VMMのソフトウェア若化を考慮したクラスタ性能の比較

光来健一

我々は Warm-VM Reboot と呼ばれる高速なソフトウェア若化手法を提案している [7]。本論文ではクラスタ環境で Warm-VM Reboot を行う際のトータルスループットについて考察する。トータルスループットを定義するために、VMM のソフトウェア若化だけでなく OS のソフトウェア若化も考慮する。Warm-VM Reboot を用いる場合と用いない場合、さらに、VMM のソフトウェア若化の前に VM の移送を行う場合についてトータルスループットを定義する。これらの手法について実験に基づいてトータルスループットの比較を行った結果、Warm-VM Reboot を用いる場合のトータルスループットが常に最大になることが分かった。

1 はじめに

ソフトウェアが時間とともに劣化する現象はソフトウェアエージング [6] と呼ばれている。その原因はシステムリソースの枯渇やデータの破壊などである。ソフトウェアエージングはソフトウェアの性能劣化やクラッシュを引き起こす。近年、仮想計算機（VM）を用いたサーバ統合が広く行われるようになるにつれて、仮想計算機モニタ（VMM）のソフトウェアエージングが問題になってきている。 このようなサーバでは多くの VM が VMM 上で動いているため、VMM のソフトウェアエージングはすべての VM に直接影響する。

Kenichi Kourai, 九州工業大学, Kyushu Institute of Technology, 科学技術振興機構, CREST, Japan Science and Technology Agency

このような問題を解決するために、ソフトウェア若化（software rejuvenation）と呼ばれる手法が提案されている [6]。ソフトウェア若化は時々VMM を停止させ、内部状態を正常に戻してから再開する。典型的な例は VMM の再起動であるが、VMM を再起動すると VM 上で動いているすべての OS を再起動することになる。このことは OS によって提供されているサービスのダウンタイムを増大させる。また、OS が保持していたファイルキャッシュは再起動によって失われるため、OS の再起動後にはキャッシュミスにより性能が低下する。このようなダウンタイムや性能低下はサーバにとって致命的である。

我々は Warm-VM Reboot と呼ばれる高速なソフトウェア若化手法を提案してきた [7]。Warm-VM Reboot は VMM の再起動中に VM のメモリイメージを保持しておき、再起動後に再利用することで、VMM だけを効率よく再起動する。Warm-VM Reboot では、オンメモリ・サスペンド機構が VM のメモリイメージをメインメモリ上に保持したまま VM を高速にサスペンドする。そして、オンメモリ・レジューム機構が保持されているメモリイメージを使って VM を高速にレジュームする。メモリイメージを保持したまま VMM を再起動するために、ハードウェア・リセットを伴わないクイック・リロード機構により VMM を再起動する。Warm-VM Reboot は VM 上の OS の再起動を必要としないため、OS のダウンタイムを減らし、キャッシュミスによる性能低下を防ぐことができる。

本論文ではクラスタ環境でソフトウェア若化を行

う際のトータルスループットについて考察する。トータルスループットを定義するために、VMM のソフトウェア若化だけでなく OS のソフトウェア若化も考慮してモデル化を行う。モデルに基づいてダウンタイムや性能低下率を用いてトータルスループットを定義する。トータルスループットは Warm-VM Reboot を用いる場合と用いない場合、さらに、VMM のソフトウェア若化の前に VM の移送を行う場合について考える。実験によってソフトウェア若化によるダウンタイムや性能低下率を見積もり、5 つの手法についてトータルスループットの比較を行った。その結果、Warm-VM Reboot を用いた場合のトータルスループットが常に最も高くなることが分かった。

以下、2 章では我々が提案している Warm-VM Reboot について述べる。3 章では様々な手法を用いた場合についてトータルスループットを定義する。4 章では実験に基づいてそれぞれの手法のトータルスループットの比較を行う。5 章では関連研究について触れ、6 章で本論文をまとめる。

2 Warm-VM Reboot

VMM のソフトウェア若化を高速化するために、我々は Warm-VM Reboot と呼ぶソフトウェア若化手法を提案している [7]。基本的なアイデアは VMM の再起動を通して VM のメモリイメージを保持し、再起動後にそのメモリイメージを再利用することである。Warm-VM Reboot は VM のオンメモリ・サスペンド／レジューム機構および VMM のクイック・リロード機構を用いて、VMM だけを効率よく再起動することを可能にする。VMM は再起動する前にオンメモリ・サスペンドを用いてすべての VM を高速にサスペンドし、クイック・リロードを用いて自分自身を高速に再起動する。VMM が再起動したら、オンメモリ・レジュームを用いてすべての VM を高速にレジュームする。

オンメモリ・サスペンドは VM が使用しているメモリイメージをそのまま凍結し、VM を一時停止させる。そのメモリイメージは VM が再開されるまで VMM の再起動を通してメモリ上に保持される。この機構はメモリイメージをディスク等の外部記憶に保存する必要がなく、フラッシュメモリ等の不揮発性メモリにコピーする必要もない。オンメモリ・サスペンドにかかる時間は VM に割り当てられているメモリサイズにほとんど依存しないため、非常に効率がよい。メモリイメージ以外の VM の実行状態については、VMM の再起動を通して保持されるメモリ領域にコピーする。

オンメモリ・レジュームは凍結されたメモリイメージを解凍し、そのメモリイメージを再利用して VM の実行を再開する。この機構もメモリイメージを外部記憶から読み込んだり、不揮発性メモリからコピーしたりする操作を必要としない。VM のメモリイメージは完全に復元されるため、再起動直後であってもキャッシュミスの頻発による性能低下を防ぐことができる。同時に、VM の実行状態も復元される。オンメモリ・サスペンド／レジューム機構は、メインメモリ上のメモリイメージにアクセスすることなくサスペンド／レジュームできるという点で、ACPI S3 状態（Suspend To RAM）に似ている。

クイック・リロードは VM のメモリイメージを保持したまま高速に VMM を再起動させる。通常、VMM の再起動は VMM の実行ファイルを読み込むためにハードウェア・リセットを必要とするが、ハードウェア・リセットはメモリの内容が保持されることを保証しない上、メモリチェックや SCSI チェックなどに時間がかかる。クイック・リロードはソフトウェア的に新しい VMM の実行ファイルをメモリ上に読み込み、そのエントリ・ポイントにジャンプして実行を開始することでハードウェア・リセットを回避する。このようなソフトウェア機構が VMM の再起動中のメモリを管理するので、メモリの内容が保持されることを保証できる。さらに、クイック・リロードは VMM の初期化の際に凍結された VM のメモリイメージが破壊されるのを防ぐ。

3 クラスタ環境におけるソフトウェア若化

ソフトウェア若化はクラスタ環境で用いるのに適している [2] [9]。クラスタ環境では複数のホストが同一のサービスを提供し、ロードバランサがリクエストをそのうちの 1 つに振り分けることができる。い

くつかのホストが VMM のソフトウェア若化のために再起動されていても、サービスのダウンタイムは 0 になる。しかし、いくつかのホストが再起動されている間、クラスタのトータルとしてのスループットは低下する。

トータルスループットを計算するには、各 VM について可用性とソフトウェア若化による性能低下を考える必要がある。ソフトウェア若化を行っている間はサービスを全く提供できないため可用性が低下し、その分だけスループットも低下する。また、ソフトウェア若化に起因する性能低下によってもスループットが低下する。

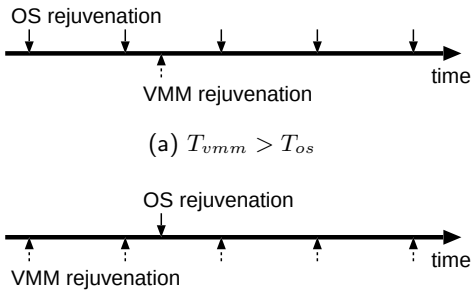
クラスタのトータルスループットを見積もるために、m ホストからなるクラスタ環境を考える。p を 1 つのホスト上のすべての VM のスループットの和とすると、すべてのホストが動いている時、トータルスループットは mp となる。1 つのホストで VMM のソフトウェア若化を行っている間、そのホストはサービスを提供できないため、トータルスループットは $(m-1)p$ に低下する。

3.1 Warm-VM Reboot

通常、VMM のソフトウェア若化は OS のソフトウェア若化と組み合わせて用いられる。簡単のために、ソフトウェア若化は一定時間経過した時に行われる（時間に基づいたソフトウェア若化 [5]）と仮定する。VMM のソフトウェア若化の間隔を T_{vmm} とし、OS のソフトウェア若化の間隔を T_{os} とする。

Warm-VM Reboot が用いられる場合、VMM のソフトウェア若化は OS のソフトウェア若化とは独立に行うことができる。Warm-VM Reboot は VMM のソフトウェア若化の間、OS をサスペンドさせており、OS のソフトウェア若化を同時には行わないためである。図 1 は $T_{vmm} > T_{os}$ の時と $T_{vmm} \leq T_{os}$ の時のソフトウェア若化のタイミングを示している。

Warm-VM Reboot を用いる場合の可用性を定義するために、まず T_{vmm} の期間のダウンタイムを見積もる。$D_w(n)$ を Warm-VM Reboot によるダウンタイムとする。この時間は VM の数 n に比例する。D_{os} を OS のソフトウェア若化によるダウンタイム、



(a) $T_{vmm} > T_{os}$

(b) $T_{vmm} \leq T_{os}$

図 1　Warm-VM Reboot を用いた際のソフトウェア若化のタイミング

N_w を T_{vmm} の期間に行われる OS のソフトウェア若化の回数の平均とする。N_w は $\frac{T_{vmm}-D_w(n)}{T_{os}}$ である。この時、ダウンタイムの合計は $D_w(n) + N_w D_{os}$ となる。よって、可用性は

$$A_w = 1 - \frac{D_w(n) + N_w D_{os}}{T_{vmm}} \quad (1)$$

と定義される。

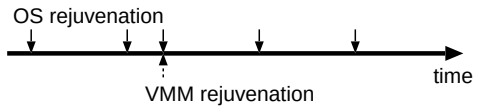
次に、ソフトウェア若化による性能低下を定義する。Warm-VM Reboot においては、サスペンドされた VM がレジュームされる時に VM 間のリソース競合によって性能が低下する。R_{vmm} を Warm-VM Reboot 後の性能が低下している期間とし、ω をその期間の性能低下率とする。これより、VMM のソフトウェア若化がトータルスループットに及ぼす性能低下の度合いは $\frac{\omega R_{vmm}}{T_{vmm}}$ となる。

一方、OS のソフトウェア若化は OS の再起動直後のキャッシュミスによる性能低下を引き起こす。さらに、OS の再起動中はリソース競合によって他の VM の性能低下を引き起こす。R_{os} をこの二種類の性能低下の期間の合計とし、δ をこの期間の性能低下率の平均とする。OS のソフトウェア若化は T_{vmm} の間に N_w 回行われるため、トータルスループットに及ぼす性能低下の度合いは $\frac{N_w \delta R_{os}}{T_{vmm}}$ となる。

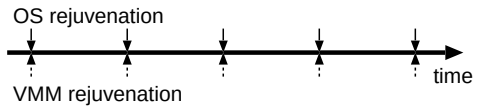
これらより、トータルスループットは

$$P_w = \left(A_w - \frac{\omega R_{vmm} + N_w \delta R_{os}}{T_{vmm}} \right) mp \quad (2)$$

と定義される。



(a) $T_{vmm} > T_{os}$



(b) $T_{vmm} \leq T_{os}$

図 2　Cold-VM Reboot を用いた際のソフトウェア若化のタイミング

3.2　Cold-VM Reboot

Warm-VM Reboot との比較のために、通常の再起動による VMM のソフトウェア若化のことを Cold-VM Reboot と呼ぶことにする。Cold-VM Reboot を行う際には、VMM を再起動する前にすべての VM 内の OS をシャットダウンし、VM を終了させる必要がある。そして、ハードウェアリセットを行って VMM を再起動し、VM を作成し、再びすべての OS を起動することになる。

Cold-VM Reboot を用いる場合、VMM のソフトウェア若化は OS のソフトウェア若化のタイミングに影響を及ぼす。VMM のソフトウェア若化を行う時には OS がシャットダウンされるため、結果的に OS のソフトウェア若化を行ったことになるためである。図 2 (a) は $T_{vmm} > T_{os}$ の時のソフトウェア若化のタイミングを示している。OS のソフトウェア若化が VMM のソフトウェア若化と同時に行われるため、OS のソフトウェア若化のためのタイマがそこでリセットされ、OS のソフトウェア若化が再スケジュールされる。$T_{vmm} \leq T_{os}$ の時は、図 2 (b) のように、OS のソフトウェア若化は常に VMM のソフトウェア若化と同時に行われる。OS のソフトウェア若化のためのタイマが発火する前に VMM のソフトウェア若化が行われ、タイマがリセットされるためである。

Cold-VM Reboot を用いる場合の可用性を定義するために、T_{vmm} の期間のダウンタイムを見積もる。$D_c(n)$ を Cold-VM Reboot によるダウンタイム、N_c を $\frac{T_{vmm}-D_c(n)}{T_{os}}$ とした時、ダウンタイムの合計は $D_c(n) + \lfloor N_c \rfloor D_{os}$ となる。$\lfloor N_c \rfloor$ は T_{vmm} の期間に行われる OS のソフトウェア若化の正確な回数である。Cold-VM Reboot を行った時に OS のソフトウェア若化のタイマはリセットされるため、OS のソフトウェア若化は必ずこの回数しか行われない。よって、可用性は

$$A_c = 1 - \frac{D_c(n) + \lfloor N_c \rfloor D_{os}}{T_{vmm}} \tag{3}$$

と定義される。

Cold-VM Reboot を用いる場合も、VMM と OS のソフトウェア若化によって性能低下が引き起こされる。R'_{vmm} を Cold-VM Reboot 後の性能が低下している期間とし、ω' をその期間の性能低下率とす。R'_{vmm} と ω' は R_{vmm} と ω とは異なる。Warm-VM Reboot と違い、Cold-VM Reboot は VMM とともに OS の再起動も必要とする。その時、VM 内のすべての OS は同時に再起動され、再起動後はキャッシュミスのためにディスクに頻繁にアクセスする。そのため、性能低下率と性能が低下する期間は増大する。よって、VMM のソフトウェア若化によるトータルスループットに対する性能低下の度合いは $\frac{\omega' R'_{vmm}}{T_{vmm}}$ となる。一方、OS のソフトウェア若化によるトータルスループットに対する性能低下の度合いは $\frac{\lfloor N_c \rfloor \delta R_{os}}{T_{vmm}}$ となる。

これらから、トータルスループットは

$$P_c = \left(A_c - \frac{\omega' R'_{vmm} + \lfloor N_c \rfloor \delta R_{os}}{T_{vmm}} \right) mp \tag{4}$$

となる。$T_{vmm} \leq T_{os}$ の時には $\lfloor N_c \rfloor$ は 0 になる。

3.3　VM 移送

VMM のソフトウェア若化を行う際に VM をサスペンドまたはシャットダウンする代わりに、VM を他のホストに移送することができる。この手法を用いると、VMM のソフトウェア若化を開始する前にすべての VM を対象ホストから別のホストに移動してから、対象ホストで VMM を再起動することができる。移送元のホストで VMM が再起動されている間も、移送された VM は移送先ホストで動き続けることができる。VM 移送機構は VM をサスペンドし、そのメモリイメージと実行状態を転送し、移送先ホストで

VM をレジュームする。VM が使うディスクは 2 つのホスト間で共有されるため、ディスクの内容を転送する必要はない。移送先ホストとして余分なホストが 1 台必要となるが、このホストはクラスタ内の他のすべてのホストとの間で移送先として共有できる。

特に、Xen のライブマイグレーション [3] は VM を移送する際のダウンタイムをほぼ 0 にすることができる。通常の移送は移送開始時に VM を停止させるのに対して、ライブマイグレーションは VM を動かしたままそのメモリイメージを転送する。メモリイメージの転送がひと通り終わったら、転送後に変更された部分を再送し、これを差分が十分小さくなるまで繰り返す。最終段階で VM を停止させ、わずかなメモリイメージの差分と VM の実行状態を転送する。他の VM が使うネットワーク帯域への影響を抑えるために、転送レートを適応的に制御することもできる。

しかし、移送を行う場合、1 台のホストは移送先ホストとして予約しておかなければならないため、移送が行われていない時でもトータルスループットは $(m-1)p$ となる。これは VM 移送を用いない場合の $\frac{m-1}{m}$ であり、m が十分大きくない時は影響が大きい。

3.3.1 通常の移送

通常の移送を行う場合のトータルスループットを定義するために、まず可用性について考える。D_m を移送による平均ダウンタイム、N_m を T_{vmm} の期間に行われる OS のソフトウェア若化の回数の平均とすると、可用性は

$$A_m = 1 - \frac{D_m + N_m D_{os}}{T_{vmm}}$$

と定義できる。

ある VM の性能は他の $n-1$ 個の VM の移送によって低下する。巨大なメモリイメージの転送は CPU 時間とネットワーク帯域を消費し、他の VM のサービスを阻害するためである。1 つの VM を移送するのにかかる時間は D_m なので、n 個の VM を順番に移送すると仮定すると、性能が低下している期間は $(n-1)D_m$ となる。γ をこの期間の性能低下率とすると、γ は

$$\gamma = 1 - \frac{V_c}{((n-1)/n)p}$$

と定義される。V_c は移送が行われている間の移送元ホストと移送先ホストで動いているすべての VM のスループットの和である。同時に 1 つの VM だけが移送されるため、この期間の最大スループットは性能低下がなかったとしても $\frac{n-1}{n}p$ である。

ここで、V_c は

$$V_c = \sum_{i=1}^{n} \frac{M(i)}{M_t} \left((n-i)V_s(i) + (i-1)V_d(i)\right)$$

と定義される。$V_s(i)$ は i 番目の VM が移送されている間に移送元ホストで動いている各 VM のスループットであり、$V_d(i)$ は移送先ホストで動いている各 VM のスループットである。移送元ホストと移送先ホストで動いている VM の数は時間とともに変化するため、これらのスループットは i の値に依存する。同じホストで動いている VM の数が減れば性能はよくなる。i 番目の VM が移送されている時、$n-i$ 個の VM が移送元ホストで動いており、$i-1$ 個の VM が移送先ホストで動いている。それゆえ、両ホスト上のすべての VM のスループットの和は $(n-i)V_s(i)+(i-1)V_d(i)$ となる。

$\frac{M(i)}{M_t}$ は総移送時間に対する i 番目の VM を移送する時間の割合である。$M(i)$ は i 番目の VM を移送するのにかかる時間、M_t は n 個の VM をすべて移送するのにかかる時間とする。つまり、M_t は $\sum_{i=1}^{n} M(i)$ である。ちなみに、$\frac{M_t}{n}$ が D_m である。

以上より、通常の移送によるトータルスループットに対する性能低下の度合いは $\frac{\gamma(n-1)D_m}{T_{vmm}}$ となる。移送を並行して行うと各 VM のダウンタイムを増大させてしまうため、VM を 1 つずつ移送すると仮定した。一方、OS のソフトウェア若化による性能低下は $\frac{N_m \delta R_{os}}{T_{vmm}}$ である。これより、トータルスループットは

$$P_m = \left(A_m - \frac{\gamma(n-1)D_m + N_m \delta R_{os}}{T_{vmm}}\right)(m-1)p \tag{5}$$

と定義される。移送を用いる場合、サービスを提供するのに $m-1$ 台のホストしか同時には使えない。

3.3.2 ライブマイグレーション

ライブマイグレーションを用いる場合、ダウンタイムは OS のソフトウェア若化によってのみ引き起こされると考えてよいため、可用性は

$$A_l = 1 - \frac{N_l D_{os}}{T_{vmm}}$$

と定義できる。N_l は T_{vmm} の期間の OS のソフトウェア若化の回数の平均であり、$\frac{T_{vmm}}{T_{os}}$ である。

ライブマイグレーションによるトータルスループットに対する性能低下の度合いは $\frac{\gamma' M_l}{T_{vmm}}$ である。γ' は性能低下率であり、M_l は総移送時間である。性能が低下している期間は M_l に等しい。

ここで、γ' は

$$\gamma' = 1 - \frac{V_c'}{p}$$

と定義できる。V_c' は通常の移送における V_c に似ているが、

$$V_c' = \sum_{i=1}^{n} \frac{M(i)}{M_l} \left((n-i)V_s(i) + V_m(i) + (i-1)V_d(i)\right)$$

と定義される。違いは $V_m(i)$ の項を含むことである。この項は移送中の VM のスループットである。ライブマイグレーションでは移送中の VM は停止されないため、移送中にダウンタイムは発生せず、性能低下が起こるだけである。それゆえ、この期間の最大スループットは p となる。

一方、OS のソフトウェア若化による性能低下は $\frac{N_l \delta R_{os}}{T_{vmm}}$ である。それゆえ、トータルスループットは

$$P_l = \left(A_l - \frac{\gamma' M_l + N_l \delta R_{os}}{T_{vmm}}\right)(m-1)p \quad (6)$$

となる。

4 実験

様々な手法を用いた際のトータルスループットを比較するための実験を行った。スループットを測定するためにサーバ上で VM を動かし、クライアントから各 VM で動く Apache ウェブサーバにリクエストを送った。サーバ上には 11 個の VM を作成し、それぞれに 1 GB のメモリを割り当てた。各 VM のウェブサーバは 5,000 個の 128 KB のファイルを提供し、クライアントはこれらのファイルに順番にアクセスするようにした。これらのファイルは全て各 VM のメモリ上にキャッシュすることができる。クライアントでは httperf プロセスを 11 個動かし、毎秒 50 リクエストをサーバに送り続けた。

4.1 Warm/Cold-VM Reboot

Warm-VM Reboot と Cold-VM Reboot を用いた際のトータルスループット P_w と P_c を見積もるために、VMM と OS のソフトウェア若化をそれぞれ行い、1 台のホストのすべての VM のスループットの和を測定した。サーバマシンには、デュアルコア Opteron Model 280 の CPU を 2 基、PC3200 DDR SDRAM を 12 GB、15,000 rpm の SCSI ディスク、ギガビットイーサネットを搭載した PC を用いた。VMM として、Warm-VM Reboot を用いる実験では Xen 3.0.0 をベースとして我々が開発した VMM を用い、それ以外の実験では Xen 3.0.0 のオリジナルの VMM を用いた。VMM 上で動く OS は Xen 用に修正された Linux 2.6.12 であった。ディスクの 1 つのパーティションを 1 つの VM の仮想ディスクとした。Xen の特権 VM であるドメイン 0 に割り当てたメモリは 512MB であった。クライアントマシンには、Xeon 3.06 GHz を 2 基、2 GB のメモリ、ギガビットイーサネットを搭載した PC を用いた。OS は Linux 2.6.8 であった。これらの PC はギガビットイーサネットスイッチで接続した。

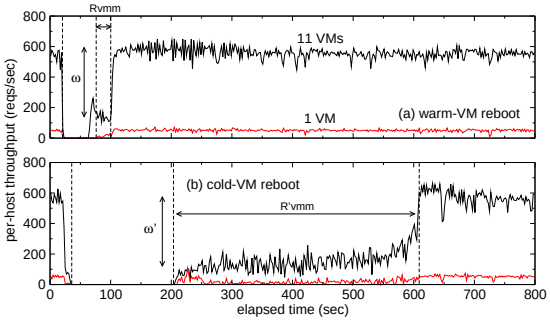


図 3　VMM のソフトウェア若化の間のスループットの変化

図 3 は Warm-VM Reboot および Cold-VM Reboot を行った時の各ホストのスループットの変化を示している。各ホストのスループット p は通常時で毎秒 560 リクエストであった。1 つの VM のスループットも示している。図 4 は OS のソフトウェア若化を行った時のホストのスループットの変化を示している。

Warm-VM Reboot によるダウンタイム $D_w(11)$ は 59 秒であった。ホストのスループットはすべての VM がレジュームされるまで低下していた。すべての VM のレジュームを完了するまでには VM 間のリソース競合により時間がかかっている。この性能低下率 ω

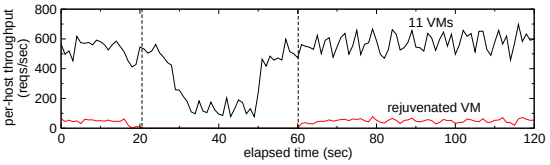


図 4　OS のソフトウェア若化の間のスループットの変化

は 0.61 であり、その期間 R_{vmm} は 25 秒であった。一方、Cold-VM Reboot によるダウンタイム $D_c(11)$ は 219 秒であった。すべての OS が起動された後、キャッシュミスのためにホストのスループットは大きく低下していた。この性能低下率 ω' は 0.72 であり、その期間 R'_{vmm} は 397 秒であった。

OS のソフトウェア若化によるダウンタイム D_{os} は 41 秒であった。OS を再起動している間、他の OS のスループットは大きく低下していた。この期間の合計は $(n-1)D_{os}$ である。一方、OS の再起動後のスループットは低下しなかった。これは毎秒 50 リクエストではキャッシュミスによるスループットの低下が起こらなかったためである。これらの結果から、性能低下率 δ は 0.35 であり、その期間 R_{os} は 410 秒であった。

4.2　VM 移送

通常の移送とライブマイグレーションを行った際のトータルスループット P_m と P_l を見積もるために、Xen の VM 移送を行っている間の移送元ホストと移送先ホストでの VM のスループットの和を測定した。移送元ホストとしては上の実験と同じ PC を用い、移送先ホストとして、デュアルコア Xeon 3.0 GHz の CPU を 2 基、PC2-5300 DDR2 SDRAM を 12 GB、15,000 rpm の SAS ディスク、ギガビットイーサネットを搭載した PC を用いた。移送先ホストのハードウェア構成に上の実験で用いた Xen 3.0.0 が対応していなかったため、Xen 3.1.0 を用いた。VM の移送を行えるようにするため、移送元ホストでも Xen 3.1.0 を用いた。両ホストで共有するディスクとして NFS サーバを用いた。NFS サーバマシンとして、Core 2 Duo E6700 の CPU を 1 基、PC2-6400 DDR2 SDRAM を 2 GB、SATA ディスク、ギガビットイーサネットを搭載した PC を用いた。これらの PC はギガビットイーサネットスイッチで接続した。

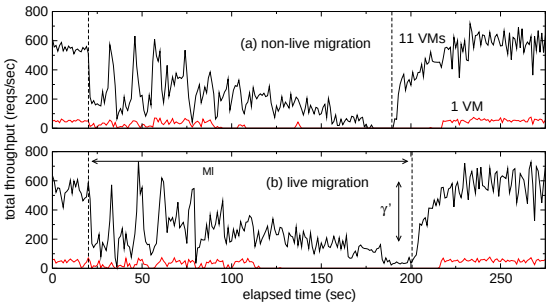


図 5　通常の移送とライブマイグレーションの間のスループットの変化

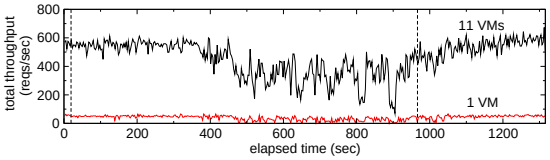


図 6　転送レート制御を行った際のスループットの変化

図 5 は通常の移送またはライブマイグレーションを行った場合のトータルスループットを示している。図 6 は適応的な転送レート制御を行いながらライブマイグレーションを行った場合について示している。Xen の転送レート制御は、100 Mbps の転送レートから始め、移送中の VM のメモリ内容が変更された割合に従って 500 Mbps まで増加させる。

通常の移送では、すべての VM の移送が終わるとスループットは回復しているが、VM の移送中はスループットが大きく低下した。これはすべての VM の移送が終わるまで、移送した VM のスループットがほぼ 0 になっているためである。この原因は移送先ホストは移送の処理を行うために CPU 時間のほとんどを使ってしまっているためと考えられる。その結果、移送元ホストで動いている VM の数が減るにつれて、スループットが低下している。総移送時間 M_t は 165 秒であり、各 VM の平均ダウンタイム D_m は 15 秒であった。

ライブマイグレーションにおけるスループットの変化は通常の移送の場合と非常によく似ている。違い

は移送している間 VM が動いていることである。このオーバヘッドのために、ライブマイグレーションが完了するのにかかる時間は通常の移送よりも少し長くなり、総移送時間 M_l は 181 秒であった。一方、適応的な転送レート制御を行った場合は、スループットの変化は大きく異なる。移送開始後 350 秒の間は性能低下は見られないが、その後は大きく性能が低下している。この制御により、総移送時間 M_l は 950 秒に増加した。

表 1 は i 番目の VM が移送されている間の、移送元ホストで動いている VM のスループット $V_s(i)$、移送中の VM のスループット $V_m(i)$、移送先で動いている VM のスループット $V_d(i)$ を示している。また、i 番目の VM の移送時間 $M(i)$ も示した。これらは実験結果を基に関数フィッティングを行ったものである。これらの結果から、通常の移送を行った場合の性能低下率 γ は 0.60、ライブマイグレーションを行った場合の性能低下率 γ' は 0.57 となる。また、適応的な転送レート制御を行うと γ' は 0.23 に改善している。

4.3 比較

トータルスループットをクラスタ内のホスト数 m に関して比較してみる。トータルスループットは m に依存して増加して比較しにくいため、トータルスループットを m ホストでの最大トータルスループットで割った値を性能指標として用いる。最大トータルスループットとは、ソフトウェア若化を行わず、すべてのホストがサービスを提供する状況でのトータルスループットであり、mp となる。図 7 は P_w、P_c、P_m、P_l についてこの性能指標をプロットしたものである。P_l については転送レート制御を行った場合と行っていない場合の両方を示してある。T_{vmm} は 28 日、T_{os} は 7 日と仮定した。

Warm-VM Reboot を用いる場合、性能指標はホスト数に依存せず常に最高であった。一方、移送を用いる場合はホスト数に依存する。ホスト数が約 11,000 を超えると、Cold-VM Reboot を用いるよりは移送を用いたほうが性能がよくなることが分かる。しかし、一般的なホスト数では移送を用いないほうが性能がよくなる。

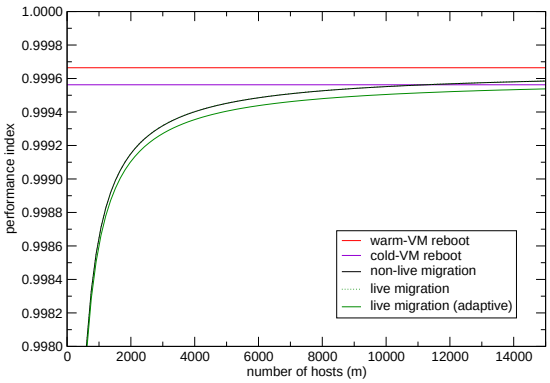


図 7　様々な手法におけるトータルスループットの比較

この分析から、Warm-VM Reboot はクラスタ環境でも有用であることが分かった。VMM のソフトウェア若化が行われるホストのダウンタイムを削減し、ソフトウェア若化後の性能低下を防ぐことで、高いトータルスループットを保つことができる。一方、複数のホストに複製できないサービスに関しては、ライブマイグレーションが有用である。他のホストをスペアとして使うことでダウンタイムの発生を防ぐことができる。

5 関連研究

マイクロリブート [1] はソフトウェア若化を行うためにアプリケーションのコンポーネントだけを再起動することができる。小さいコンポーネントの再起動でソフトウェア若化が行えなければ、そのコンポーネントを含むもう少し大きなコンポーネントを再帰的に再起動する。より小さいコンポーネントの再起動だけでソフトウェア若化が行えれば、アプリケーションのダウンタイムを減らすことができる。同様に、マイクロカーネル OS ではプロセスとして実装されている OS のサブシステムだけを再起動することができ、Nooks [8] では OS のデバイスドライバだけを再起動することができる。これらはサブコンポーネントを高速に再起動する手法であるのに対し、Warm-VM Reboot はサブコンポーネントの状態を保持しながら親コンポーネントを高速に再起動する手法である。

Xen のデバイスドライバのソフトウェア若化を高速化するために、デバイスドライバをドライバドメイ

表 1　i 番目の VM を移送している間のスループットおよび移送時間

	$V_s(i)$	$V_m(i)$	$V_d(i)$	$M(i)$
通常の移送	$4.4i+21$	-	22 $(i=2)$ $e^{-0.71i+5.7}$ $(i>2)$	$0.61i+12$
ライブマイグレーション	$4.2i+19$	$3.9i+19$	19 $(i=2)$ $e^{-0.57i+5.2}$ $(i>2)$	$0.57i+13$
ライブマイグレーション （転送レート制御あり）	50 $(i<6)$ 34 $(i\geq 6)$	48 $(i<6)$ 29 $(i\geq 6)$	48 $(i<5)$ 30 $(i\geq 5)$	$-0.55i+89$

ンと呼ばれる VM で動作させる手法が提案されている [4]。通常、Xen のデバイスドライバはドメイン 0 で動作するため、デバイスドライバのソフトウェア若化を行うにはドメイン 0 の OS の再起動が必要となる。Xen においてはこれはシステム全体の再起動となるため、ソフトウェア若化によるダウンタイムが長くなる。一方、ドライバドメインを用いれば、デバイスドライバのソフトウェア若化はドライバドメインの OS を再起動するだけで済む。しかし、VMM の他の部分のソフトウェア若化を行う際にはドライバドメインの OS の再起動も必要となるため、ダウンタイムは長くなる。

6　まとめ

本論文ではクラスタ環境でソフトウェア若化を行う際のトータルスループットについて考察した。VMM のソフトウェア若化だけでなく OS のソフトウェア若化も考慮したモデル化を行い、そのモデルに基づいてダウンタイムや性能低下率を用いてトータルスループットを定義した。Warm-VM Reboot を用いる場合と用いない場合、さらに、VMM のソフトウェア若化の前に VM の移送を行う場合の比較を行うために、実験を行ってダウンタイムや性能低下率を見積もった。その結果、Warm-VM Reboot を用いる場合のトータルスループットが常に最大になることが分かった。今後の課題は、実際のクラスタ環境でソフトウェア若化を行い、トータルスループットを直接測定して比較を行ってみることである。

参考文献

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

[2] Castelli, V., Harper, R., Heidelberger, P., Hunter, S., Trivedi, K., Vaidyanathan, K., and Zeggert, W.: Proactive Management of Software Aging, *IBM J. Research & Development*, Vol. 45,No. 2(2001), pp. 311–332.

[3] Clark, C., Fraser, K., Hand, S., Hansen, J., Jul, E., Limpach, C., Pratt, I., and Warfield, A.: Live Migration of Virtual Machines, *Proc. Symp. Networked Systems Design and Implementation*, 2005, pp. 1–11.

[4] Fraser, K., Steven, H., Neugebauer, R., Pratt, I., Warfield, A., and Williamson, M.: Safe Hardware Access with the Xen Virtual Machine Monitor, *Proc. Workshop on Operating System and Architectural Support for the on demand IT InfraStructure*, 2004.

[5] Garg, S., Huang, Y., Kintala, C., and Trivedi, K.: Time and Load Based Software Rejuvenation: Policy, Evaluation and Optimality, *Proc. Fault Tolerance Symp.*, 1995, pp. 22–25.

[6] Huang, Y., Kintala, C., Kolettis, N., and Fulton, N.: Software Rejuvenation: Analysis, Module and Applications, *Proc. Int'l Symp. Fault-Tolerant Computing*, 1995, pp. 381–391.

[7] Kourai, K. and Chiba, S.: A Fast Rejuvenation Technique for Server Consolidation with Virtual Machines, *Proc. Int'l Conf. Dependable Systems and Networks*, 2007, pp. 245–254.

[8] Swift, M., Bershad, B., and Levy, H.: Improving the Reliability of Commodity Operating Systems, *Proc. Symp. Operating Systems Principles*, 2003, pp. 207–222.

[9] Vaidyanathan, K., Harper, R., Hunter, S., and Trivedi, K.: Analysis and Implementation of Software Rejuvenation in Cluster Systems, *Proc. ACM SIGMETRICS Int'l Conf. Measurement and Modeling of Computer Systems*, 2001, pp. 62–71.