仮想計算機を用いたサーバ統合における高速なリブートリカバリ

光来 健一　　千葉 滋

東京工業大学 情報理工学研究科 数理・計算科学専攻

要 旨

仮想計算機（VM）を用いたサーバ統合を行う場合，仮想計算機モニタ（VMM）や VM の再起動後のリブートリカバリに時間がかかってしまう．VMM の再起動時にはすべての VM を起動し直す必要があり，VM の再起動時には OS のファイルキャッシュがすべて失われる．これらの問題を解決するために，本稿では Warm-VM Reboot と Warm-cache Reboot からなる高速なリブートリカバリを提案する．Warm-VM Reboot は計算機の再起動を伴う VMM の再起動時に VM のメモリイメージをディスクに保存することなく高速に VM をサスペンドし，再起動後に高速にレジュームする．Warm-cache Reboot は VM の再起動後に VM 内の OS が再起動前に保持していたファイルキャッシュを再利用することによりキャッシュミスを軽減する．我々はこれらの機構を Xen およびその上で動く Linux に実装し，リブートリカバリが高速化されることを実験により確認した．

1　はじめに

複数の計算機で運用されていたサーバ群がサーバ統合（server consolidation）により 1 つの計算機で運用されるようになっている．サーバ統合はサーバ間で計算機のリソースを共有することによる有効活用を可能にし，TCO 削減に役立つ．一方，Intel の VT（Virtualization Technology）や AMD の Pacifica のような CPU による仮想化のサポートや，PAE（Physical Address Extension）や EM64T，AMD64 による 4 GB を超える物理メモリのサポートにより，サーバ統合に仮想計算機（VM）を用いるのも現実的になってきている．VM を用いることによる性能低下が改善され，VM に多くのメモリを割り当てることが可能になるためである．サーバ統合に VM を用いる利点としては，統合されたそれぞれのサーバを論理的に分離することによる安全性の確保や，それぞれのサーバが用いていた別々の OS を動かせることなどが挙げられる．加えて，VM 内の OS がクラッシュした場合でもリモートログインして再起動できるため，管理を容易にすることができる．

しかし，VM を用いてサーバ統合を行うと，仮想計算機モニタ（VMM）や VM を再起動した時に，再起動前のサービス品質を回復するためのリブートリカバリに時間がかかってしまう．第 1 に，VMM を再起動する時には計算機の再起動を伴なうため，再起動後にすべての VM を起動し直す必要がある．VM を起動した時には OS の起動およびサーバの起動が行われるため，VM の数に比例して時間がかかる．第 2 に，VM を再起動すると VM 内の OS が保持していたファイルキャッシュが失なわれ，再起動後にはキャッシュミスを頻発するようになる．最近では VM に多くのメモリを割り当てることができるようになっており，再起動前の十分にキャッシュされている状態に戻るまでには時間がかかる．そのため，サービスが再開されてからしばらくは最大性能を発揮できない可能性がある．

これらの問題を解決して再起動の影響を最小限に抑えるために，本稿では Warm-VM Reboot と Warm-cache Reboot の 2 つの機構からなる高速なリブートリカバリを提案する．Warm-VM Reboot は VMM を再起動する時に VM のメモリイメージをディスクに保存することなく高速に VM をサスペンドし，計算機の再起動を伴う VMM の再起動後に VM を高速にレジュームすることを可能にする．この機構により，VMM の再起動後に VM 内で提供されているサービスをすぐに再開でき，サーバのダウンタイムを短くすることができる．一方，Warm-cache Reboot は VM を再起動した後で VM 内の OS が再起動前に保持していたファイルキャッシュを再利用することを可能にし，再起動後のキャッシュミスを軽減する．この機構により，再起動直後からサーバの最大性能を引き出せるようになる．

我々はこれらの機構を Xen VMM およびその上の VM 内で動作する Linux に実装し，再起動後のリブートリカバリをどの程度高速化できているかを調べる実験を行った．この実験の結果，Warm-VM Reboot を用いることにより，我々の実験環境ではサーバのダウンタイムを約半分に減らせることが分かった．また，Warm-cache Reboot を用いることにより再起動直後のサーバの性能低下を 8%以下に抑えられることが分かった．

2 再起動の影響

サーバにおける OS やサーバプログラムなどのソフトウェアの再起動は提供するサービスの品質をそこなうので可能な限り避けるべきであるが，完全に避けるのは難しい．例えば，ソフトウェアをアップデートした後には再起動が必要になる場合が多い．また，ソフトウェアを長期間動かすことによりメモリリークなどが原因で不安定になる場合には定期的に再起動する必要がある．さらには，ソフトウェア自体のバグによってクラッシュして再起動が必要になる場合もある．このような場合に部分的な再起動を行うことにより高速化する研究も行われている [7, 3] が，あらゆる場合に完全な再起動を避けられるわけではない．

問題の箇所の再起動は避けられないとしても，その再起動の影響は最小限に抑えるべきである．しかし，VM を用いてサーバ統合を行う場合，VMM や VM を再起動した後で再起動前のサービス品質を回復するためのリブートリカバリに時間がかかってしまう．第 1 に，VMM を再起動する時には，一旦すべての VM をシャットダウンし，VMM の再起動後に再度起動し直す必要がある．Xen [6] や VMware ESX Server [8] のように VMM がハードウェア上に直接置かれる場合，VMM の再起動は計算機の再起動を伴うためである．VM を起動し直した時には OS の起動およびサーバの起動が行われるため，VM の数に比例して時間がかかる．その結果，VM 内で提供されているサービスのダウンタイムが増加する．さらに，Xen の場合は VM の管理などを行うための制御用 VM が存在し，制御用 VM を再起動した場合も VMM が再起動される．制御用 VM 内では通常の OS が動作し，シンプルな VMM より再起動を必要とする頻度は高いため，VMM が再起動される頻度も高くなる．

第 2 に，VM を再起動すると VM 内の OS が保持していたファイルキャッシュがすべて失われ，再起動後にキャッシュミスを頻発するようになる．VM の起動時には SCSI などのハードウェアチェックが行われず，VM を使わない場合と比べて高速に再起動できるため，再起動にかかる時間に対してキャッシュミスの影響が相対的に大きくなる．さらに，最近のサーバ計算機は 4 GB を超える物理メモリを搭載できるようになってきているため，必要ならば 1 つの VM に数 GB のメモリを割り当てることも可能である．VM に割り当てられたメモリの内，OS によって使われていないメモリはファイルキャッシュとして利用されるため，再起動前の十分にキャッシュされている状態に戻るまでには時間がかかる．その間はファイルアクセスの性能が低下することになり，サーバの最大性能を発揮することができない．また，キャッシュミスによるディスクアクセスが他の VM のディスクアクセスと競合する場合，さらに性能低下を引き起こす可能性もある．

VMM の再起動の影響を抑えるために，VMM を再起動する前に VM をサスペンドしてメモリイメージ等をディスクに保存しておき，VMM の再起動後に VM をレジュームする方法がある．VM をレジュームした後はサスペンドした地点から実行を再開できるため，OS やサーバをシャットダウンして再び起動するオーバヘッドを回避できる．しかし，VM のメモリイメージの保存には VM に割り当てられているメモリサイズに比例して時間がかかる．VM のメモリイメージを保存している間 VM は停止させられるので，その時間はダウンタイムとなる．また，VM のメモリイメージ等をディスクに保存する必要があるため，ディスクへの保存を行うのに必要な制御用 VM や VMM がクラッシュした時にはこの方法を用いることはできない．

VM の再起動の影響を抑えるために，Windows XP のように OS の起動時にファイルの先読みを行う方法が考えられる．Windows XP ではアプリケーショ ンによってよく使われるファイルの記録を取っておき，その統計情報に基づいて起動時にアクセスを行いキャッシュしておく．OS やサーバの起動とオーバラップして先読みを行うことにより，キャッシュミスを減らすことができるが，ディスクやメモリアクセスの競合により，先読みを行うこと自体がオーバヘッドになり，起動時間を長引かせる可能性がある．また，先読みを行ってもディスクアクセス自体は発生するため，他の VM への影響は避けられない．

3 高速なリブートリカバリ

問題の箇所を再起動した後でその再起動の影響を最小限に抑えるために，再起動前のメモリの内容を再起動後に利用することによる高速なリブートリカバリを提案する．このリブートリカバリは VMM と VM のそれぞれのリブートリカバリを高速化するために，Warm-VM Reboot と Warm-cache Reboot の 2 つの機構を提供する．

Warm-VM Reboot は VMM の再起動時に，VM のメモリイメージをディスクに保存することなく高速に VM をサスペンドし，計算機の再起動を伴う VMM の再起動後に高速にレジュームすることを可能にする．この機構は VM に対して ACPI [1] の S3 状態（Suspend To RAM）を実現するものである．VM をサスペンドする

時にはVMのメモリイメージをコピーすることなくそのままメモリ上で凍結し，VMの状態もメモリ上に保存する．そして，BIOSのメモリチェック省略機能を利用することで，計算機の再起動を通してメモリの内容を保持し続ける（warm reboot）．この機能はクイックブート機能の一部として最近の多くのPCのBIOSで提供されている．VMをレジュームする時には，メモリ上のVMのメモリイメージの凍結を解除し，保存しておいたVMの状態を復元するだけでよい．

Warm-cache RebootはVMを再起動した後で，VM内のOSが再起動前に保持していたディスクキャッシュを再利用することにより，キャッシュミスを軽減することを可能にする．再起動を通してVMに割り当てられているメモリの内容を保持させることで，VM内のOSが再起動前のファイルキャッシュが置かれているメモリにアクセスできるようにする．VM内のOSが再起動前のファイルキャッシュを管理し，対応するファイルがアクセスされた時にそのファイルキャッシュを再利用する．

VMMを再起動する時にはWarm-VM Rebootを用いることにより，制御用VMを除くすべてのVMを高速にレジュームし，ダウンタイムを減少させることができる．レジュームしたVMにはサスペンド時のファイルキャッシュが残されているため，キャッシュを失うことによる性能低下もない．ただし，制御用VMはVMMと密接に関連しているためサスペンドできず，VMMの再起動後に起動し直すことになる．一方，VMを再起動する時にはWarm-cache Rebootを行うことにより，再起動の前後でファイルのアクセスパターンが大きく変わらない限り，キャッシュミスによる性能低下を防ぐことができる．それぞれのVMでは再起動の前後で同じサービスが動くので， アクセスパターンは類似することが多いと考えられる．

4 実装

我々はVMMとしてXen，VM内で動作させるOSとしてLinuxをベースにしてWarm-VM RebootとWarm-cache Rebootを実装した．i386アーキテクチャへの実装は完了しており，x86_64アーキテクチャへの実装は途中まで完了している．

4.1 Xenの概要

Xenにおいて VMはドメインと呼ばれ，図1のように制御用のドメイン0とそれ以外のドメインUが存在

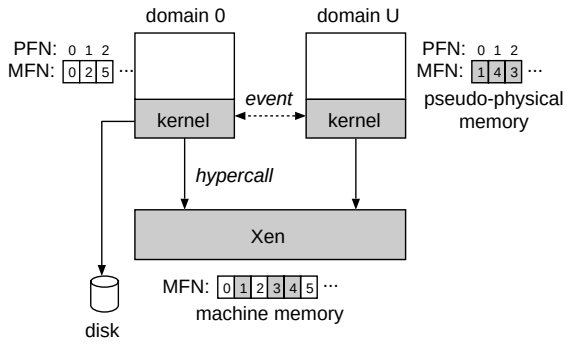

図 1: Xenのアーキテクチャ

する．ドメインがXenの機能を明示的に利用する時には，ハイパーコールと呼ばれるシステムコールに似た機構が用いられる．一方，ドメイン0とドメインUの間の通信にはイベント通知機構が用いられる．ドメインUがディスクやネットワークにアクセスする時には，このイベント通知機構を用いて，ドメイン0を介して物理的なディスクやネットワークにアクセスを行う．ドメイン0はXen本体と密接に結びついているため，ドメイン0を再起動するとXenも再起動される．

Xenはドメインに仮想的な物理メモリを提供するために，マシンメモリと疑似物理メモリという概念を導入している．マシンメモリは計算機に搭載されている物理メモリを指し，4 KBごとの連続した物理メモリページに0から順番にマシンページフレーム番号（MFN）が割り当てられる．一方，疑似物理メモリはドメインに割り当てられる物理メモリを指し，割り当てられた物理メモリページに対して0から順番に疑似物理ページフレーム番号（PFN）が割り当てられる．疑似物理メモリはドメインに連続していない物理メモリページを割り当てた場合でも，ドメインが連続したメモリページとして扱えるようにするために用いられる．

4.2 メモリマッピングの管理

Xenを最初に起動した時には，ドメインに割り当てられるマシンメモリを管理するためのP2Mマッピングテーブルを作成する．このテーブルは各ドメインに対してPFNからMFNへのマッピングを管理する．Xenは通常MFNからPFNへのマッピングのみを管理しており，オプションでPFNからMFNへのマッピングを管理させることもできる．しかし，Xenの管理するPFNからMFNへのマッピングテーブルは扱いにくいため，独自のP2Mマッピングテーブルを実装した．このテーブルのサイズはマシンが搭載している物理メモリサイズに依存し，物理メモリ1 GBに対して32ビッ

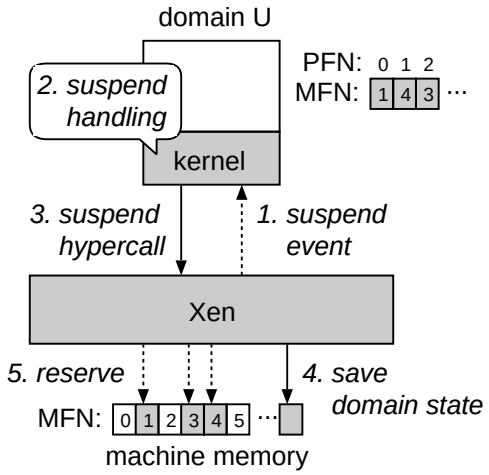

図 2: ドメイン U のサスペンド処理

ト環境では 1 MB，64 ビット環境では 2 MB となる．

ドメインを作成した時には，そのドメインに割り当てられたマシンページフレームを P2M マッピングテーブルに登録する．ドメインは最初に割り当てられたマシンメモリを動的に解放または再確保することがあるため，その度に P2M マッピングテーブルを更新する．ドメインを破棄する時には，ドメインによって使われていたマシンメモリが Xen によって再利用されるのを防ぐために，解放せずに予約しておく．ドメインの疑似物理メモリを後で再構築できるように，P2M マッピングテーブルからも登録を削除しない．

4.3 Warm-VM Reboot

Xen ではドメイン U をサスペンドする時に，ドメイン U 内の OS にサスペンド処理を行わせる必要がある．通常，ドメイン 0 からコマンドを実行してドメイン U のサスペンドを行うため，イベント通知機構を用いてドメイン 0 からドメイン U にサスペンドイベントが送られる．しかし，ドメイン 0 内の OS がクラッシュした時にはこのイベントを送ることができないので，Xen からドメイン U にサスペンドイベントを送られるようにする機構を実装した（図 2）．

ドメイン U 内の OS のサスペンド処理が完了すると，ドメインをサスペンド状態に移行させるために Xen のハイパーコールが呼ばれるので，その中でドメインの状態をメモリに保存する．ドメインの状態は，レジスタなどの実行コンテキストおよびイベントチャンネルの状態などの共有情報からなる．ドメインの状態の保存に必要なメモリサイズは i386 アーキテクチャで 8 KB 以下，x86_64 アーキテクチャで 16 KB 以下である．サスペンドを完了してドメインが破棄された時には，4.2 節で述べたようにドメイン U に割り当てられていたマシンメモリを予約しておく．

Xen を再起動する時には，すべての CPU のキャッシュをフラッシュする．Pentium や Opteron の CPU キャッシュはライトバックキャッシュであり，メモリに書き込む命令を実行した時には CPU キャッシュのみに書き込まれる．Xen の再起動に伴う計算機の再起動時に CPU キャッシュに残っている値がメモリに書き込まれずに失われるのを防ぐために，CPU キャッシュをフラッシュする wbinvd 命令を実行する．この命令は外部キャッシュの内容がメモリに書き出されるまで待たないため，メモリに書き終わるまで一定時間待つ．理想的な待ち時間はキャッシュサイズをメモリ帯域で割った時間になるが，実際のメモリ帯域は一定ではないので，現在の実装では 500 ms 待っている．

Xen を再起動した時には計算機も再起動されるが，メモリの内容を 0 で埋めるメモリチェックを BIOS の設定で省略することにより，メモリの内容を保持したまま計算機を再起動させることができる．その後 Xen が再び起動したら，ドメインのメモリを再構築するために必要なマシンメモリを Xen に使われないように，P2M マッピングテーブルに従って再起動前に各ドメインに割り当てられていたすべてのマシンメモリを予約する．同様に，ドメインの状態を保存したメモリも予約する．

Xen の再起動後にドメイン U をレジュームする時には新しいドメインを作成し，P2M マッピングテーブルに従ってレジュームするドメインに割り当てられていたマシンメモリを割り当てる．次に，保存しておいたドメインの状態から実行コンテキストおよび共有情報を復元して，ドメインの実行を再開する．ドメイン内の OS の実行はサスペンド状態に移行するハイパーコールが終了した位置から再開され，レジューム処理を行った後で通常の処理を再開する．

4.4 Warm-cache Reboot

ドメイン内の Linux カーネルを最初に起動する時には，ファイルキャッシュとして使われているメモリページ（ページキャッシュ）を管理するキャッシュマッピングテーブルを作成する．このテーブルは，デバイス番号，inode 番号，ファイルオフセットからページキャッシュとして使われているメモリページのフレーム番号を得るためのハッシュテーブルである．同様のテーブルは Linux カーネルも管理しているが，カーネルの内部構造に依存した構造になっているため，独自のキャッ

シュマッピングテーブルを実装した．このテーブルのサイズはドメインに割り当てられた物理メモリ 1 GB に対して 32 ビット環境で 4 MB，64 ビット環境で 8 MB となる．

カーネルがディスクからページキャッシュにデータを読み込んだ時には，キャッシュマッピングテーブルにエントリを追加する．ページキャッシュが破棄された時には，キャッシュマッピングテーブルからエントリを削除する．

一方，ドメインを再起動した時には，Xen が P2M マッピングテーブルに従って再起動前と同じ物理メモリページを割り当てる．そして，ドメイン内の Linux が再起動したら，キャッシュとして使われていたメモリページがカーネルによって使われないように，キャッシュマッピングテーブルに登録されているメモリページを予約する．ただし，i386 アーキテクチャにおいてはこの時点で 896 MB を超えるハイメモリのページを予約することができないため，ハイメモリに置かれたページキャッシュは後で予約する．

再起動後にディスクにアクセスする時には，カーネルはまず再起動後にキャッシュされたページキャッシュから探し，見つからなければキャッシュマッピングテーブルを調べて再起動前にキャッシュされていたページキャッシュが存在するかどうか調べる．キャッシュマッピングテーブルに存在すれば，そのメモリページの予約を解除してカーネルの管理下に移動させることでページキャッシュを再利用する．

メモリが不足した場合は再起動前のファイルキャッシュを再利用するために予約されているメモリページを解放する必要があるが， この機能はまだ未実装である．

5 実験

我々は提案手法によりリブートリカバリをどの程度高速化できるかを調べる実験を行った．実験対象の計算機には，Pentium 3.0 GHz の CPU，デュアルチャンネル PC3200 DDR SDRAM メモリ 2 GB（6.4 GB/s），AMI BIOS v2.51，160 GB のシリアル ATA のディスク（150 MB/s，7,200 rpm，8 MB キャッシュ），1 Gbps の NIC を備えた計算機を用いた．VMM には Xen 3.0.0，その上の VM で動く OS には Linux 2.6.12.6 を用いた．VM が使用するディスクには物理ディスクのパーティションを割り当て，制御用 VM に割り当てるメモリサイズは 256 MB とした．一方，クライアント計算機には Pentium 3.0 GHz の CPU，1 GB のメモリ，1 Gbps の NIC を備えた計算機を用い，OS は Linux 2.6.10 であった．

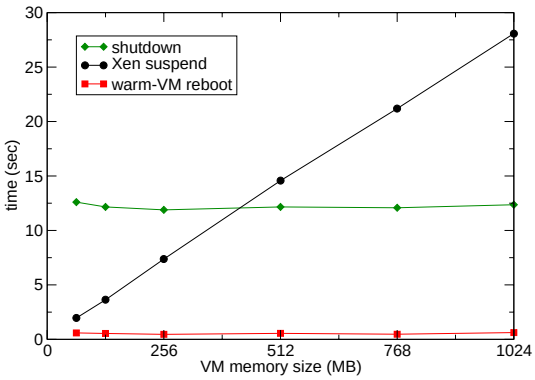

図 3: VM のメモリサイズとサスペンド時間

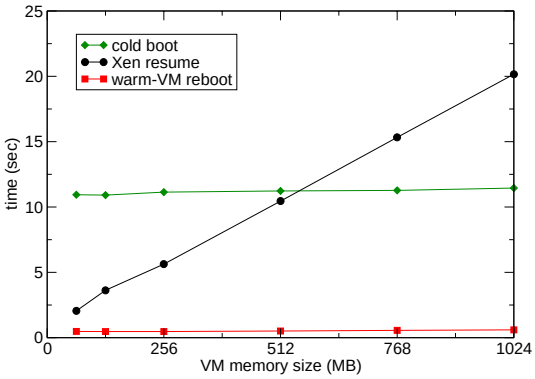

図 4: VM のメモリサイズとレジューム時間

5.1 Warm-VM Reboot による高速化

5.1.1 VM のサスペンド・レジューム時間

Warm-VM Reboot の効果を調べるために，VM のサスペンドとレジュームにかかる時間を測定した．この実験は，(1) Warm-VM Reboot を用いた場合，(2) メモリイメージをディスクに保存する Xen のサスペンド機能を用いた場合，(3) 比較対象として VM をシャットダウンして起動し直した場合について行った．VM で動作させるサービスは sshd のみとした．

まず，1 つの VM に対して，割り当てるメモリサイズを変化させた時の VM のサスペンド・レジューム時間について測定を行った．Warm-VM Reboot を用いる場合は Xen のサスペンド機能を用いる場合と条件を揃えるために，制御用 VM からサスペンドを行うようにした．VM のメモリサイズを 64 KB から 1 GB まで変化させた時のサスペンド時間を図 3 に，レジューム時間を図 4 に示す．

これらの図から，Warm-VM Reboot を用いた場合は

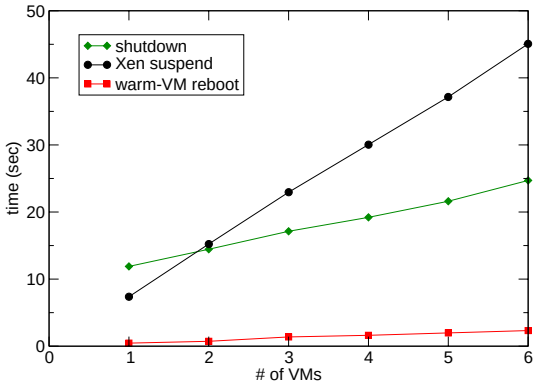

図 5: VM の数とサスペンド時間

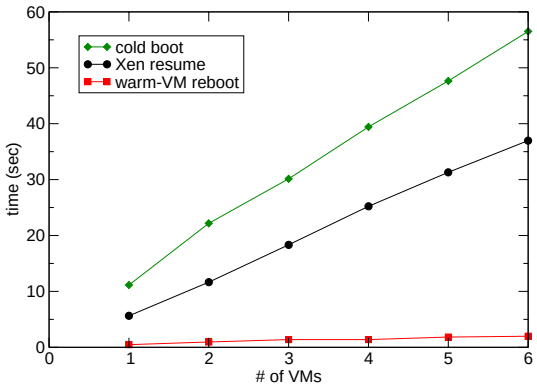

図 6: VM の数とレジューム時間

VM のメモリサイズにはほとんど依存せず，1 GB のメモリを割り当てた時でもサスペンドは 0.62 秒で終了し，レジュームも 0.60 秒で終了している．これは Warm-VM Reboot がサスペンド時に VM のメモリをコピーしないためである．一方，Xen のサスペンド機能を用いた場合は VM のメモリサイズに依存し，1 GB を割り当てている時にはサスペンドに 28 秒かかり，レジュームには 20 秒かかっている．これは VM の全メモリをディスクに書き込んでいるためである．他方，VM を起動し直す場合は VM のメモリサイズにはほとんど依存せず，シャットダウンには 12 秒程度，起動には 11 秒程度かかっている．ただし，VM を起動し直す場合はファイルキャッシュを含めて全ての実行状態を失なうため，レジュームした場合と比べると一般的にはさらにコストがかかる．

次に，複数の VM を用いた場合について，すべての VM を並列にサスペンド・レジュームするのにかかる時間を測定した．VM のメモリサイズは 256 MB に固定し，VM の数を 1 から 6 まで変化させた時のサスペンドにかかる時間を図 5 に，レジュームにかかる時間を図 6 に示す．どの場合も VM の数が増えると所要時間が増加している．しかし，Warm-VM Reboot を用

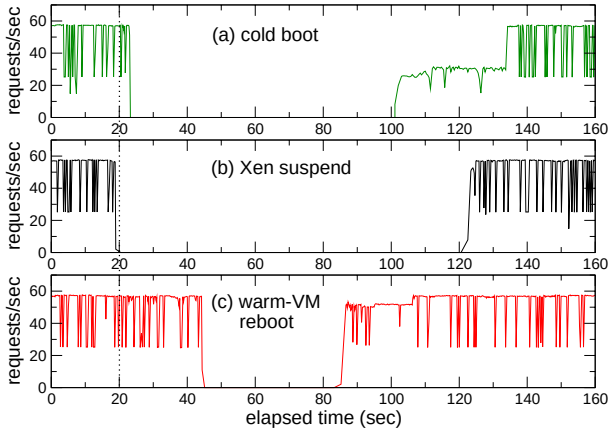

図 7: VMM 再起動時のサーバ処理性能

いた場合には VM の数が 6 の時でもサスペンドに 2.3 秒，レジュームに 2.0 秒しかかかっていないのに対し，Xen のサスペンド機能を用いた場合にはサスペンドに 45 秒，レジュームに 37 秒かかっている．VM を起動し直す場合はさらに時間がかかり，VM の数が 6 の場合にはシャットダウンに 25 秒，起動に 56 秒かかっている．

5.1.2 ダウンタイム

Warm-VM Reboot によりサービスの実際のダウンタイムがどの程度削減されるかを調べた．この実験では VM 上で Apache ウェブサーバを動かし，別のホストから httperf を使ってリクエストを送り続けておく．リクエストはサーバに置かれた 1 MB のファイル 900 個に順番に送り続けた．VM に割り当てるメモリサイズを 1 GB としたため，すべてのファイルの内容をキャッシュできる．その上で，制御用 VM を reboot コマンドで再起動して計算機が再起動された後でサービスが再び正常に動作するようになるまでのサーバの処理性能を測定した．測定したのは，(a) VM を起動し直した場合，(b) Xen のサスペンド機能を用いた場合，(c) Warm-VM Reboot を用いた場合である．

図 7 はそれぞれについてのサーバ処理性能の推移を示している．20 秒の地点で制御用 VM の再起動を開始した．(a) の VM を起動し直した場合は，VM がシャットダウンされた時点でサービスが処理されなくなり，78 秒たった後で VM 内の OS の起動が完了してサービスが再開されている．ただし，その後 33 秒はキャッシュミスの増加により 68%の性能しか出ていない．(b) の Xen のサスペンド機能を用いた場合は，VM のサスペンドを始めた時点でサービスが処理されなくなり，104 秒後にレジュームされるまでサービスが停止していた．

表 1: ファイル読み込みレート（MB/s）

条件	従来	Warm-cache Reboot
起動後 1 回目	53.2	52.9
起動後 2 回目	1260	1240
再起動後 1 回目	53.1	807
再起動後 2 回目	1260	1260

表 2: ウェブサーバの性能（リクエスト/秒）

条件	従来	Warm-cache Reboot
起動後 1 回目	18.2	17.3
起動後 2 回目	61.5	60.8
再起動後 1 回目	18.5	55.7
再起動後 2 回目	60.9	60.1

ただし，VM が復元されるとすぐにサーバ処理性能は元の水準に戻った．一方，(c) の Warm-VM Reboot を用いた場合は，制御用 VM の再起動を開始しても実際に計算機が再起動される直前までの 25 秒間はサービスが動き続けていた．VM がサスペンドされてサービスが停止した後，40 秒で VM がレジュームされてサービスが再開した．その後 20 秒間は 8%性能が低下しているが，その原因については現在調査中である．この結果から，我々の用いた実験環境では，Warm-VM Reboot は従来の手法と比べてダウンタイムを約半分にできていることが分かる．

5.2 Warm-cache Reboot による高速化

5.2.1 ファイル読み込み性能

Warm-cache Reboot の効果を調べるために，VM を再起動する前後でのファイル読み込みレートを測定した．この実験では，VM には 1 GB のメモリを割り当て，Warm-cache Reboot を用いた場合と用いない場合について，起動後と再起動後に 2 回ずつ 1 MB のファイルを 896 個読み込むのにかかる時間を測定した．この実験の結果を表 1 に示す．

この表より，従来は VM を再起動した直後のファイル読み込みレートは VM を最初に起動した後と変わらず 53 MB/s なのに対して，Warm-cache Reboot を用いた場合は再起動直後でも 807 MB/s まで向上している．このレートが再起動後 2 回目のファイルアクセスと比べて低いのは，キャッシュを再利用するための操作のオーバヘッドおよびファイルの内容以外のディレクトリエントリなどのキャッシュミスのためである．それでも，従来の 15 倍の性能が得られている．一方，Warm-cache Reboot を実現するためのオーバヘッドは 0〜1.6%であった．

5.2.2 ウェブサーバの性能

実際のサービスで Warm-cache Reboot の効果を調べるために，VM 上で Apache ウェブサーバを動かし，VM を再起動する前後で httperf を用いてウェブサーバの性能の変化を測定した．実験には httperf を 4 つ用い，それぞれがサーバに置かれた異なる 1 MB のファイル 224 個に 1 回ずつリクエストを送った．VM に割り当てたメモリサイズは 1 GB なので，リクエストされたすべてのファイルの内容をキャッシュすることができる．Warm-cache Reboot を用いた場合と用いなかった場合とで，起動後と再起動後に 2 回ずつ測定した結果を表 2 に示す．この表より，Warm-cache Reboot 用いると VM の再起動直後の性能が改善されていることが分かり，起動直後と比較すると 3 倍の性能が達成できている．一方，Warm-cache Reboot を実現するためのオーバヘッドは 1.2〜5.0%であった．

5.2.3 再起動後の性能

Warm-cache Reboot により VM の再起動時のサーバ処理性能がどの程度向上するか調べた．この実験では 5.1.2 節と同様に VM 上で動いている Apache ウェブサーバに httperf でリクエストを送り続けた．その上で，対象の VM を再起動し，サービスが再び正常に動作するようになるまでのリクエスト処理数を測定した．測定は (a) 通常の再起動を行った場合，(b) 再起動時にウェブサーバがアクセスするファイルの先読みを行った場合，(c) Warm-cache Reboot を用いて再起動した場合について行った．先読みは各ページキャッシュの先頭 1 バイトを読むことで行った．また，VM に割り当てるメモリサイズは 1 GB とした．

図 8 はそれぞれについてのサーバ処理性能の推移を示している．20 秒の地点で VM の再起動を開始した．どの場合も約 45 秒の地点でサービスを再開するまでは同様の挙動を示しているが，それ以降のサーバ性能は大きく異なる．(a) の通常の再起動では，ウェブサーバが起動してから 33 秒間は再起動前の 56%の性能しか達成できていない．(b) の先読みを行った場合は，先読み処理がウェブサーバの処理を阻害して，先読みを行わない場合より性能が悪化している．それらに対して，(c) の Warm-cache Reboot を用いた場合には，ウェブサーバが起動してから 20 秒間は最大性能を発揮できて

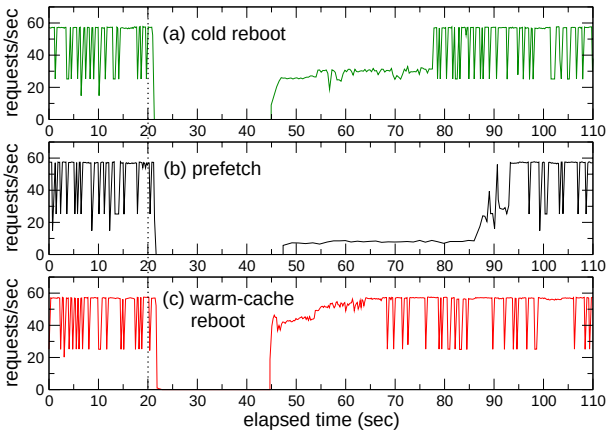

図 8: VM 再起動時のサーバ処理性能

いないが，それでも再起動前の 94%の性能を達成できている．

6 議論

6.1 適用可能性

VMM のアップデートにより再起動する場合，VM の状態として保存されるデータの構造に変更がなければ Warm-VM Reboot を行うことができる．このような変更は大幅なバージョンアップの際にだけ行われることが多いため，通常は問題にならない．制御 VM 内の OS のアップデートのために VMM を再起動する場合には VMM には変更が及ばないため，アップデートの内容に関わらず Warm-VM Reboot が可能である．一方，OS のファイルキャッシュはファイルの内容を保持しているだけであるので，OS の内部構造が大幅に変更されても Warm-cache Reboot により再利用可能である．

リソース不足などが原因で VMM が不安定になった場合には，VM をサスペンドして VM の状態をメモリに保存することができれば Warm-VM Reboot を用いることができる．VM をサスペンドできなかった場合でも VM の Warm-cache Reboot を行うことはできる．一方，VM 内の OS が不安定になった場合は OS を再起動できなくなる場合があるが，VM ごと強制終了させた後で Warm-cache Reboot を行うことができる．制御用 VM 内の OS が不安定になって VMM を再起動する場合は常に Warm-VM Reboot が可能である．

VMM がクラッシュした場合には，VM のメモリイメージ，VM の状態，P2M マッピングテーブルが破壊されず，VM をサスペンドすることができれば Warm-VM Reboot を行うことができる．ただし，VMM は OS と比べると小さいソフトウェアであるため，クラッシュする頻度は低いと考えられる．一方，OS のクラッシュにより再起動する場合には，ページキャッシュおよび 4.4 節で述べたキャッシュ・ビットマップとハッシュテーブルが破壊されなければ Warm-cache Reboot を行うことができる．ページキャッシュについては必要な時以外は書き込み禁止にすることで破壊を防ぐことができる [4]．キャッシュ・ビットマップとハッシュテーブルは VMM 経由で更新することにより破壊を防ぐことができると考えている．

6.2 ファイルキャッシュを保持する位置

我々は再起動後に VM 内部のファイルキャッシュを再利用できるようにするために Warm-cache Reboot を提案したが，VM 外部のファイルキャッシュを再利用する方法も考えられる．Xen では VM がファイルアクセスを必要とする時には管理用 VM が実際のディスクアクセスを行うため，管理用 VM でファイルキャッシュを保持しておくことにより，VM の再起動後にそのファイルキャッシュを再利用することができる．この場合，OS が再起動前のファイルキャッシュを管理する必要がなくなるため，OS に透過的にファイルキャッシュを再利用することができる．しかし，再利用の際には管理用 VM からファイルキャッシュの内容をコピーする必要があり，全ての VM のファイルキャッシュを保持するために管理用 VM に多くの物理メモリを割り当てなければならなくなる．

それに対して，Warm-cache Reboot を用いる場合は，再起動後に直接 VM 内部のファイルキャッシュにアクセスすることができ効率がよい．ただし，VM がクラッシュした時にはファイルキャッシュの内容が破壊される危険性があるため，6.1 節で述べたように保護する必要がある．

7 関連研究

ACPI [1] の S3 状態（Suspend To RAM）はノート PC では古くから実現されている機能であり，計算機をサスペンドして停止させた後，メモリ上のデータを保持し続けることで高速なレジュームを可能にする．Warm-VM Reboot はこの機能を VM に対して提供するものであり，VMM および計算機の再起動中に VM のメモリイメージを保持し続けることで実現されている．

Recovery Box [2] は OS やアプリケーションの状態を不揮発性メモリに保存しておき，OS の再起動後にその状態を使ってリカバリすることで再起動後の処理を

高速化する．Warm-VM Reboot と似ているが，アプリケーション毎に保存する状態を決めてリカバリを行う必要がある．それに対して，Warm-VM Reboot は VM ごとに保存・復元を行うため，汎用的に用いることができる．また，Recovery Box では再起動時にメモリ中のカーネルイメージを再利用しているが，ディスクが高速化している現在ではカーネルイメージを読み込む程度であればほとんどオーバヘッドにはならない．

Rio ファイルキャッシュ [4] は OS の再起動時にディスクに書き戻されていないダーティキャッシュの内容を保存しておき，再起動後にディスクに書き戻すことでファイルの整合性を保つ．ディスクに書き込まれていないデータを OS のクラッシュから守るのが目的であるため，ダーティでないキャッシュは破棄される．この点で，再起動後のディスクアクセスを減らすことを目的とした Warm-cache Reboot とは異なる．

Xen のライブマイグレーション [5] や VMware の VMotion [8] は実行中の VM をほとんど停止させることなく他のホストに移動させるため，ダウンタイムを短く保つことができる．VMM を再起動する時に全ての VM を他のホストに移動させれば，再起動の影響を小さくすることができる．しかし，マイグレーションを完了するには時間がかかり，VMM や制御用 VM 内の OS がクラッシュするとマイグレーションさせることができない．

8 まとめ

本稿では，VM を用いたサーバ統合において再起動の影響を最小限に抑えるために，Warm-VM Reboot と Warm-cache Reboot からなる高速なリブートリカバリを提案した．Warm-VM Reboot は VMM の再起動時に VM のメモリイメージをディスクに保存することなく高速に VM をサスペンドし，再起動後に高速にレジュームする．Warm-cache Reboot は VM を再起動した後で OS が保持していたファイルキャッシュを再利用することで，再起動直後のキャッシュミスを軽減する．我々はこれらの機構を Xen およびその上で動作する Linux に実装し，その効果を確認する実験を行った．その結果，我々の実験環境では，Warm-VM Reboot によってダウンタイムを約半分に減らすことができ，Warm-cache Reboot によって再起動直後のサーバの性能低下を 8%以下に抑えることができた．

今後の課題としては，より多くのメモリを搭載できる x86_64 アーキテクチャでの実装を完了させることが挙げられる．また，今回は提案手法に対して最も効果的な実験を行ったが，より現実的な実験での評価を行う必要がある．さらに，VM 内の OS や VMM がクラッシュした場合に備えて再起動後に利用するデータを保護するようにし，実際にフォールトを挿入した実験も行う必要があると考えている．

参考文献

[1] Advanced Configuration and Power Interface Specification, `http://www.acpi.info`.

[2] Baker, M. and Sullivan, M.: The Recovery Box: Using Fast Recovery to Provide High Availability in the UNIX Environment, in *Proc. of the Summer USENIX Conf.*, pp. 31–44 (1992).

[3] Candea, G., Kawamoto, S., Fujiki, Y., Friedman, G. and Fox, A.: Microreboot – A Technique for Cheap Recovery, in *Proc. of Symp. on Operating Systems Design and Implementation*, pp. 31–44 (2004).

[4] Chen, P., Ng, W., Chandra, S., Aycock, C., Rajamani, G. and Lowell, D.: The Rio File Cache: Surviving Operating System Crashes, in *Proc. of the Int'l Conf. on Architectural Support for Programming Languages and Operating Systems*, pp. 74–83 (1996).

[5] Clark, C., Fraser, K., Hand, S., Hansen, J., Jul, E., Limpach, C., Pratt, I. and Warfield, A.: Live Migration of Virtual Machines, in *Proc. of Symp. on Networked Systems Design and Implementation* (2005).

[6] Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Pratt, I., Warfield, A., Barham, P. and Neugebauer, R.: Xen and the Art of Virtualization, in *Proc. of Symp. on Operating Systems Principles* (2003).

[7] Swift, M., Bershad, B. and Levy, H.: Improving the Reliability of Commodity Operating Systems, in *Proc. of Symp. on Operating Systems Principles* (2003).

[8] VMware Inc., : VMware, `http://www.vmware.com/`.