分割メモリVMの効率的かつ柔軟なライブチェックポイント・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，分割マイグレーションは複数のホストに VM のメモリを分割して転送する．しかし，マイグレーション後の分割メモリ VM はホスト間でメモリデータを交換しながら実行されるため，ホストやネットワークの障害の影響を受けやすくなる．障害対策として VM のチェックポイント・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が交換されるためチェックポイント性能が低下する．また，従来のリストア手法では分割メモリ VM として復元することができない．本稿では，分割メモリ VM の効率的かつ柔軟なライブチェックポイント・リストアを可能とするシステム D-CRES を提案する．D-CRES は複数ホストで独立して分割メモリ VM の状態を保存することで，高速なライブチェックポイントを実現する．その際に，実行中の VM がホスト間で交換するメモリデータを考慮してメモリを過不足なく保存する．障害発生時には複数ホストを用いて並列に分割メモリ VM を復元する．D-CRES を KVM に実装し，分割メモリ VM に従来手法を適用した場合よりも最大 5.4 倍の性能向上を確認した．

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）が提供されるようになっている．例えば，Amazon EC2 では 24TB のメモリを持つ VM が提供されている [1]．このような VM はビッグデータの解析 [2][3] やインメモリ・データベース [4][5] 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．マイグレーションは VM 全体のメモリを転送するため，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運用の柔軟性の面で望ましくない．

そこで，大容量メモリを持つ VM のマイグレーションを容易にするために，分割マイグレーション [6] が提案されている．分割マイグレーションは VM のメモリを分割して複数のホストに転送する．仮想 CPU や仮想デバイス等の 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 本体が動作し，サブホストはメインホストに VM のメモリの一部を提供する．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．分割メモリ VM はホスト間でリモートページングと呼ばれる機構を用いてメモリデータの交換を行う．VM 本体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，最もアクセスされそうにないメモリのデータをサブホストに転送（ページアウト）する．

分割メモリ VM は複数のホスト間で通信を必要とするため，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，障害対策として VM のチェックポイント・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，チェックポイント性能が低下することである．これは，サブホスト上のメモリを一旦，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

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

ないことである．そのため，リストア時には常に十分な空きメモリを持ったホストが必要となる．

これらの問題を解決するために，本稿では，分割メモリ VM の効率的かつ柔軟なライブチェックポイント・リストアを可能とするシステム D-CRES を提案する．D-CRES のライブチェックポイントは分割メモリ VM のメモリを各ホストで独立して保存する．これにより，サブホストとメインホスト間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，すべての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．D-CRES のリストアは複数のホストに分割された状態での分割メモリ VM の復元を可能にする．すべてのホストで並列にメモリを復元することで高速なリストアを実現する．また，リストアを行う前に VM のメモリ分割を変更することで，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 のライブチェックポイント・リストアを実現した．D-CRES はチェックポイントの間，実行中の VM が発生させるリモートページングを考慮してメモリを過不足なく保存する．リモートページングによって移動したメモリは移動先のホストで必ず保存し，移動して存在しなくなったメモリは移動元のホストのチェックポイントから削除する．D-CRES はディスクのアクセス効率を向上させるために，メモリデータをファイルに追記して保存する．しかし，チェックポイント中に更新されるメモリデータ量に応じてファイルサイズが肥大化し，更新される前の古いデータも保存された順に復元しなければならないためリストア性能が低下する．そこで，チェックポイント完了後に追記保存されたメモリデータを変換することで，チェックポイントのファイルサイズを最小化し，リストア性能を向上させる．D-CRES を用いて実験を行った結果，分割メモリ VM に従来手法を適用した場合よりも，最大 5.4 倍の性能向上を確認した．

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 の効率的かつ柔軟なライブチェックポイント・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. 分割メモリ VM の障害対策

VM はホストのメンテナンスや負荷分散の際に別のホストにマイグレーションされるが，その際には，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を常に維持しておくコストや運用の柔軟性の低下が問題となる．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 VM の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仮想 CPU や仮想デバイスなどの VM 本体の状態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アクセス履歴に基づいて予測する．

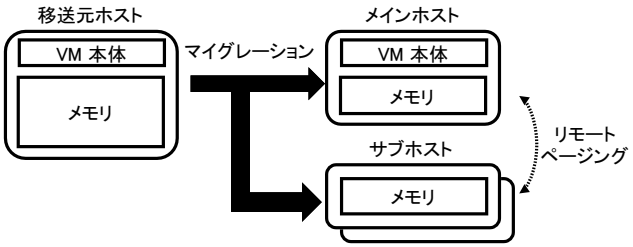

図 1　分割マイグレーション

分割マイグレーション後はメインホスト上で VM 本体が動作し，サブホストはメインホストへ VM のメモリの一部を提供する．このように複数の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，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 本体がサブホストに存在するメモリを必要とした場合には，当該メモリをネットワーク経由で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，最もアクセスされそうにない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，メインホストとサブホスト間で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，障害から復旧するまでは VM の実行を行うことができなくなる．

従来，VM の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 VM の状態をチェックポイントと呼ばれるバックアップファイルに保存する．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，別のホスト上に VM を作成して最新のチェックポイントから VM の状態を復元する．一から新しい VM を起動する場合と比べて，復旧にかかる時間を短縮し，障害によるデータの損失を最小限に抑えることができる．

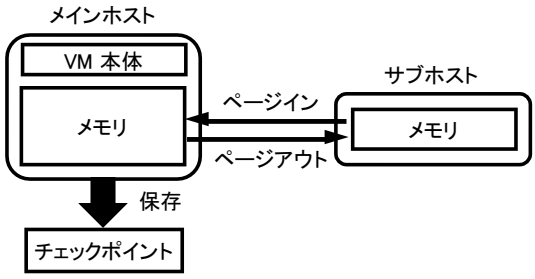

図 2 従来手法を用いた分割メモリ VM のチェックポイント

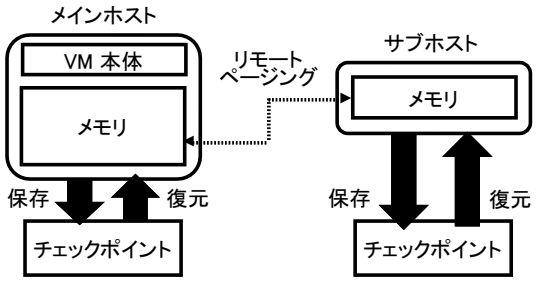

図 3 D-CRES のライブチェックポイント・リストア

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，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，チェックポイント性能が大幅に低下することである． 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旦，メインホストにページインされてからメインホスト上で保存される．これはチェックポイントを実行している 1 台のホスト上でのみ VM の状態が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，ページアウトも行われる．ページアウトしたメモリをその後で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，従来手法では分割メモリ VM として VM を復元することができないことである．従来のチェックポイントは一つのチェックポイント・ファイルに VM 全体のメモリデータを保存しており，分割メモリ VM と通常の VM を区別することができない．そのため，リストア時には常に一つのホスト上で動作する通常の VM として復元が行われる．その際には VM のメモリサイズ以上の空きメモリを持ったホストが必要となるが，大容量メモリを持つ VM の場合はそのようなホストを用意するのが困難な場合がある．適切なホストが用意できなければ，用意できるまで復元を行うことができない．

3. D-CRES

本稿では，分割メモリ VM の効率的かつ柔軟なライブチェックポイント・リストアを可能とするシステム D-CRES を提案する．D-CRES のライブチェックポイントは，図 3 のように各ホストで独立して 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チェックポイントに起因するリモートページングを発生させないようにする．また，メモリの保存を並列に行うことにより，分割メモリ VM の高速なチェックポイントを実現する．障害が発生した際には，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．これによりリストアも高速化することができる．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 は分割メモリ VM のチェックポイントを取得する際，メインホスト上にある VM のメモリをメインホストのチェックポイントとして保存する．また，仮想 CPU や仮想デバイスなどの VM 本体の状態も保存する．仮想ディスクのスナップショットも通常の 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ある 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冗長性のあるネットワークストレージに保存する．

D-CRES は分割メモリ VM をほとんど停止させずにチェックポイントを取得することができる．そのために，VM を動かしたまま VM のメモリを保存し，保存中に更新されたメモリについては追加で保存する．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．移動したメモリは移動先のホストで必ず保存し，移動して存在しなくなったメモリは移動元のホストのチェックポイントから削除する．更新されたメモリが十分に少なくなったら VM を停止し，残りのメモリとその他の状態を保存する．

障害発生時，D-CRES は合計で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，メインホストに VM を作成して，最新のチェックポイントを用いて分割メモリ VM を復元する．まず，各ホストで VM のメモリの一部をそれぞれ復元する．次に，メインホストでは VM 本体と仮想ディスクの状態を復元する．すべてのホストで状態の復元が完了すると，メインホストとサブ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 台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

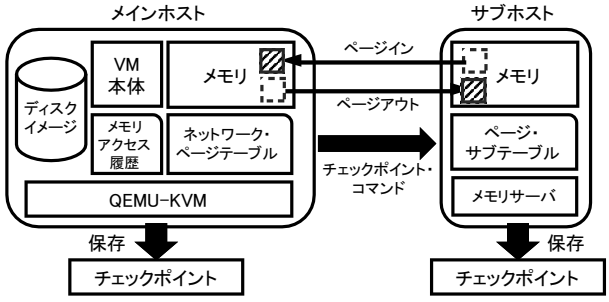

図 4 分割メモリ VM のライブチェックポイント

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 のメモリ管理

分割メモリ VM は各ホストで専用の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，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・ページテーブルはメモリページ番号からホスト ID を検索するための表である．QEMU-KVM 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，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．メモリサーバはこのテーブルを参照することでそのサブホストに存在しているメモリページの情報を取得する．

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，QEMU-KVM がネットワーク・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 QEMU-KVM に転送する．その後，メインホストで最もアクセスされそうにないページを探し，メモリサーバ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 メインホストでのライブチェックポイント

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，図 4 のようにすべての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データの保存を行う．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，メモリ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．一通りすべてのメモリを保存し終わった後は，実行中の VM によって更新されたメモリの保存を繰り返す．

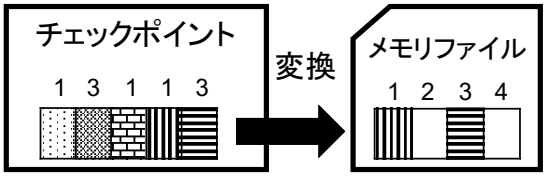

図 5 チェックポイントのメモリデータ変換

既存の QEMU-KVM は通常の VM のライブチェックポイントをサポートしているが，その実装を分割メモリ VM に用いると問題が生じる．メモリの保存中に VM がリモートページングを発生させた場合に，保存したメモリデータに不整合が発生する可能性があるためである．メインホストにメモリがページインされた場合，QEMU-KVM のチェックポイント機構がそのことに気づかず，タイミングによってはメインホストのチェックポイント・ファイルに正しく保存されないことがある．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，メインホストのチェックポイント・ファイルから当該ページのメモリ情報は削除されない．そのため，リストア時に複数のホストで重複してメモリが復元されてしまう．

この問題に対処するために，D-CRES ではページインが行われた場合，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する．QEMU-KVM はダーティビットマップにビットがセットされているメモリを後で再度，保存するため，ページインされたメモリが必ず保存されることを保証することができる．ページインされたメモリについて既に最新のデータが保存されている場合もあるが，そのような場合に不要な保存を行わないようにするのは今後の課題である．一方，ページアウトされたメモリについては，ネットワーク・ページテーブルのエントリの削除のみを行う．このテーブルにエントリがない場合にはチェックポイント・ファイル中にメモリデータがあったとしても無視する．

既存の QEMU-KVM の実装を大容量メモリを持つ VM に適用すると性能面での問題も生じる．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ためである．大容量メモリを持つ VM のライブチェックポイント中には大量のメモリが更新される

ため，そのデータを追記するとチェックポイント・ファイルが肥大化する．さらに，リストア時には保存された順番に，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することを繰り返すことになり非効率である．

この問題に対処するために，D-CRES はチェックポイント完了後に図 5 のように，チェックポイント・ファイルに追記されたメモリデータを別の形式のメモリファイルに変換する．メモリファイルはオフセットが VM のメモリアドレスに 1 対 1 に対応しており，最新のメモリデータだけが保持される．そのため，リストア時に古いメモリデータを復元する必要はなくなる．また，メモリファイルはスパースファイルとなっており，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．そのため，すべてのホストのメモリファイルの合計サイズは VM のメモリサイズに抑えられる．チェックポイント時に直接，メモリファイルの形式で保存する方法も考えられるが，シークのオーバヘッドが大きくなる [7] ためチェックポイント時にはより高速な追記での保存を行う．

メインホストにおいて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，残りのメモリとネットワーク・ページテーブルを保存し，仮想 CPU や仮想デバイスなどの VM 本体の状態を保存する．最後に，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し，すべての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．

4.3 サブホストでのライブチェックポイント

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，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．一通りすべてのメモリを保存し終わった後は，実行中の VM が保存中に発生させたページアウトによってメモリサーバに送られてきたメモリの保存を繰り返す．メインホストとは異なり，ページアウトされたメモリ以外は更新されない．ページインされてサブホストに存在しなくなったメモリについては，ページ・サブテーブルのエントリのみ削除する．

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，残りのメモリとページ・サブテーブルを保存する．すべての状態の保存が完了するとメインホストに完了通知を返す．チェックポイント完了後に，D-CRES はメインホストと同様に，チェックポイント・ファイルに追記されたメモリデータをメモリファイルに変換する．

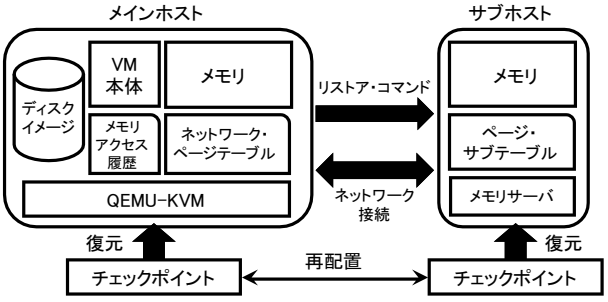

図 6　D-CRES のリストア

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 がリストアのために使われる migrate-incoming コマンドを受け取ると，サブホストのメモリサーバにリストア・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チェックポイント・ファイルに保存されたネットワーク・ページテーブルから，メインホストに存在していたメモリを復元する．すべてのメモリの復元後にメモリアクセス履歴と仮想 CPU や仮想デバイスなどの VM 本体の状態，仮想ディスクの状態を復元し，サブホストでのリストア完了を待つ．すべてのメモリサーバ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，メモリファイルとチェックポイント・ファイルに保存されたページ・サブテーブルから，サブホストに存在していたメモリを復元する．すべてのメモリの復元が完了した後，QEMU-KVM 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．

D-CRES は新しく利用可能なホスト群の空きメモリ量に合わせて分割メモリ VM のメモリを再配置する．そのために，D-CRES はリストアを行う前にメモリファイル間でメモリデータの移動を行う．その際に，チェックポイント・ファイルに保存されたメモリアクセス履歴を参照して，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．同時にチェックポイント・ファイル内のネットワーク・ページテーブルとページ・サブテーブルも書き換える．現在，すべてのサブホストからメインホストへメモリ全体を再配置する機能だけが実装されている．

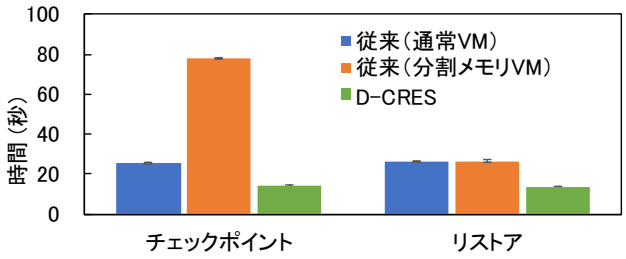

図 7　チェックポイント・リストア時間

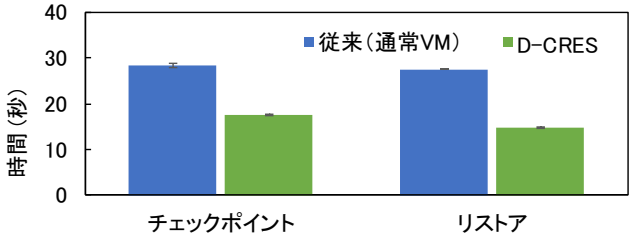

図 8　ライブチェックポイント・リストア時間

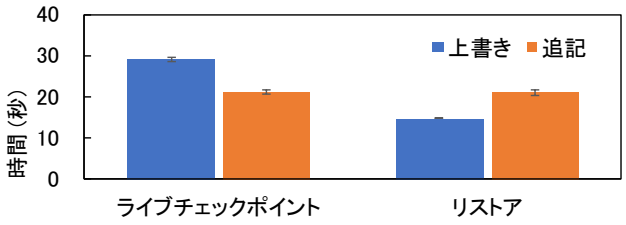

図 9　チェックポイント・ファイルフォーマットの性能への影響

5.　実験

D-CRES を用いて分割メモリ VM のライブチェックポイント・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，4GB 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とリストアにかかる時間を測定した．比較のために，分割メモリ VM に従来手法を適用した場合についても測定した．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 2GB ずつ割り当てた．

まず，VM によるメモリ更新の影響を除外するために VM を停止させた状態でチェックポイントを取得した．チェックポイントと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分割メモリ VM に従来手法を適用するよりも 5.4 倍高速であることが分かった．通常 VM のチェックポイントを取得する場合と比べても 77%高速であった．しかし，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 の場合の半分にはならなかった．これはメインホストにおいてメモリ以外の状態を保存する必要があるためである．

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 89%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同じリストア時間となった．チェックポイントと同様に D-CRES を用いると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．これもメモリ以外の状態を復元するのに時間がかかるためである．

次に，D-CRES を用いて動作中の分割メモリ VM のライブチェックポイントとリストアにかかる時間を測定した．比較のために，通常 VM に従来手法を適用した場合についても測定した．この実験では，実行中の VM によって生じるリモートページングの影響を調べるために，VM 内でメモリを 2GB 確保し書き換え続けるプログラムを実行した．ライブチェックポイントとリストアにかかった時間を図 8 に示す．実験結果より，D-CRES のライブチェックポイントは通常 VM に対して従来手法を適用するよりも 61%高速であった．チェックポイント・ファイルのメモリデータ部分をメモリファイルに変換するのに 40 秒かかったが，この処理はチェックポイントとは独立に行うことができる．一方，D-CRES のリストアは従来手法を通常 VM に適用するよりも 86%高速に復元できることがわかった．

5.2　チェックポイント・ファイルの変換の効果

D-CRES はライブチェックポイント時にはメモリデータを追記保存し，メモリファイルに変換してリストアに用いる．この変換の効果を確認するために，まずチェックポイント・ファイルの合計サイズを調べた．メモリの更新を行いつつリモートページングの影響を排除するために，VM 内でメモリ 1GB を確保し書き換え続けるプログラムを実行した．実験の結果，チェックポイント・ファイルの合計サイズは 5.3GB となり，VM のメモリサイズの 4GB よりかなり大きくなった．チェックポイント・ファイルをメモリファイルに変換すると，ファイルサイズの合計を 4GB に削減することができた．

次に，追記保存したチェックポイント・ファイルを変換せず，そのまま用いて VM を復元した場合のリストア時間を測定した．また，メモリファイルに直接，上書き保存した場合のチェックポイント時間も測定した．D-CRES の

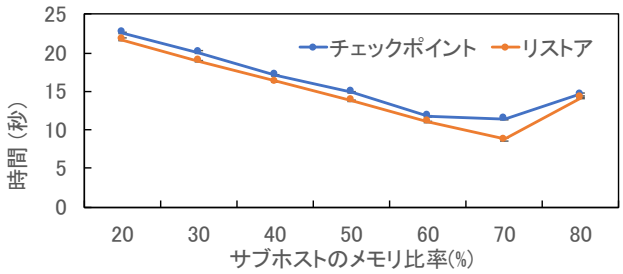

図 10 異なるメモリ分割比率でのチェックポイント・リストア時間

チェックポイント・リストア時間との比較を図 9 に示す．追記保存したファイルを用いて復元を行うとリストアが 42%遅くなった．これは保存された順に古いメモリデータも復元したためである．一方，メモリファイルに直接保存するとライブチェックポイントが 37%遅くなった．これはメモリファイルの上書き時にシークが大量に発生したためである．

5.3 VM のメモリ分割比率の影響

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とリストアにかかる時間を測定した．図 10 はメモリ比率ごとにかかった時間を示す．サブホストに VM のメモリの 70%を割り当てた時，チェックポイントとリストアともに所用時間が最小であった．これはメインホストの QEMU-KVM はメモリだけでなく VM 本体の状態や仮想ディスクのスナップショットを保存，復元するためである．

6. 関連研究

VM の高可用性を実現するために，Remus[8] 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．Kemari [9] 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 [10] 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．

Emulab の分散チェックポイント [11] は実験ネットワーク上の複数 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ことができる．

VM のマイグレーションはチェックポイント・リストアと似ている．マイグレーションでは，移送元ホストで VM のチェックポイントを取得して移送先ホストに転送した後，チェックポイントから VM を復元する．特に，分割メモリ VM の置換マイグレーション [12] の実装は 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．しかし，D-CRES はメモリファイルを用いたリストアにおいて大きな変更が必要であった．

VMCoupler[13] や D-MORE[14] 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 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．

PMigrate[15] 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と効果的である．

7. まとめ

本稿では，分割メモリ VM の効率的かつ柔軟なライブチェックポイント・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，チェックポイントに起因するリモートページングを発生させないようにする．また，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を考慮して動作中の VM のメモリを過不

足なく保存する．チェックポイント完了後にチェックポイント・ファイルを変換することにより，ファイルサイズを最小化し，効率のよいリストアを実現する．リストアを行う際には，複数のホストそれぞれで並列に分割メモリ VM を復元する．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，ライブチェックポイント・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，最大 5.4 倍の性能向上を確認した．

今後の課題は，大容量メモリを持つ VM を用いてライブチェックポイント・リストアにかかる時間を測定することである．現在，240GB の VM を用いて実験を行う準備をしているところである．また，チェックポイントのオーバヘッドを削減するために分割メモリ VM の差分チェックポイントを実装することである．分割メモリ VM のメモリはリモートページングによって移動するため，複数ホストにまたがってメモリの差分を効率よく検出する必要がある．

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] Amazon Web Services, Inc. Amazon EC2 High Memory Instances. https://aws.amazon.com/ec2/instance-types/high-memory/, 2019.
[2] Apache Software Foundation. Apache Spark - Lightning-Fast Cluster Computing. http://spark.apache.org/.
[3] Facebook, Inc. Presto: Distributed SQL Query Engine for Big Data. https://prestodb.io/.
[4] SAP SE. What is SAP HANA? An Unrivaled Data Platform for the Digital Age. https://www.sap.com/products/hana.html.
[5] Microsoft Corporation. SQL Server 2017 on Windows and Linux. https://www.microsoft.com/en-us/sql-server/sql-server-2017.
[6] 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, pages 285-293, 2018.
[7] 村田時人，光来健一: 複数ホストで動作する分割メモリ VM のチェックポイント・リストア, 情報処理学会研究報告，Vol.2019-OS-147, 2019.
[8] 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, pages 161-174, 2008.
[9] Y. Tamura. Kemari: Virtual Machine Synchronization for Fault Tolerance using DomT. In Xen Summit Boston 2008, 2008.
[10] Y. Dong, W. Ye, Y. Jiang, I. Pratt, S. Ma, J. Li, and H. Guan. COLO: COarse-grained LOck-stepping Virtual Machines for Nonstop Service. In Proc. Annual Symp. Cloud Computing, 2013.
[11] A. Burtsev, P. Radhakrishnan, M. Hibler, and J. Lepreau. Transparent Checkpoints of Closed Distributed Systems in Emulab. In Proc. European Conf. Computer Systems, 2009.
[12] T. Kashiwagi and K. Kourai. Flexible and Efficient Partial Migration of Split-memory VMs. In Proc. Int. Conf. Cloud Computing, pages 248-257, 2020.
[13] K. Kourai and H. Utsunomiya. Synchronized Comigration of Virtual Machines for IDS Offloading in Clouds. In Proc. Int. Conf. Cloud Computing Technology and Science, pages 120-129, 2013.
[14] S. Kawahara and K. Kourai. The Continuity of Out-of-band Remote Management across Virtual Machine Migration in Clouds. In Proc. Int. Conf. Utility and Cloud Computing, pages 176-185, 2014.
[15] X. Song, J. Shi, R. Liu, J.Yang, and H. Chen. Parallelizing Live Migration of Virtual Machines. In Proc. Int. Conf. Virtual Execution Environments, pages 85-96, 2013.