

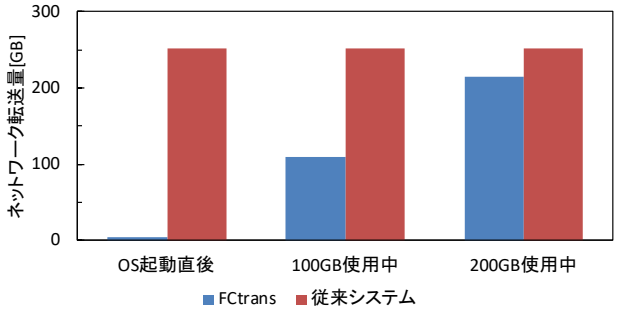

図 8 分割マイグレーション中のネットワーク転送量

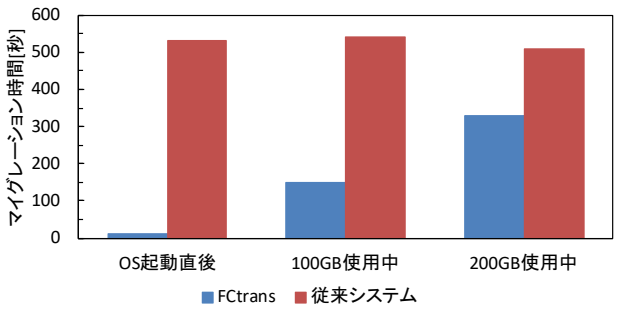

図 9 分割マイグレーション時間

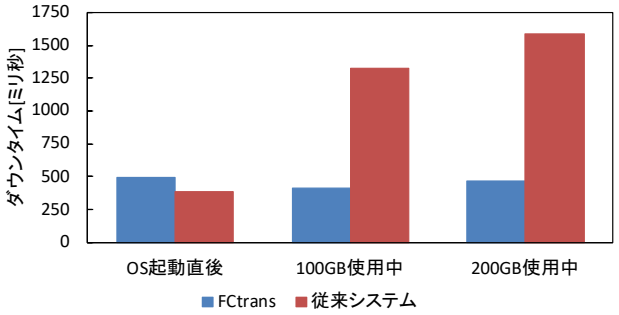

図 10 ダウンタイム

5.2 分割マイグレーション後の VM の性能

分割マイグレーション後の VM の性能向上について調べるために，VM 上でインメモリ・データベースの memcached [10] を動作させた．リモートページングを発生させやくするために，この実験では 8GB のメモリを持つ VM を用い，2 台のホストに 4GB ずつに分割した．この VM 上で動作する memcached に 6GB のメモリを割り当て，memaslap ベンチマーク [11] を用いてそのデータにアクセスした．

まず，memcached に対して 64 バイトずつの読み書きを行い，memcached のメモリを 500MB 程度使用した．この場合には memcached のメモリがメインホストに入り切った．この時のページング回数とスループットの推移をそれぞれ図 11，図 12 に示す．従来システムでは 1 秒間の平均ページング回数が 27 回であったのに対し，FCtrans ではベンチマーク開始直後を除いてページングは発生しなかっ

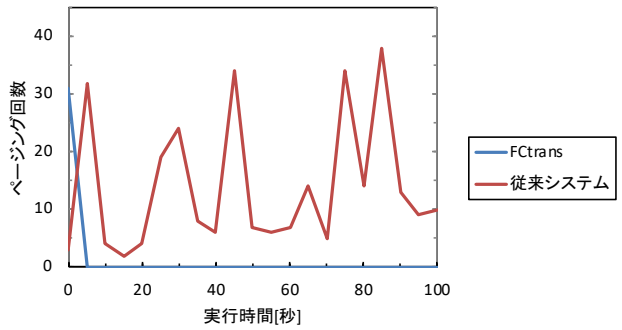

図 11　ページング回数（データ量：小）

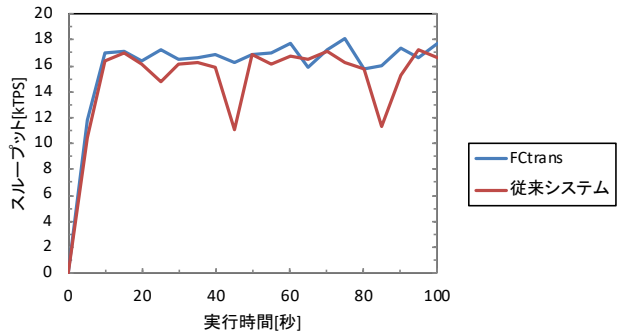

図 12　memcached の性能（データ量：小）

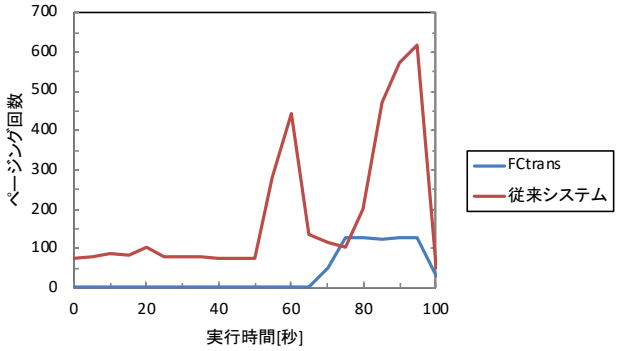

図 13　ページング回数（データ量：大）

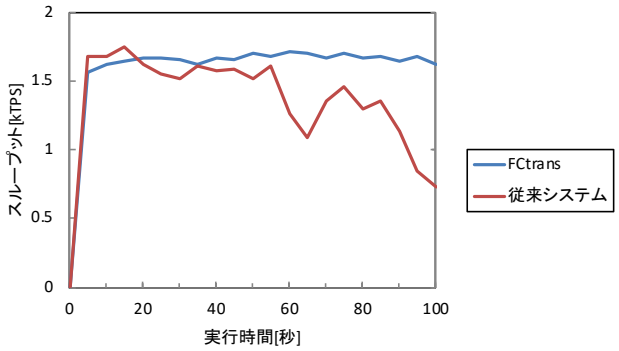

図 14　memcached の性能（データ量：大）

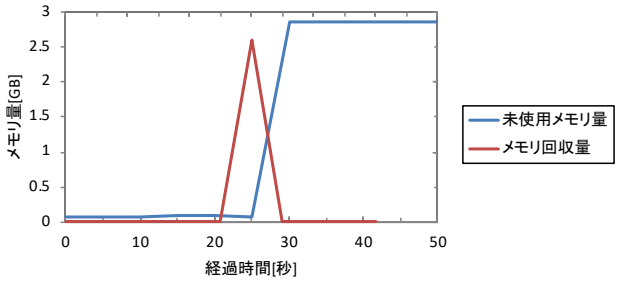

図 15　FCtrans によって回収されたメモリ量

た．これは memcached が未使用メモリに初めてアクセスした時に，従来システムではサブホストからのデータ転送が必要になる場合があったのに対し，FCtrans ではその必要がなかったためである．これにより，FCtrans においてスループットは 7%向上した．性能向上率が小さいのは要求するデータサイズが小さく，通信のオーバヘッドが大きかったためだと考えられる．

次に，memcached に対して 1MB ずつの読み書きを行い，memcached のメモリを 6GB 程度使用した．この場合には memcached のメモリがメインホストに入り切らなかった．この時のページング回数とスループットの推移をそれぞれ図 13，図 14 に示す．従来システムでは途中からページング回数が大幅に増加しており，それに伴ってスループットも低下している．それに対して，FCtrans では途中まではページングが発生せず，memcached のメモリがメインメモリに入り切らなくなるとページアウトのみが発生した．これは memcached が未使用メモリに初めてアクセスした時に割り当てられる空きメモリを確保するためである．ページインが発生しなかったため，スループットへの影響はほとんどなかった．

5.3　ゲスト OS が解放したメモリの回収性能

FCtrans がゲスト OS によって解放されたメモリに対応する VM のメモリを未使用状態に戻す回収性能を調べるた．5.2 節のベンチマーク終了 30 秒前からの 50 秒間に FCtrans によって回収されたメモリ量を測定した．この実験では 3GB のメモリを持つ VM を用いて，2 台のホストに 1.5GB ずつに分割した．この VM 上で動作する memcached に 3GB のメモリを割り当てた．ベンチマーク終了時に回収されたメモリ量を図 15 に示す．ベンチマーク終了時に memcached が使用していたメモリをすべて回収可能であることを確認した．ゲスト OS によって解放されたメモリを回収する時間は VM のメモリ 3GB あたり 77 ミリ秒であった．

6.　関連研究

VM マイグレーションについては，未使用メモリを転送しないようにする様々な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，VM マイグレーションにおい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ており，マイグレーションを開始するまでは追跡を行わない．

ME2 [6] や SonicMigration [7] ではゲスト OS 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ゲスト OS の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おいてゲスト 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 においてゲスト OS が解放したメモリを特定する手法はこれらの手法と同じであるが，FCtrans ではゲスト OS のメモリ管理情報を VM のメモリ管理に統合しており，マイグレーション時のオーバヘッドが小さい．

VSwapper [3] は仮想メモリを用いてページングを行う場合の 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．

7. まとめ

本稿では，未使用メモリの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 のメモリ使用状況を管理することにより，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．そのために，分割マイグレーションの開始時から VM の未使用メモリの追跡を行う．ゲスト OS のメモリ管理情報を VM のメモリ管理と統合することにより，OS が解放したメモリも未使用メモリとして扱えるようにする．FCtrans を Linux の userfaultfd 機構と LLView[2] を用いて QEMU-KVM に実装した．StarBED を用いた実験の結果，分割マイグレーションの時間を最大で 98%短縮することができた．また，複数ホストにまたがる VM 上で動作するアプリケーションの性能を改善することができた．

今後の課題は，ゲスト OS のメモリ管理情報を取得する際に 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，取得中は 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ない方法を検討する必要がある．また，この実装はゲスト OS の種類やバージョンに依存するため，VM 内で動作する様々な OS に同時に対応できるようにする必要がある．さらに，部分マイグレーション [5] などの他のマイグレーション手法への適用も計画している．

謝辞

本研究成果は，国立研究開発法人情報通信研究機構の委託研究により得られたものです．

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] Y. Ozaki, S. Kanamoto, H. Yamamoto, and K. Kourai: Detecting System Failures with GPUs and LLVM , n Proceedings of the 10th ACM SIGOPS Asia-Pacific Workshop on Systems (APSys 2019), pages 47-53, pp.47-53, 2019.

[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] T. Takahiro and K. Kourai: Flexible and Efficient Par-

tial Migration of Split-memory VMs, In Proceedings of the IEEE 13th International Conference on Cloud Computing (CLOUD 2020), pp.248-257, 2020.
[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.
[10] memcached, A Distributed Memory Object Caching System, http://memcached.org/.
[11] memaslap, Load Testing and Benchmarking a Server, http://docs.libmemcached.org/bin/memaslap.html.
[12] 田内聡一朗, 光来健一: 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, 第 146 回 OS 研究会, 2019.
[13] StarBED, https://starbed.nict.go.jp/.