複数ホストにまたがって動作するVMのチェックポイント・リストア

村田 時人[1]　光来 健一[1]

概要：近年，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，複数のホストにメモリを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．しかし，マイグレーション後の分割メモリ VM は複数のホストにまたがって動作するため，ホストやネットワークの障害の影響を受ける可能性が高くなる．障害対策としてチェックポイント・リストアが用いられているが，従来のチェックポイント手法を分割メモリ VM に適用すると，ホスト間で大量のメモリデータのやりとりが発生するためオーバヘッドが大きい．また，従来のリストア手法では複数ホストに分割された状態で VM を復元することができない．本稿では，複数ホストにまたがって動作する分割メモリ VM の柔軟で効率のよいチェックポイント・リストアを可能とするシステム D-CRES を提案する．D-CRES のチェックポイントは各ホストで並列に分割メモリ VM のメモリを保存することで，ホスト間でのメモリデータのやり取りを発生させず，高速なチェックポイントを実現する．D-CRES のリストアは各ホストで取得したチェックポイントから複数のホストそれぞれで並列に復元を行うことにより，分割メモリ VM としての復元を可能にする．D-CRES 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，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，2019 年には 24TB のメモリを持つ VM も提供予定である．このような VM はインメモリ・データベースやビッグデータの解析などに利用されている．VM が動作しているホストのメンテナンスや負荷分散を行う際には，VM を別のホストにマイグレーションする必要がある．マイグレーションでは，VM のメモリを移送元ホストから移送先ホストへ転送するため，移送先ホストには 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 に対して，十分な空きメモリを持ったホストを常に確保し続けるのはコストや柔軟性の面で望ましくない．

そこで，複数のホストにメモリを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は VM 本体とアクセスが予測されるメモリ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をサブホストに転送する．分割マイグレーション後はメインホスト上で VM が動作し，サブホストはメインホストにメモリを提供する．このようにメインホストとサブホストにまたがって動作する VM は分割メモリ VM と呼ばれる．VM がサブホストに存在するメモリを必要とした場合には，当該メモリのデータをメインホストに転送（ページイン）する．同時に，メインホスト上のメモリの内，アクセスされないことが予測されるメモリのデータをサブホストに転送（ページアウト）する．この一連の処理はリモートページングと呼ばれる．

このように，分割メモリ VM は複数のホスト間で通信しながら動作するため，1 台のホスト上で動作する VM と比較して，ホストやネットワークの障害の影響を受ける可能性が高くなる．従来の障害対策としては，チェックポイント・リストアが用いられてきた．この手法は，定期的に VM の状態をバックアップとして保存（チェックポイント）し，障害発生時には最新のチェックポイントから VM を復元（リストア）する．しかし，分割メモリ VM に従来手法を適用すると二つの問題が生じる．一つは，チェックポイント時に大量のリモートページングが発生し，オーバヘッドが大きいことである．これは，サブホスト上のメモリを一度メインホストにページインしてからメインホスト上で保存するためである．もう一つの問題は，従来手法では複数のホストに分割された状態で VM を復元することができないことである．そのため，リストア時には十分な空きメ

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

モリを持ったホストが必要となる．

本稿では，複数ホストにまたがって動作する分割メモリ VM の柔軟で効率のよいチェックポイント・リストアを可能とするシステム D-CRES を提案する．D-CRES のチェックポイントはメインホストに存在するメモリだけをメインホストで保存し，サブホストに存在するメモリはサブホストにおいて保存する．各ホストでメモリの保存を行うことで，チェックポイントのためにサブホストとメインホスト間でリモートページングを発生させないようにすることができる．また，各ホストで並列にメモリを保存することで高速なチェックポイントが実現できる．VM を動作させたまま保存を行うライブチェックポイントでは，実行中の VM によって引き起こされるリモートページングを考慮してメモリを保存する．D-CRES のリストアは複数のホストに分割された状態での VM の復元を可能にする．そのために，各ホストで取得したチェックポイントから複数のホストそれぞれで並列に復元を行う．また，リストア時に VM のメモリ分割を変更できるようにすることで，利用可能なホスト構成に応じたリストアを可能とする．

D-CRES を KVM に実装し，分割メモリ VM のチェックポイント・リストアを実現した．メインホストにおけるチェックポイント処理では，ネットワーク・ページテーブルを用いてメインホストに存在する VM のメモリだけを保存し，サブホストに存在するメモリはサブホスト ID のみを保存する．サブホストにおけるチェックポイント処理では，ページ・サブテーブルを用いてサブホストに存在するメモリだけを保存する．ライブチェックポイントに対応するために，各ホストで VM のメモリ空間に 1 対 1 に対応するスパースファイルを用いてメモリデータを保存する．リストア処理では，VM のメモリおよび，メインホストのネットワーク・ページテーブル，サブホストのページ・サブテーブルを復元する．D-CRES を用いて実験を行った結果，分割メモリ VM に従来手法を適用するよりも，チェックポイントとリストアの性能が大幅に向上することがわかった．

以下，2 章では分割メモリ VM のチェックポイント・リストアの問題点について述べる．3 章では分割メモリ VM の柔軟で効率のよいチェックポイント・リストアを可能とするシステム D-CRES を提案し，4 章でその実装について述べる．5 章では，D-CRES の性能を調べるために行った実験の結果について述べる．6 章で関連研究に触れ，7 章で本稿をまとめる．

2. VM のチェックポイント・リストア

IaaS 型クラウドでは大容量メモリを持つ VM が提供されるようになっている．ホストのメンテナンス等で VM をマイグレーションする際には，移送先ホストに十分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 の場合には移送先となるホストの維持コストや運用の柔軟性が問題となる．そこで，図 1 のように一つのメインホストと複数のサブホストにメモリを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は VM 本体と今後アクセスが予測されるメモリを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をサブホストへ転送する．マイグレーション後の VM のメモリアクセスは LRU に基づいて予測する．

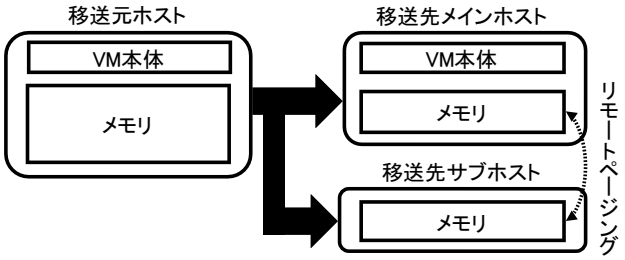

図 1 分割マイグレーション

分割マイグレーション後はメインホスト上で VM が動作し，サブホストはメインホストへメモリを提供する．この VM は分割メモリ VM と呼ばれ，必要に応じてホスト間でリモートページングを行いながら動作する．VM がサブホストに存在するメモリを必要とした場合には，当該メモリをメインホストにページインする．同時に，メインホスト上のメモリの内，アクセスされないことが予測されるメモリをサブホストにページアウトする．このようにすることで，十分な空きメモリがないメインホスト上でも大容量メモリを持つ VM を動作させることができる．

一方，分割メモリ VM は複数のホスト間で通信を行いながら動作するため，1 台のホスト上で動作する通常の VM と比べると，ホストやネットワークの障害の影響を受ける可能性が高くなる．例えば，サブホストのいずれかで異常が発生し停止してしまった場合，VM のメモリの一部が失われてしまうため，VM の実行を継続できなくなる．また，ネットワーク障害が発生した場合，リモートページングで要求されるメモリデータをメインホストに転送することができなくなるため，障害から回復するまでは VM の実行を行うことができなくなる．

従来，障害対策としてチェックポイント・リストアと呼ばれる手法が用いられてきた．この手法は定期的にチェックポイントを取得して，VM の状態をディスク等にバックアップとして保存する．VM の状態は仮想 CPU や仮想デバイスの状態，メモリデータ，ディスクデータなどからなる．障害発生時にはリストアを行い，別のホスト上で VM の実行を再開する．その際に，VM をチェックポイント時に保存した状態に復元することで復旧時間を短くし，失われるデータを最小にすることができる．

しかし，分割メモリ VM に従来手法を適用すると二つ

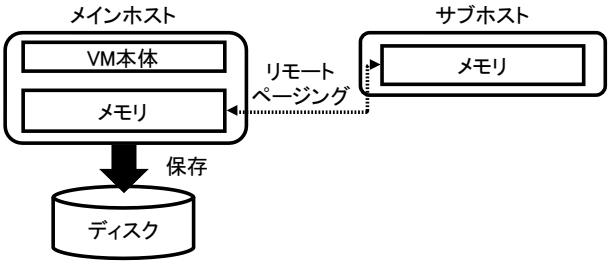

図 2 従来手法を用いた分割メモリ VM のチェックポイント

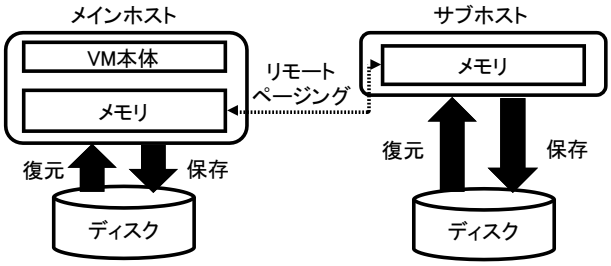

図 3 D-CRES のチェックポイント・リストア

の問題が生じる．一つは，図 2 のようにチェックポイント時に大量のリモートページングが発生し，オーバヘッドが大きいことである．メインホスト上のメモリは従来通りメインホスト上で保存されるが，サブホスト上のメモリは一度メインホストにページインしてからメインホスト上で保存されるためである．同時に，メインホストの空きメモリを確保するために，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も行われる．その後でページアウトしたメモリを保存することになると，再びページインが発生する．

もう一つの問題は，従来手法では複数のホストに分割された状態で VM を復元することができないことである．チェックポイント時にメインホスト上で VM のメモリ全体が保存されるため，1 台のホスト上で動作する通常の VM を保存したのと同じことになる．そのため，リストア時には分割前の VM のメモリサイズを確保できる，十分な空きメモリを持ったホストが必要となる．

3. D-CRES

本稿では，分割メモリ VM の柔軟で効率のよいチェックポイント・リストアを可能とするシステム D-CRES を提案する．D-CRES のチェックポイントは，図 3 のように各ホストで並列に VM のメモリの保存を行う．これにより，メインホストとサブホスト間でリモートページングを発生させないようにし，分割メモリ VM の高速なチェックポイントを実現する．また，D-CRES のリストアは各ホストで最新のチェックポイントを用いて複数のホストそれぞれで並列に分割メモリ VM の復元を行う．リストアを行う際に VM のメモリの再配置を行うことで，チェックポイント時とは異なるホスト構成で復元を行うこともできる．

D-CRES のチェックポイントは，メインホストではそのホスト上に存在する VM のメモリと仮想 CPU や仮想デバイスなどの VM 本体の状態だけを保存する．また，ディスクイメージの状態も通常の VM と同様にメインホストにおいて保存する．一方，サブホストではそのホスト上に存在する VM のメモリだけを保存する．メインホストとすべてのサブホストで状態を保存し終えるとチェックポイント処理が完了する．その後で，ホストやネットワークの障害に備えて，取得したチェックポイントを別のホストへ転送する．

D-CRES はライブチェックポイントにより VM をほとんど停止させずに VM の状態を取得することができる．ライブチェックポイントでは VM を動かしたままメモリを保存し，保存中に更新されたメモリについて追加で保存する．更新されたメモリが十分に少なくなったら VM を停止し，残りのメモリと VM 本体の状態を保存する．その際に，実行中の VM が発生させたリモートページングによるホスト間でのメモリの移動を考慮し，過不足なくメモリの保存を行う．メインホストにページインされたメモリはメインホストで保存し，サブホストで保存済みであればサブホストのチェックポイントから削除する．ページアウトされたメモリはサブホストで保存し，メインホストで保存済みであればメインホストのチェックポイントから削除する．

D-CRES のリストアを行う際には，まず，十分な空きメモリを持つメインホストとサブホストを探す．そして，最新のチェックポイントを用いて，メインホストでは VM のメモリと VM 本体の状態を復元し，サブホストでは VM のメモリだけを復元する．メインホストとサブホストで状態の復元が完了すると，ホスト間でリモートページングのためのネットワーク接続を確立して分割メモリ VM の実行を再開する．

D-CRES はリストア時に VM のメモリ分割を変更し，異なるホスト構成で分割メモリ VM を再開することができる．これにより，チェックポイント時と同じ空きメモリを持ったホスト群が存在しない場合でも，柔軟に分割メモリ VM の復元を行うことができる．また，1 つのホストに十分な空きメモリがある場合にはそのホスト上で通常の VM として再開させることもできる．VM のメモリ分割を変更する際には，保存されたメモリアクセス履歴を基にメモリの配置を変更する．分割マイグレーションを行う時と同様に，アクセスされることが予測されるメモリができるだけメインホストに配置されるようにする．

4. 実装

分割マイグレーションおよびリモートページングが実装された QEMU-KVM 2.4.1 に D-CRES を実装した．

4.1 分割メモリ VM

分割メモリ VM は各ホストでメモリ管理を行うことによって実現されている．メインホストでは QEMU-KVM が動作しており，VM のメモリページと各ホストの対応を登録したネットワーク・ページテーブルを管理している．ネットワーク・ページテーブルはページ番号からホスト ID を検索するための表である．メインホストはこのテーブルを参照することでどのページがどのホストに存在しているかという情報を取得する．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が動作しており，VM のどのページがそのサブホストに存在しているかを登録したページ・サブテーブルを管理している．ページ・サブテーブルはページ番号からメモリデータを検索するための表である．

メインホストがサブホストからページインを行う際には，ネットワーク・ページテーブルから当該ページが存在するサブホストを検索してページイン要求を送る．サブホストのメモリサーバはページ・サブテーブルから当該ページを検索して，そのメモリデータをメインホストに転送する．その後，メインホストでアクセスされないことが予測されるページを探し，サブホストにページアウト要求を送る．リモートページングを行うたびに，それぞれのテーブルが各ホストで更新される．メモリのアクセス予測については，各ページのアクセス履歴を管理して LRU に基づいて行われる．

4.2 メインホストでのチェックポイント

メインホストの QEMU-KVM はチェックポイントのために使われる migrate コマンドを受け取ると，サブホストにチェックポイント・コマンドを送信する．その後，VM の状態を保存するためのチェックポイント・ファイルを生成し，ネットワーク・ページテーブルに基づいてメモリの保存を行う．メインホストに存在するページについては，そのアドレスとデータ，アクセス履歴を保存する．サブホストに存在するメモリについては，そのアドレスとサブホスト ID のみを保存する．すべてのメモリの保存が完了すると VM 本体の状態を保存し，サブホストでのチェックポイントの取得完了を待ってチェックポイント処理を完了する．

従来の QEMU-KVM は通常の VM のライブチェックポイントに対応しているが，その実装には 2 つの問題がある．一つは，メモリの保存中に更新されたページの情報がチェックポイント・ファイルに追記されていくことである．メモリが更新されるたびにそのデータやアドレスが追記されるためファイルが肥大化する．また，リストア時には保存された順に，古いメモリ情報を復元してから新しいメモリ情報で上書きを繰り返すことになり非効率である．二つ目の問題は，メモリの保存中にリモートページングが発生した時に，チェックポイントに不整合が発生する可能性があることである．メモリがページインされた場合，タイミングによっては保存されない場合がある．また，メモリがページアウトされるとそのページはメインホストに存在しなくなるが，チェックポイントから当該ページのメモリ情報は削除されない．

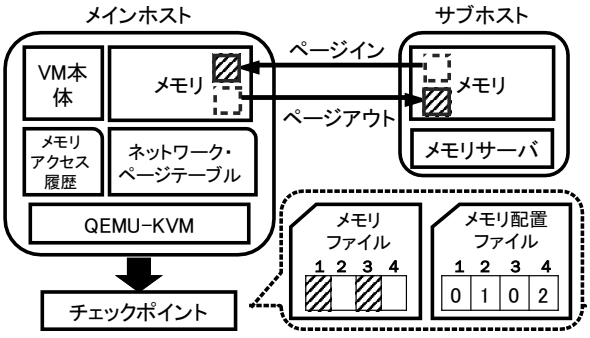

図 4 メインホストでのチェックポイント

これらの問題に対処するために，D-CRES では図 4 のようにメモリをメモリファイルと呼ばれるスパースファイルへ保存する．このメモリファイルのサイズは VM のメモリサイズと同一であり，メモリデータが保存されていないブロックは空（ホール）となる．メモリファイルのオフセットがページのアドレスに 1 対 1 に対応するため，更新されたページの上書き保存が容易に行える．ページインが行われた場合には，QEMU-KVM のダーティビットマップの対応するビットをセットし，適切なタイミングでメモリファイルの対応するブロックにメモリデータを保存する．ページアウトが行われた場合には，メモリファイルの対応するブロックをホールにすることで，ファイルからデータを削除する．このように，VM のメモリはメモリファイルに効率よく格納される．また，各ページが存在するホストの情報をページ配置ファイルと呼ばれるファイルに保存し，リモートページングのたびに対応する箇所を直接更新する．

保存すべきページが十分に少なくなったらサブホストに同期のための通知を送り，すべてのサブホストが同様の状態になるまで待つ．すべてのホストで同期がとれたら，処理途中のリモートページングの完了を待ってサブホストに最終処理のための通知を送り，VM を停止する．その後，残りのメモリとメモリアクセス履歴を保存し，QCOW2 の機能を用いてディスクイメージのスナップショットを高速に作成する．最後に，VM 本体の状態を保存し，サブホストからのチェックポイントの完了通知を待って VM を再開する．ライブチェックポイント中のリモートページングへの対処については現在，実装中である．

4.3 サブホストでのチェックポイント

メインホストからチェックポイント・コマンドを受信すると，サブホストのメモリサーバは VM のメモリを保存するためのチェックポイント・ファイルを作成する．そして，

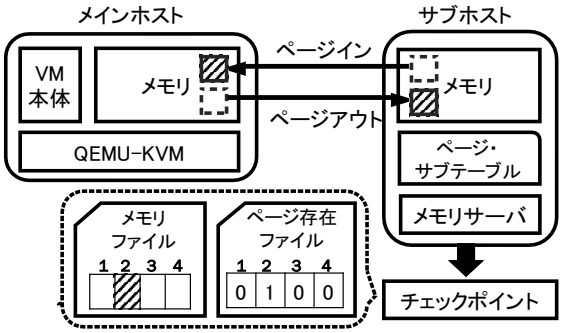

図 5　サブホストでのチェックポイント

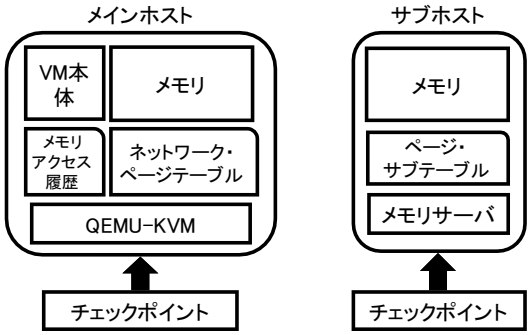

図 6　D-CRES のリストア

ページ・サブテーブルに基づいて，サブホストに存在するメモリについてそのアドレスとデータを保存する．すべてのメモリの保存が完了すると，メインホストに完了通知を送る．

ライブチェックポイント時にはメインホストでのチェックポイントと同様の問題が生じるため，図 5 のようにメモリデータをスパースファイルへ保存する．ページアウトが行われた場合には，メモリファイルの対応するブロックにメモリデータを保存する．ページインが行われた場合には，メモリファイルの対応するブロックをホールにすることでデータを削除する．メインホストと異なり，メモリの保存中に VM による直接のメモリ更新は行われないため，一旦，すべてのメモリを保存した後はリモートページングへの対処のみとなる．また，サブホストに存在するページの情報についてはページ存在ファイルと呼ばれるファイルを用意して保存および更新を行う．

メモリサーバはメインホストから同期のための通知を受け取ると，保存すべきメモリが十分に少なくなった時点で応答を返す．その後，最終処理のための通知を受け取ると，残りのメモリを保存する．すべてのメモリの保存が完了するとメインホストにチェックポイントの完了通知を返す．

4.4　分割メモリ VM のリストア

分割メモリ VM のリストアを行う際には，まず，取得したチェックポイントを復元先のホスト群に転送する．次に，メインホストの QEMU-KVM がリストアのための migrate-incoming コマンドを受け取ると，サブホストにリストア・コマンドを送信する．その後，図 6 のように，メモリデータが格納されたメモリファイルとページが存在するホストの情報が格納されたページ配置ファイルから，メインホストに存在していたメモリとネットワーク・ページテーブルを復元する．すべてのメモリの復元後にメモリアクセス履歴と VM 本体の状態を復元し，サブホストのリストア完了を待つ．すべてのサブホストからの完了通知を受け取ると，サブホストとの間でリモートページングのためのネットワーク接続を確立し，分割メモリ VM を再開する．

サブホストのメモリサーバはメインホストからリストア・コマンドを受信すると，メモリファイルとページの存在の有無を格納したページ存在ファイルからサブホストに存在していたメモリとページ・サブテーブルを復元する．すべてのメモリの復元が完了した後，メインホストに完了通知を送る．そして，メインホストとのネットワーク接続を確立すると，リモートページングの要求待ちに入る．

リストアの際に VM のメモリ再配置を行う場合は，メインホストとサブホストでそれぞれ保存したメモリファイル間でメモリデータの移動を行う．その情報をメインホストで保存したページ配置ファイルと，サブホストで保存したページ存在ファイルに反映する．メモリ再配置については現在，未実装である．

5.　実験

D-CRES を用いて分割メモリ VM のチェックポイント・リストアの性能向上について調べる実験を行った．メインホストとサブホストにはそれぞれ Intel Core i7-7700 の CPU，8GB のメモリ，ギガビットイーサネットを持つマシンを用いた．これらのマシンの OS として Linux 4.4.169 を動作させ，仮想化ソフトウェアとして QEMU-KVM 2.4.1 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た．

5.1　従来手法との比較

D-CRES を用いて分割メモリ VM のチェックポイント・リストアにかかる時間を測定した．比較のために，従来手法を用いて分割メモリ VM のチェックポイント・リストアにかかる時間も測定した．分割メモリ VM のライブチェックポイントには完全に対応できていないため，VM を停止させた状態でチェックポイントの取得を行った．また，1 台のホストで動作する通常の VM に対して従来手法を適用した場合の時間も測定した．分割メモリ VM のメモリはメインホストとサブホストに 512MB ずつ割り当てた．

チェックポイント・リストアにかかった時間を図 7 に示す．実験結果より，D-CRES のチェックポイントは従来手法で分割メモリ VM のチェックポイントを取得するよりも 51%高速であることが分かった．通常 VM のチェッ

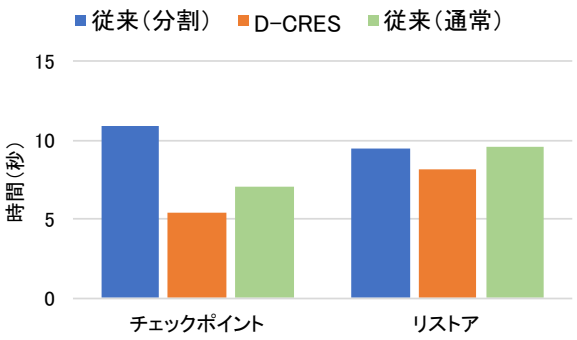

図 7　チェックポイント・リストア時間

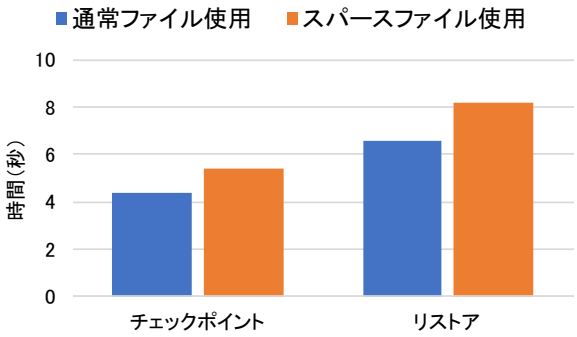

図 8　異なるチェックポイント・フォーマットを用いた場合のチェックポイント・リストア時間

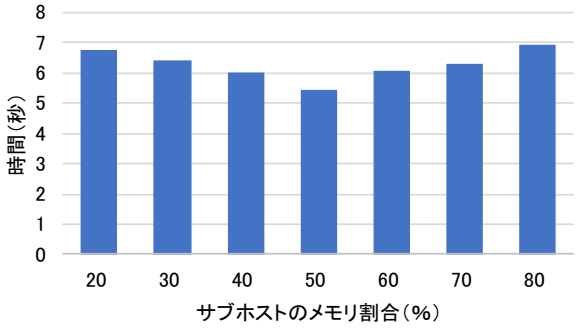

図 9　異なるメモリ分割比率でのチェックポイント時間

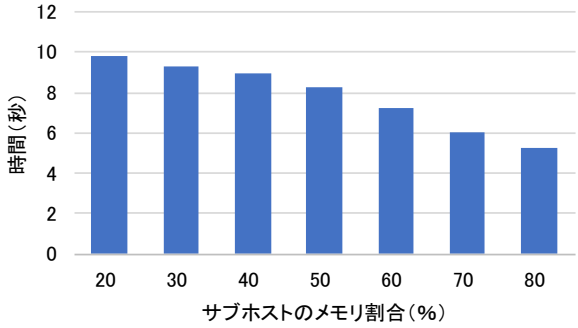

図 10　異なるメモリ分割比率でのリストア時間

クポイントを取得する場合と比べても 24%高速であった．D-CRES を用いると各ホストで保存するメモリ量は半分になったが，チェックポイント時間は通常 VM の半分にはならなかった．これはメインホストで VM 本体を保存する必要があることと，ライブチェックポイントに対応するためにチェックポイント・ファイルのフォーマットを変更したことが原因である．

一方，D-CRES のリストアについては，従来手法で VM を復元するよりも 14%高速に復元できることが分かった．従来手法では通常 VM としてしか復元できないため，通常 VM と分割メモリ VM のどちらのチェックポイントから復元してもほぼ同じリストア時間となった．D-CRES では各ホストで復元するメモリ量は半分になったが，リストア時間は半分にはならなかった．これも VM 本体の復元に時間がかかることと，チェックポイントのフォーマットを変更したことが原因である．

5.2　チェックポイント・フォーマットの影響

D-CRES において，異なるチェックポイント・フォーマットを用いた場合にチェックポイント・リストアにかかる時間を測定した．一つは従来手法と同様に通常のファイルを用いた場合であり，もう一つはライブチェックポイントのためにスパースファイルを用いた場合である．それぞれのチェックポイント・リストアの測定結果を図 8 に示す．実験結果より，スパースファイルを用いた場合は通常のファイルを用いた場合と比べてチェックポイント時間が 19%，リストア時間が 20%長くなった．これはスパースファイルを用いた場合には多くのシークが発生したためと考えられる．しかし，通常ファイルを用いた場合には，復元時に繰り返し同じデータを復元する可能性があり，その場合にはリストア時間が長くなると考えられる．

5.3　VM のメモリ分割比率の影響

メインホストとサブホストに割り当てるメモリ比率を変更した場合のチェックポイント・リストアにかかる時間を測定した．各比率での測定結果を図 9，図 10 に示す．実験結果より，チェックポイントにかかる時間が最短になるのはメインホストとサブホストに半分ずつ VM のメモリを割り当てた場合であった．また，メモリ割り当てがメインホストまたはサブホストに偏るほど，チェックポイントにかかる時間は増加することが分かった．一方のホストに割り当てるメモリ量が多ければ，チェックポイント取得の完了を待つためにもう一方のホストの待機時間が長くなるためである．

一方，サブホストに割り当てるメモリ量が増加するほどリストアにかかる時間は短くなることが分かった．メインホストはメモリの復元だけでなく VM 本体の復元も行い，この復元処理に時間がかかるためだと考えられる． VM のメモリサイズが小さいため VM 本体の復元にかかる時間の

割合が大きいが，メモリサイズを大きくするにつれて図 9 のチェックポイント時間と同様の傾向になると予測される．

6. 関連研究

IPmigrate[2] が提供している分割メモリ VM の置換マイグレーションは D-CRES のチェックポイント・リストアに類似している．置換マイグレーションはメインホストまたはサブホスト単位でライブマイグレーションを行う手法である．移送元ホストでの処理がチェックポイントに対応し，移送先ホストでの処理がリストアに対応する．置換マイグレーションのネットワーク送受信をディスク入出力に置き換えることでチェックポイント・リストアを実現できるが，単純な置き換えでは 4.2 節で指摘した問題が発生する．また，マイグレーション中のリモートページング発生時に移送先ホストで行うメモリの無効化をそのまま利用することはできない．

Emulab[3] は実験ネットワーク上に分散している複数 VM の状態をネットワークの状態と合わせて保存することができる．この手法は VM 内の OS カーネルを変更して，チェックポイント時に VM 内の実行と時間を一時停止する．一部の VM が先に停止されることによるパケット遅延や送信中パケットの発生を避けるために，時刻の同期によりすべての VM を同時に停止させる．また，ネットワーク上の遅延ノードのチェックポイントを取得することにより，ネットワーク遅延のために送信中となっているパケットが失われないようにする．D-CRES の場合には，ホスト間ではリモートページングが同期的に行われるだけであるため，最終的にリモートページングの同期をとることで送信中パケットが存在しない状態でチェックポイントを取得する．

VMCoupler[4] や D-MORE[5] は従属関係にある 2 つの VM の同期をとりながらマイグレーションを行う．VM-Coupler は監視対象 VM と IDS のオフロード先 VM を同時にマイグレーションし，監視を正常に継続できるように移送元の VM の停止・終了，移送先での VM の作成・再開の際に同期をとる．D-MORE は管理対象 VM と帯域外リモート管理用 VM を同時にマイグレーションし，管理を正常に継続できるようにより多くの箇所で同期をとる．D-CRES では VM 間ではなく，VM とメモリサーバ間で同期をとりながらチェックポイント・リストアを行うため，同期をとる箇所は少なくて済む．

Remus[6] はアクティブ VM とバックアップ VM の 2 台の VM を用意し，アクティブ VM の状態の差分をバックアップ VM に転送して同期を行う．ネットワーク送信やディスク書き込みは同期が完了するまでバッファリングされる．障害によりアクティブ VM が停止してしまっても，バックアップ VM に切り替えることで透過的に VM の実行を継続することができる．しかし，同期処理を高頻度で行うためオーバヘッドが大きい．Kemari [7] は VM からネットワーク送信やディスク書き込みを行う際にだけ同期をとることにより，同期の頻度を減らしている．COLO [8] はアクティブ VM が受信した要求パケットをバックアップ VM にも配送し，両方の VM からの応答が一致するまで待機させることで同期をとる．これらの手法は VM を 2 台用意する必要があることから，大容量メモリを持つ VM には適用するのが難しい場合が多い．

PMigrate[9] は余っている CPU や NIC を使って VM マイグレーションを並列化する．そのために，データ並列とパイプライン並列を用いる．並行実行される mmap と munmap 処理のスケーラビリティを向上させるために，Range Lock と呼ばれる手法が提案されている．D-CRES ではホスト単位でチェックポイントとリストアの並列化を行っているが，大容量メモリを持つ VM に対してはホスト内での並列化も併用することが効果的である．

7. まとめ

本稿では，複数ホストにまたがって動作する分割メモリ VM の柔軟で効率のよいチェックポイント・リストアを可能とするシステム D-CRES を提案した．D-CRES は分割メモリ VM が動作している各ホストで並列にメモリの保存を行うことで，チェックポイントのためのリモートページングを発生させないようにすることができる．また，スパースファイルを用いることでライブチェックポイント時にメモリデータを効率よく保存することができる．リストアを行う際には，複数のホストそれぞれで VM の状態を並列に復元することができる．その際に，VM のメモリの再配置を行うことでチェックポイント時と異なるホスト構成での復元を可能にする．D-CRES を KVM に実装し，チェックポイント・リストアの性能について調べる実験を行なった．その結果，従来手法を分割メモリ VM に適用した場合と比べて，チェックポイントは 51%，リストアは 14%の高速化ができることが分かった．一方，スパースファイルを用いることによるオーバヘッドが大きいことも分かった．

今後の課題はライブチェックポイントの実装を完成させることである．そのためには，チェックポイント中に VM が発生させるリモートページングに対処できるようにすることが必要である．その際に，スパースファイルを用いるとチェックポイント・リストアの性能が低下することが分かったため，他の方法も検討する．また，チェックポイントをさらに高速化するには，メモリの差分チェックポイントを実現する必要もある．分割メモリ VM ではリモートページングによってメモリの配置が変わるため，差分の検出に工夫が必要である．

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, In Proc. Int. Conf. Cloud Computing, pp.285-293, 2018.
[2] 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.
[3] A. Burtsev, P. Radhakrishnan, M. Hibler, and J. Lepreau.: Transparent Checkpoints of Closed Distributed Systems in Emulab. In Proc. EuroSys, 2009.
[4] K. Kourai and H. Utsunomiya: Synchronized Co-migration of Virtual Machines for IDS Offloading in Clouds, In Proc. Int. Conf. Cloud Computing Technology and Science, pp.120-129, 2013.
[5] Sho Kawahara and Kenichi Kourai: The Continuity of Out-of-band Remote Management across Virtual Machine Migration in Clouds, In Proc. Int. Conf. Utility and Cloud Computing, pp.176-185, 2014.
[6] B. Cully, G. Lefebvre, D. Meyer, M. Feeley, N. Hutchinson, and A. Warfield: Remus: High Availability via Asynchronous Virtual Machine Replication, In Proc. Symp. Networked Systems Design & Implementation, pp.161-174, 2008.
[7] 田村芳明, 柳澤佳里, 佐藤孝治, 盛合敏: Kemari: 仮想マシン間の同期による耐故障クラスタリング, 情報処理学会 ACS 論文誌, Vol.3, No.1, pp.13-24, 2010.
[8] Y. Dong, W. Ye, Y. Jiang, I. Pratt, S. Ma, J. Li, and H. Guan: COLO: COarse-grained LOck-stepping Virtual Machines for Non-stop Service, In Proc. Annual Symp. Cloud Computing, 2013.
[9] X. Song, J. Shi, R. Liu, J. Yang, and H. Chen: Parallelizing Live Migration of Virtual Machines, In Proc. Int. Conf. Virtual Execution Environments, pp.85-96, 2013.