クラウドコンピューティングにおける
メモリ資源の動的な調整に関する研究

2012SE135 リョウ ガイ　2011SE021 コウ ウン
指導教員：宮澤 元

1 はじめに

クラウドコンピューティング（クラウド）では、非常に多数の利用者からの要求を効率的に処理するために、仮想マシンを利用している。利用者からの要求を処理する仮想マシンを必要に応じて複数の物理マシン間で移動させることにより、データセンタ内における負荷分散を実現できる。また、こうした仮想マシンの負荷が物理マシンの処理能力に比べて十分低ければ、複数の仮想マシンを同一物理マシンに集約することにより、CPU やメモリ、ネットワークなどの資源の利用率を大幅に高めることができる。

しかし複数の仮想マシンを一台の物理ノードに集約すると、各仮想マシンのメモリ要求の変動により、局所的には物理マシン単位でメモリが不足することがある。従来の解決方法の一つはメモリリソースを静的に割り当てることである。具体的には、各仮想マシンのピーク需要に応じてにメモリ資源を割り当てる。この方法では、各仮想マシンに常に必要十分なメモリを確保できるのでメモリ不足による性能低下を避けられるが、メモリ使用率が低下してしまう問題がある。もう一つの方法は、利用者の需要に応じて割り当てるメモリ量を動的に調整する方法である。たとえば、Xen[1]、vSphere[2],KVM[3]などではメモリバルーニング[5]などの割り当てメモリ量を動的に調整する仕組みを提供している。しかし、これらの方法では、単一物理マシン上で仮想マシンのメモリ需要を調整できるがメモリ不足の問題を根本的に解決することができない。単一の物理マシン上でメモリが不足していても、データセンタ全体としてはメモリに余裕があることも考えられるが、これらの手法で、複数の物理マシン間でメモリ需要をバランシングして、クラウド上の仮想マシン全体の性能を向上することは考えられていない。

本研究の目的はクラウド上の各仮想マシンをそのメモリ要求に基づいて適切な物理マシンにスケジューリングし、動的にメモリ資源を割り当てることである。バルーンドライバを用いて各物理マシンでメモリ使用量を監視し、監視した情報に基づいて仮想マシンの複数物理マシン間のスケジューリングを行うとともに、各物理マシン上で仮想マシンのメモリ使用状況や最大メモリ量などの情報を用いて、仮想マシンに割り当てたメモリ資源を動的に調整する。提案手法を用いることで、メモリ資源の利用率が向上することと、仮想マシンにメモリ資源を適切に割り当てることができるかどうかを、実験で確認する

2 背景

本節では、研究の背景技術として、仮想マシンやクラウドで従来から用いられているメモリ資源の動的調整メカニズムについて述べる。

2.1 メモリバルーニング

Xen, VMware[4]と vSphere などでは、メモリバルーニングと呼ばれる仮想マシンのメモリ割り当てを動的に調整する仕組みを用意している。メモリバルーニングでは、仮想マシンの OS で特殊なデバイスドライバ（バルーンドライバ）を動作させ、メモリ使用状況を監視する。バルーンドライバは必要に応じて自らが仮想マシン上のメモリを確保したり解放したりして仮想マシンの実質的なメモリ使用量を動的に調整することができる。バルーンドライバはハイパーバイザと連携しているので、バルーンドライバが確保したメモリは同一物理マシン上の他の仮想マシンで利用できる。

2.2 単一物理マシン上のメモリ割当の動的調整

仮想マシンのメモリが不足した際にプロセスが強制終了されることを防ぐためのシステムとして CUDSwap がある[7]。CUPSwap は、仮想マシン上でメモリの使用状況を監視して、メモリが不足しそうな際には swap パーティションを動的に追加する。これにより、仮想マシン上で利用可能な仮想メモリ領域が大きくなるので、メモリ不足によって仮想マシン上のプロセスが強制終了されることを防ぐことができる。CUDSwap では仮想マシン上での仮想メモリの不足を解消することができるが、仮想マシンに割り当てた物理メモリ不足を解消する効果はない。

文献[8]では、仮想マシン上での物理メモリ不足に対応するために、仮想マシンに対するメモリ割当を動的に最適化する DMSS を提案している。DMSS では、すべての仮想マシン上のゲスト OS にモニタを配置することにより、

各仮想マシンのメモリ使用状況を取得する。この情報に基づいて、各仮想マシンに対するメモリ割当を最適化する。

DMSS では、単一物理マシン上で、仮想マシンに対するメモリ割当を最適化することはできるが、物理マシン上の物理メモリが不足するような状況には対処できない。

3　クラウドにおけるメモリ資源の動的調整システム

本節では、我々が提案するクラウド上で動作する仮想マシンに対するメモリ資源の割り当てを動的に調整するシステムについて述べる。提案したシステムは主に、各物理マシンで動作するメモリ監視モジュール（monitor)と、システム全体で一つ動作するメモリ割り当て動的調整モジュール（Scheduler)の二つのモジュールからなる。Monitor はその物理マシンで実行している各仮想マシンを監視し、メモリ使用状況をリアルタイムで取得し、Scheduler に送信する。Scheduler は Monitor から送信された各仮想マシンのメモリ使用状況や最大メモリサイズなどの情報を用いて、各仮想マシンへの割当メモリ量を調整する。

割り当てメモリ量の調整は以下のように行われる（図1)。Monitor によって監視した各物理マシンのメモリ使用率が90%未満だった場合、物理マシンのメモリには余裕があると判断する。この場合、各仮想マシンのメモリ使用率を参照して90%以上であれば、その仮想マシンがメモリ不足であると判断し、その仮想マシンに割り当てるメモリ量を2倍にする。仮想マシンのメモリ使用率が40%以下だったら逆に余っているメモリを回収する。一方、物理マシンのメモリ使用率が90%以上だったら、物理マシンがメモリ不足であると判断し、仮想マシンのマイグレーションを行う。この場合、対象の物理マシン上でメモリ使用率が最も高い仮想マシンをほかのメモリ使用率に余裕がある物理マシンへマイグレートさせる。

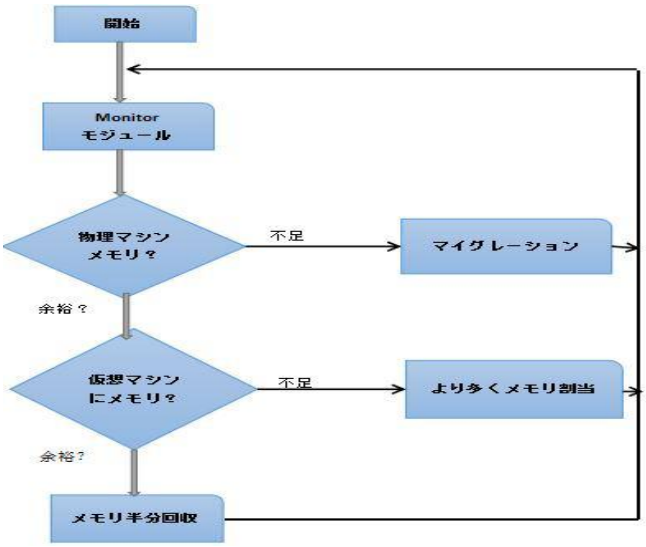

図1. 割当メモリ量調整の流れ

4　実装

提案のシステムのプロトタイプを Xen 上の libvirt ライブラリおよび libxc ライブラリを用いて実装した。

表 1　実験用 PC の仕様

CPU	Intel(R)Core i7-4770
コア数	4
メモリ	2GB
OS	Ubuntu 64 bit Server 16.04
クラウド基盤ソフトウェア	Eucalyptus 3.1

4.1　メモリ監視モジュール

Monitor モジュールは各仮想マシンのメモリ使用状況を常に監視するために、仮想マシンの ID を用いて、物理マシン上で動作するすべての仮想マシンを調べる。libxc ライブラリ関数を用いて仮想マシンの current memory と max memory という情報を取得する。

4.2　Scheduler モジュール

クラウドのすべての物理マシンで動作する仮想マシンのメモリ割当を動的に調整するために、Scheduler モジュールは Monitor モジュールから受け取った各物理マシン、各仮想マシンのメモリ情報を利用して、適切な仮想マシン配置を計算する。

計算結果に基づき、libxc 関数を用いて各仮想マシンのメモリ割り当てを調整するとともに、必要に応じて各仮想マシンを適切な物理マシンにマイグレートさせる。

5　実験

本提案システムの効果と性能を確認するために、実装したプロトタイプを用いて実験を行う。実験環境として表1に示す仕様の PC を2台用意して、1000Base-T ギガビットイーサネットで接続した。1台の物理マシン上で二つの仮想マシン VM1,VM2 を作成した。もう1台の物理マシンは、マイグレーションを行う際にターゲットマシンとして用いる。提案システムの有効性を確認するために Apache のストレステストソフトウェア JMeter とアプリケーションサーバー Tomcat を用いた実験を行う。JMeter は多数のクライアントからのリクエストをエミュレートし、Tomcat サーバのメモリ使用状況を変化させる。物理マシンのメモリに余裕がある状態と物理マシンのメモリが不足している状態の二通りの状況で実験を行った。

5.1 物理マシンのメモリに余裕がある状態

物理マシンのメモリに余裕がある状態で提案したシステムの有効性を確認するため、二つの仮想マシンに別々に 512MB のメモリを割り当てた。VM1 に Tomcat アプリケーションサーバと JMeter ストレステストソフトウェアをインストールした。実装した提案システムのプロトタイプを用いて各仮想マシンのメモリ使用状況を記録する。次に JMeter のシミュレーションスレッドを増やし続けて VM1 の使用メモリを増加させていく。VM1 のメモリ使用率がある限界に達したら、JMeter のシミュレーションスレッドを減らし始めて、実験の各時間における VM1 の使用メモリ量と最大メモリ量を記録する。実験結果のグラフを図3に示す。

横軸は時間を、縦軸はメモリ量を指している。赤い線は VM1 の使用メモリの量の変化をあらわしている。青い線は VM1 の最大メモリ容量の変化をあらわしている。JMeter のシミュレーションスレッドの増加にともない、VM1 の使用しているメモリも増えている。時刻t7に VM1 の使用してるメモリが最大メモリ容量(512MB)に近くなったとき、最大メモリ容量が 1024MB に増加している。時刻 t9 で JMeter のシミュレーションスレッドを減らすと、VM1 の使用しているメモリも減っている。時刻 t12 と t13 で、VM1 のメモリ使用率が 40%以下になった時、VM1 の余っているメモリの半分を提案システムが回収している。この結果から、物理マシンのメモリに余裕がある状態で提案システムが有効に動作していることがわかる。

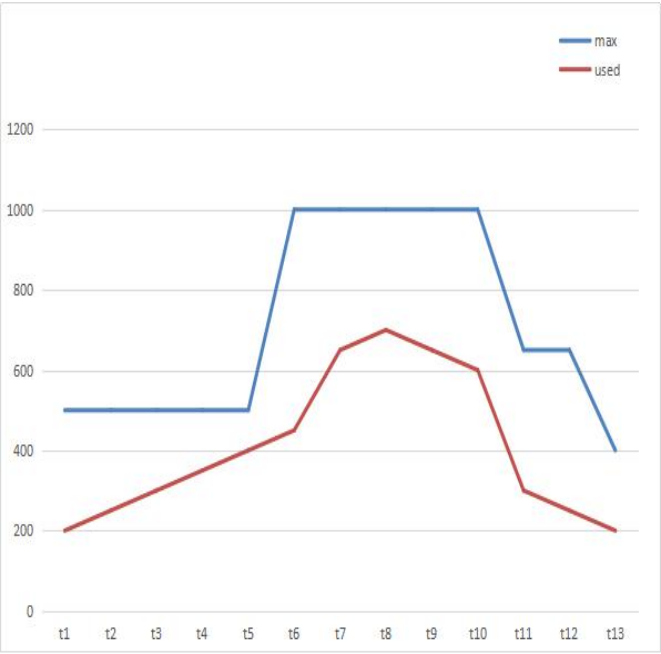

図3 VM1 の使用メモリ量と最大メモリ容量の変化

5.2 物理マシンのメモリが不足している状態

物理マシンのメモリが不足している状態での提案システムの有効性を確認するために実験を行なった。

5.2.1 Tomcat の平均応答時間

各仮想マシンに 1000MB のメモリを割り当て、Tomcat アプリケーションサーバを実行させる。JMeter ストレステストソフトウェアからシミュレーションスレッドを 10 個起動して Tomcat にリクエストを送信し続けることで、Tomcat の使用メモリを増加させてメモリ不足の状態を発生させる。VM のマイグレーションを使用する場合と使用しない場合で、Tomcat の平均応答時間を比較した。実験結果を図4に示す。横軸は実験回数、縦軸は応答時間(ms)を表す。

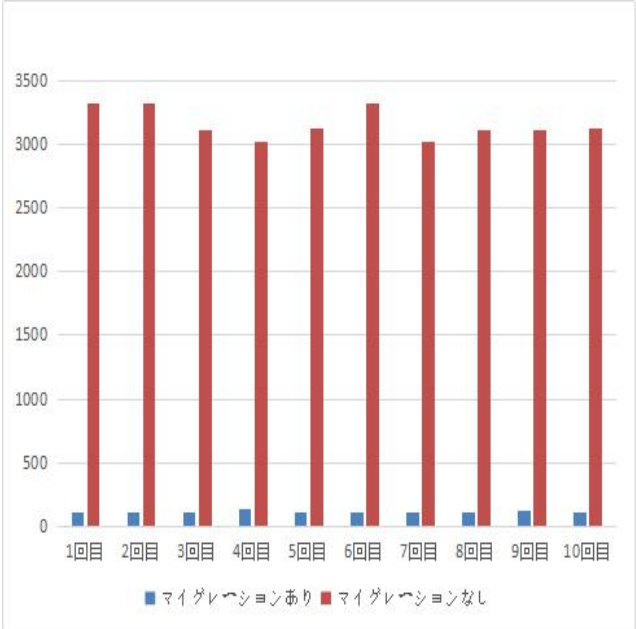

図 4 、Tomcat の平均応答時間

マグレーションを行う場合の平均応答時間が 117ms であるのに対し、マイグレーションを行わない場合の平均応答時間は 3214ms だった。

この結果から、物理マシンのメモリが不足する状態では、仮想マシンをマイグレーションさせることにより、仮想マシンで動作するアプリケーションの実行性能を改善させるとこができる場合があるとこがわかる。

5.2.2 仮想マシンのメモリ回収による影響

物理マシンのメモリが不足している状態で、使用メモリ量が少ない仮想マシンからメモリを回収した場合のサービス性能に対する影響を調べる実験を行った。

JMter で 10 個のシミュレーションスレッドを動作させ、VM1 上の Tomcat にリクエストを送信する。このような負荷がかかっている状態で、VM1のメモリを回収する。回収したメモリ量と、回収前後での応答時間の違いを測定

した。図5に実験結果を示す。グラフの横軸が回収したメモリ量、縦軸はメモリ回収後の VM1の応答時間に対するメモリ回収前の応答時間の比率である。例えば 50MB を回収した場合、VM1 の応答時間は 300ms,回収前の応答時間は 280ms,その比率は 280/300 なので、93%となる。なお、VM1への割り当てメモリ量は 512MB である。

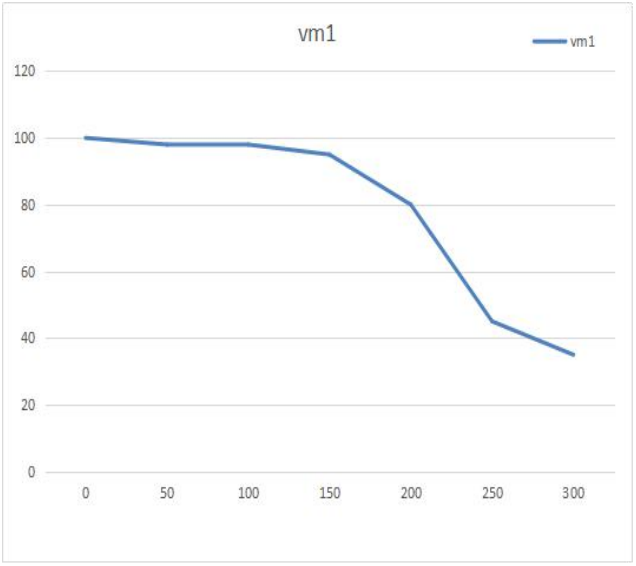

図 5　メモリ回収後の Tomcat の応答時間のメモリ回収前に対する比率

実験結果から見ると、最大メモリが 512MB の仮想マシンでは、回収するメモリ量が 150MB より小さければ、応答時間にはほとんど影響しないが回収メモリ量が 150MB より大きい場合、応答時間への影響が大きくなる。提案したシステムを用いて仮想マシンへのメモリ割当を調整する場合には、余っているメモリの 50%を回収することでの仮想マシンの性能にたいする影響は無視していいと推測できる。

6　おわりに

クラウドコンピューティングが発展するのにともない、クラウド上での仮想マシンのメモリ割当は大きな問題になっている。メモリなどの資源を要求に応じて動的に割り当てあることにより、クラウド上の資源利用率とシステム全体でのサービス性能を向上して、クラウドの利用コスト低減につながる可能性がある。本稿では、クラウドにおけるメモリ資源の割当を動的に調整する手法を提案した。libvirt ライブラリおよび libxc ライブラリを用いて仮想マシンのメモリ使用量の監視と動的な仮想マシンのスケジューリングを実現することにより、提案システムのプロトタイプを実装した。プロトタイプを用いた実験により、提案手法がクラウド上のメモリ割当問題の解決に有効であることがわかった。一方、クラウド上で利用される資源はメモリだけではなく、CPU やネットワーク、電気エネルギーなど様々なものがある。今後は、メモリだけでなく、こうした様々な資源を考慮した仮想マシンスケジューリングシステムの実現を目指す。

参考文献

[1] Steven Hand, Jacob Gorm Hansen, “The powerful opensource industry standard for virtualization” . http：//www. xen. org/. 2014.

[2] Dutch Meyer, “The virtuabzation platform for building cloud infrastructures”, http : //www.vmware.corn/products/vsphere. 2014.

[3] Brendan Cully, Geoffrey Lefebvre, “A full virtualization solution for Linux on x86 hardware containing virtualzation extensio”, http://www.linux-kvm.org/.2014.

[4] Ardalan Kangarlou,Dongyan, “VMware virtualization Software for Desktop ” Servers&Virtual Machines for Public and Private Cloud Solutions.http：//www. VMware. com/. 2014.

[5] Carl A.Waldspurger.“Mermory Resource Management in VMware ESX-Server.” In Proceedings of the 5 th Symposium on Operating Systems Design and Implementation (OSDI 2002).pp.181-194, 2002.

[6] Pin Lu and Kai Shen, “Virtual Machine Memory Access Tracing with Hypervisor Exclusive Cache.” In Proceedings of the 2007 USENIX Annual Technical Conference,pp.29-43, 2007.

[7] Jose Antonio Navas Molina,et al., “Addressing Virtual Machines in a Cloud Environment,” In Proceedings of the 43 rd IEEE/IFIP International Conference on Dependable Systems and Networks(DSN),pp.1-6, 2003.

[8] Lanzheng Liu,et al., “DMSS:A Dynamic Memory Scheduling System in Server Consolidation Environments” In Proceedings of the 15the IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops,pp.70-75, 2012.