Unikernel クラウド向けの VM グループ管理

2011SE252 田尻 翔太　2011SE269 遠山 恭平
指導教員：宮澤 元

1　はじめに

近年，ネットワーク経由でサービスを提供する，クラウドコンピューティング (クラウド) が普及している．クラウドでは，データセンタに設置された複数のサーバ計算機を多数のユーザが利用する．利用するユーザの多様さから，幅広い要求に対応することが求められる．そのため，CPU やメモリ，ストレージなどの物理的な資源を，その物理的構成にとらわれずに柔軟な形で提供したり，限りあるそれら物理資源を効率的に活用したりする必要がある．そこで，クラウドでは仮想化技術が取り入れられている．

仮想化技術を用いると，物理資源をその実構成にとらわれずに管理することができる．仮想化を実現するには，仮想マシンモニタ (VMM:Virtual Machine Monitor) が利用される．VMM 上では，仮想マシン (VM:Virtual Machine) が動作し，そこで通常の OS やアプリケーションを動かすことができる．

仮想化技術を用いる際の課題として，VM のパフォーマンスの問題がある．例えば，VMM が VM をスケジューリングし，VM がさらに内部で動いているプロセスのスケジューリングを行うといった，スケジューリング処理の多重化によるオーバヘッドが生じる．また，ディスクやネットワークアクセス時に発生する I/O（Input/Output）エミュレーションでオーバヘッドが生じる．こうしたオーバヘッドは，VM のパフォーマンスに悪影響を及ぼす．

VM のパフォーマンスを改善するため，様々な研究が行われており，Unikernel[1] もそのような研究の 1 つである．Unikernel は，VM 上で動作する OS を Library OS を活用して軽量化するとともに，単一の VM で単一のサービスを提供するものである．これによって VM 内でプロセスのスケジューリングを行うために起こる，スケジューリングの多重化を解消できる．同時に，VM 内でスケジューリングを行うプリエンプティブスレッドは提供しない．

一方，大規模な並列処理を行うためにクラウドを用いることがある．例えば，クラウド上の VM で Hadoop[2] を用いて大規模計算を行うといったように，並列処理を利用することは少なくない．Unikernel では，並列処理が必要な場合，複数の VM を用いて VM 間通信を行っている．しかし，VM 間通信を行うとアドレス空間を頻繁にまたぐ必要があるので，余計なオーバヘッドが生じてしまう．

本稿では，Unikernel のような単一のアプリケーションだけを VM で動作させるようなクラウドにおいて，大規模並列計算を支援するための VM のグループ管理について述べる．並列処理を行っている VM のグループ化を行うことにより，グループ化された VM のスケジューリングやメモリ割り当てを最適化可能である．

2　クラウドにおける仮想化環境

IaaS クラウドを構築するためにさまざまな仮想化環境が提案され，利用されている．本節ではこのような仮想化環境のうち Mirage OS(Mirage)，Xen，およびコンテナ型仮想化について述べる．

2.1　Mirage

Mirage は Unikernel の実装の 1 つで，モバイルプラットフォームや様々なクラウドコンピューティングで動作するセキュアでハイパフォーマンスな OS である [1, 3]．Mirage 向けのアプリケーションは関数型言語 OCaml で開発される．ソフトウェアスタック全体，言語ランタイムおよびシステムライブラリはハイパーバイザ上で直接動作するブート可能な VM イメージにコンパイルされる．図 1 のように Mirage は既存の VM と比較すると様々なソフトウェア層を省略していることがわかる．

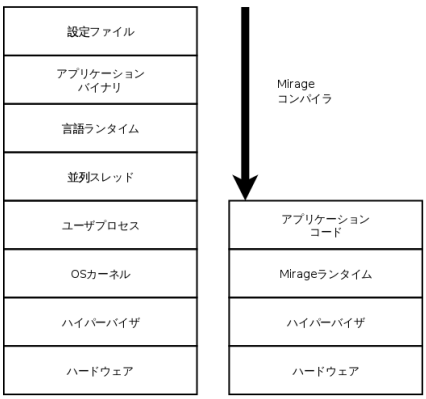

図 1　VM のソフトウェア層 (左:既存の VM．右:Mirage)

2.2　Xen の概要

Xen[4] はオープンソースの VMM の 1 つである．Xen にはコントロールドメインと呼ばれる Domain-0 と，ゲストドメインと呼ばれる Domain-U の 2 種類の VM が存在する．それぞれが VMM によって仮想化され，資源の提供を受けて動作している (図 2)．

Xen の仮想化の方式には以下の 2 種類の方式がある

完全仮想化　ゲスト OS のコードを修正せず,Intel VT や AMD-V といった CPU の仮想化支援機能を利用し OS を動作させる．

準仮想化　Xen 上で動作させるためにゲスト OS のコードを修正し，ハードウェアエミュレーションのオーバーヘッドを減らす．

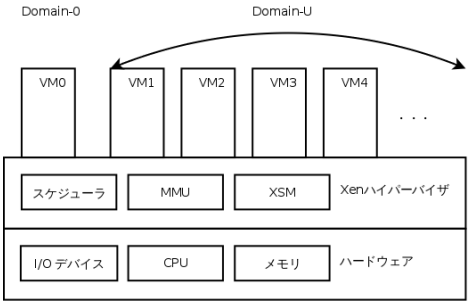

図 2 Xen の構成要素

2.2.1 Xen の共有メモリ

Xen が従来持つ共有メモリを提供する機能として Grant Mechanism [5] がある．Grant Mechanism は，VM と VMM で共有される grant reference を利用してメモリの共有を行う．grant reference には，共有するメモリページのアドレスや，相手の VM の ID などの情報が含まれている．相手の VM が共有されたページにアクセスするときに，ページ所有者が変更されることはないので，安全に処理が行われる．Grant Mechanism は，Domain-0 と Domain-U 間で処理が分かれる場合など，VM 間で通信が必要な場合に，処理の効率化のため用いられる．

2.3 コンテナ型仮想化

VMM のオーバヘッドのない Linux Containers(LXC) [6] や Docker[7] などのコンテナ型仮想化が開発されている．起動する全てのプロセスは物理マシンにインストールされた OS 上で動作する．そのプロセスの一部を図 3 のようにグループ化し，他のグループやグループに属していないプロセスから隔離した空間で動作させる．コンテナ型仮想化では OS は固定され，プロセスごとに様々な OS を利用することはできない．また，現状の Linux Kernel ではプロセスのマイグレーションもサポートされておらず，ハードウェアをまたぐようなコンテナを利用することはできない．

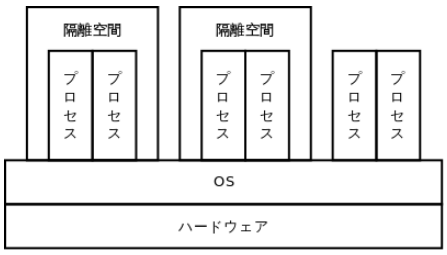

図 3 コンテナの仕組み

3 Unikernel クラウド向けの VM グループ管理の設計

我々は，Unikernel 型のクラウドの VMM において大規模並列計算のサポートを行うために VM をグループとして管理する機構を導入する．本節では VM のグループ化，VM グループを考慮した VM スケジューリングと VM へのアドレス空間の割り当てについて述べる．

3.1 VM のグループ化

Unikernel 型のクラウドで並列処理を行う場合，それぞれ単一スレッドを動作させる複数の VM が互いに通信を行う．このように関係の強い VM をグループとしてまとめて管理することで，VM のスケジューリングと VM へのアドレス空間の割り当てを VM グループを考慮して行うことができる．

図 4 の例では，VM0，VM1，VM2 の 3 つの VM が起動しており，このうち VM1 と VM2 が互いに通信を行っている．VM1 と VM2 をグループとしてまとめ，1 つの VM グループとして VM のスケジューリングとアドレス空間を割り当てる．

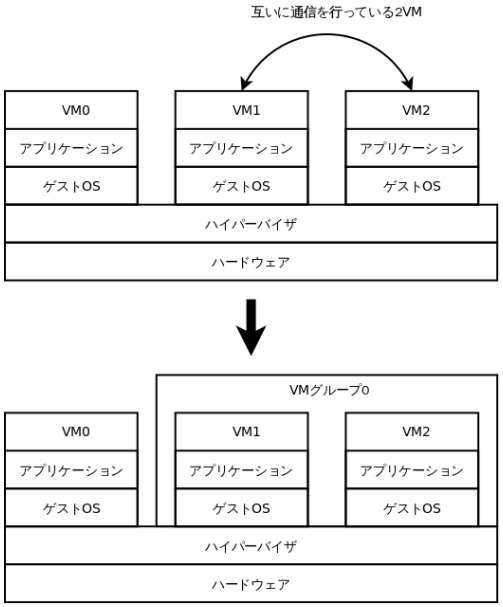

図 4 関係の強い VM をグループ化する

3.2 VM グループを考慮した VM スケジューリング

Unikernel 型のクラウドでは 1 つの VM が 1 スレッドと考えられるが，各 VM の仮想 CPU に物理 CPU が VM スケジューラによってバラバラに割り当てられる．VM をグループ化した場合，図 5 のように，VM のグループを複数 CPU が割り当てられた 1 つの VM と見なし,VM スケジューリングを行う．

3.3 共有アドレス空間の提供

Unikernel 型のクラウドにおける現状のメモリ割り当ては，VM をプロセスのように扱っているとも考えられる．VM はそれぞれ独自のアドレス空間を所有し，その上で処理を行う．この方法では，並列処理を行う際に，アドレス空間の切り替えに伴うオーバヘッドが生じてしまう．図 6 は，通常の VM へのメモリ割り当てを表している．各 VM

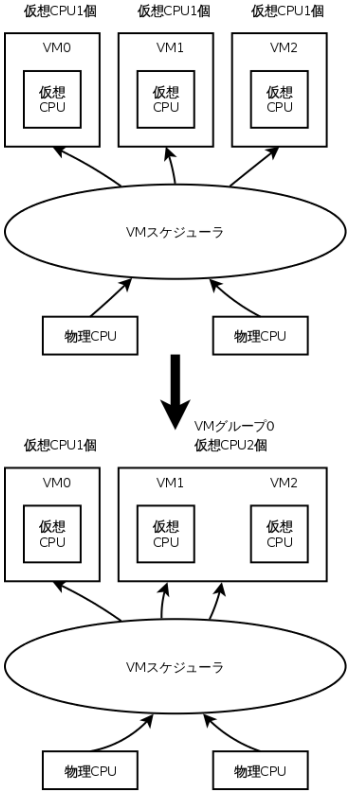

図 5 VM グループを考慮した VM スケジューリング

のアドレス空間には，それぞれ別の物理メモリが割り当てられる．Xen の Grant Mechanism を使えば，メモリの一部を他の VM に見せることができるが，アドレス空間をまるごと共有することは考えられていない．

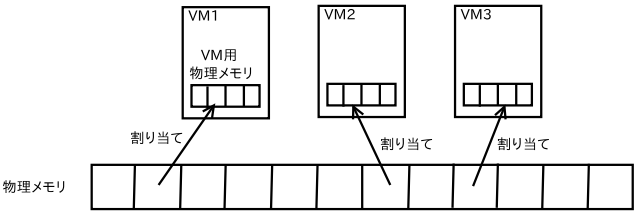

図 6 VM グループのない場合のアドレス空間の割り当て

我々は，VM グループを考慮したメモリの割り当てを行う．同一の VM グループに属する VM 同士でアドレス空間を共有することにより，各 VM をスレッドのように扱うことができる．アドレス空間が共有されるので，アドレス空間の切り替えによるオーバヘッドを削減できる．図 7 は，VM グループを考慮した場合のメモリ割り当ての例である．同じグループに属する VM は，アドレス空間が共有される．Unikernel のように単一 VM で単一サービスを提供する仕様と合わせると，VM がスレッドのように扱われることになる．

4 VM グループの実装

3.1 節の VM のグループ化と 3.2 節の VM グループを考慮した VM スケジューリングを Xen のバージョン 4.4.1

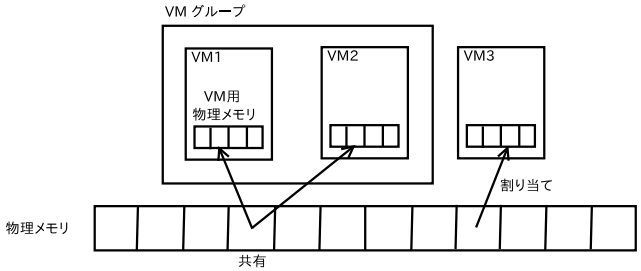

図 7 VM グループを考慮したアドレス空間の割り当て

をベースに行った．なお，3.3 節の共有アドレス空間は未実装である．

4.1 VM のグループ化

VM のグループ化を実装するにあたり，Xen の VM の作成処理と VM の削除処理を拡張した．具体的な実装を以下に示す

- Xen ハイパーバイザに新たに VM グループ構造体を追加した．VM グループは複数ありそれぞれ VM グループ ID を持っている．
- 作成する VM の情報に追加する VM グループの ID を追加し，VM グループ ID を指定した VM の作成を可能にした．実装は xl create コマンドとハイパーコールの処理を拡張する形で行った．
- VM の削除処理を拡張し，VM をグループから外す処理を追加した．VM グループに VM が存在しなくなる場合にはその VM グループを削除する．

4.2 VM グループを考慮したスケジューリング

デフォルトのドメインスケジューラである Credit スケジューラに変更を加えた．Unikernel 型の IaaS クラウドで並列処理を行う場合，各 VM が並列処理のスレッドとして動作し，VM 同士が互いに通信を行っている．グループ内の VM のどれかが動作した後には，VM 間の通信の待ち時間を減らすために，同一グループ内の別の VM が優先して動作するようにスケジューラを拡張した．

5 実験

VM グループを考慮したスケジューラの効果を確認するために実験を行った．具体的には，Hadoop を使ったモンテカルロ法により円周率の近似値を算出するアプリケーション [8] を動作させ，その計算時間を測定する．

5.1 実験環境

表 1 に示される PC1 台を実験に使用した．並列計算処理を行わせる VM をこの PC の上に 3 つ作成する．作成した 3 つの VM を VM1，VM2，VM3 とした．作成した VM の仕様を表 2 に示す．Hadoop 環境の構築には Cloudera 社の Cloudera Manager[9] バージョン 5.2.1 を使用し，VM1 を Cloudera Manager サーバ，VM2 と VM3 を並列計算処理を行わせるホストとした．

表 1 PC の仕様

CPU	Intel®Core™i7-2600
クロック周波数	3.4GHz
メモリ	8GB
HDD 容量	500GB
コア数	4 コア 8 スレッド
OS	Ubuntu Srver 14.04 64bit

表 2 VM の仕様

	VM1	VM2，VM3
CPU コア数	2	
メモリ	2GB	1.5GB
HDD 容量	32GB	
OS	Ubuntu Srver 14.04 64bit	
Java バージョン	1.6.0_31	

5.2 実験結果

計算プログラムのマップ数と 1 マップ当たりのサンプル数をそれぞれ変化させ，計算時間を測定した．それぞれ 10 回ずつ計測を行い計算時間の平均値を求めた．マップ数が 10 の場合の実験結果を図 8 に，100 の場合を図 9 に示す．

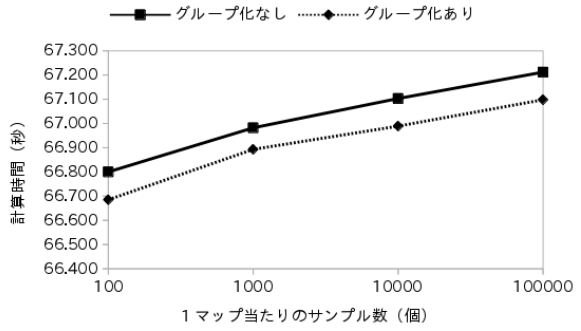

図 8 マップ数 10 の場合の実験結果

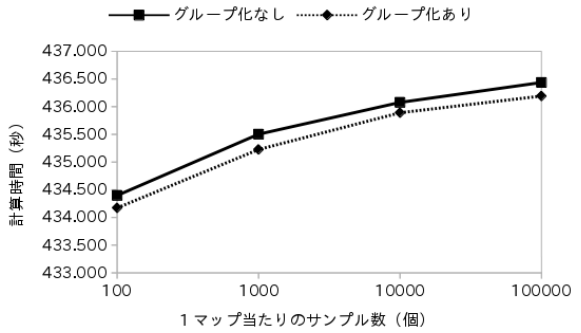

図 9 マップ数 100 の場合の実験結果

マップ数や，1 マップ当たりのサンプル数にかかわらず，3 つの VM が同一のグループに属して処理を行う場合の処理時間が短い．VM グループを考慮した VM スケジューラの効果が期待できることがわかる．

6 まとめと今後の課題

我々は，Unikernel 型の IaaS クラウドにおける大規模並列計算を支援するために，VM のグループ化を提案した．VM をグループ化することで，VM スケジューリングとアドレス空間の割り当てを VM グループを考慮した形で行うことができる．我々が提案する機能を Xen ハイパーバイザに追加することで実装を行った．VM グループを考慮したスケジューラを用いた実験により，スケジューラの効果が確認できた．

今後は，VM グループを考慮した共有アドレス空間を実装すること，実装した機能の有用性を確認するために更に実験を行うことが必要である．また，VM グループの機能をネットワーク環境に対応させるための検討を行う．

参考文献

[1] A. Madhavapeddy, R. Mortier, C. Rotsos, D. Scott, B. Singh, T. Gazagnaire, S. Smith, S. Hand, and J. Crowcroft, “Unikernels: Library Operating Systems for the Cloud,” *SIGPLAN Not.*, vol. 48, no. 4, pp. 461–472, Mar. 2013. [Online]. Available: http://doi.acm.org/10.1145/2499368.2451167

[2] The Apache Software Foundation, “Welcome to Apache™Hadoop®!” [Online]. Available: http://hadoop.apache.org/, 2015 年 1 月 10 日アクセス

[3] “Mirage OS.” [Online]. Available: http://openmirage.org/, 2015 年 1 月 10 日アクセス

[4] “The Xen Project, the powerful open source industry standard for virtualization.” [Online]. Available: http://www.xenproject.org/, 2015 年 1 月 10 日アクセス

[5] K. K. Ram, J. R. Santos, and Y. Turner, “Redesigning xen’s memory sharing mechanism for safe and efficient I/O virtualization,” in *WIOV’10: Proceedings of the 2nd conference on I/O virtualization*. Berkeley, CA, USA: USENIX, 2010.

[6] “Linux Containers.” [Online]. Available: https://linuxcontainers.org/, 2015 年 1 月 10 日アクセス

[7] “Docker - Build, Ship, and Run Any App, Anywhere.” [Online]. Available: https://www.docker.com/, 2015 年 1 月 10 日アクセス

[8] *Hadoop* 徹底入門 第 *2* 版. 翔泳社，2013. [Online]. Available: https://books.google.co.jp/books?id=4AgfBAAAQBAJ

[9] Cloudera, “Cloudera Manager 5.” [Online]. Available: http://www.cloudera.com/content/cloudera/en/documentation/core/v5-2-x/topics/cm_intro_primer.html, 2015 年 1 月 10 日アクセス