IaaS 環境におけるマルチプロセスアプリケーションを考慮した仮想化基盤の動的構成管理に関する研究

M2015SE012 田尻翔太
指導教員：野呂昌満

1 はじめに

インターネット経由でさまざまなサービスを利用者に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．例えば IaaS クラウドでは，仮想化技術を用いて計算基盤そのものをサービスとして提供している．利用者は仮想マシン (VM) 上にオペレーティングシステム (OS) などのシステムソフトウェアを含めて計算機環境を構築し自由にアプリケーションを動作させることができる．例えば，Web サーバのようなアプリケーションを動作させれば，利用者自身がクラウドを用いてサービスを提供することができる．また，複数の VM を利用して Hadoop や Spark といった分散並列計算環境を IaaS クラウドを使って実現することもできる．

IaaS クラウドで VM を利用する際の問題点の一つは，パフォーマンスである．VM を利用する際には，一般に各物理マシン上のハイパーバイザを利用して VM を動作させる．各 VM 上では，OS が動作しており，プロセスのスケジューリングを行っている．そこで，ハイパーバイザの VM のスケジューリングと OS のスケジューリングが二重に行われパフォーマンスを低下させてしまう．

VM のパフォーマンスを改善するためのさまざまな研究が行われている．コンテナ型仮想化 [1, 2] は，ハイパーバイザを用いない通常の OS 上でアプリケーションの実行環境を隔離して動作させる技術である．ゲスト OS を必要としないので二重のスケジューリングが解消されパフォーマンスが向上する．一方 Unikernel[3] では，VM 上のゲスト OS によるスケジューリングのオーバヘッドを削減するために Library OS[4] などの軽量 OS を用いて単一のアプリケーションを単一の VM として動作させる．VM 上のゲスト OS によるアプリケーションのスケジューリングを削減できるので，パフォーマンスが向上する．

既存の IaaS クラウド基盤ソフトウェアは，コンテナや VM に割り当てるメモリや CPU などの計算資源の情報から必要な計算資源を提供できる物理マシンを選択し，そのマシン上にコンテナや VM を動作させる．この際，コンテナや VM 上で動作させるアプリケーションの特性を考慮しないので，並列計算を行うアプリケーションのような高い並列度で動作させるべきコンテナや VM が同一の物理マシンで動作する場合，物理的な並列度が低下してしまうという問題がある．

本研究では，アプリケーションの特性に応じて動的に適切な仮想化環境を選択することのできる IaaS クラウド基盤を提案する．コンテナや VM 上のアプリケーションが物理マシンの物理的な並列度を超える並列度で動作しているような場合，別の物理マシンへコンテナや VM を移動させ並列度を向上できる．具体的には，アプリケーションのパフォーマンスを向上させるためにアプリケーションの特性を考慮しコンテナまたは軽量 OS を用いた VM でアプリケーションを動作させるように動的に再構成する．本稿では，試作したコンテナと VM の動作状況の取得機能について述べ，いくつかのアプリケーションを用いた実験によりコンテナや VM の構成を変更することによってパフォーマンスが向上することを示す．

2 IaaS クラウドで用いられる仮想化技術

IaaS クラウドを構築しサービスを提供するための仮想化環境が提案されている．本節では，ハイパーバイザ型仮想化，コンテナ型仮想化，軽量 OS を用いた VM について説明する．

2.1 ハイパーバイザ型仮想化

ハイパーバイザ型仮想化では，複数の VM がハイパーバイザによって仮想化され，その上で OS とアプリケーションを動作させる (図 1)．各 VM 上では異なる OS を動作させることができる．ハイパーバイザが VM をスケジューリングし，VM 上で OS がプロセスをスケジューリングする二重のスケジューリングが起こっている．

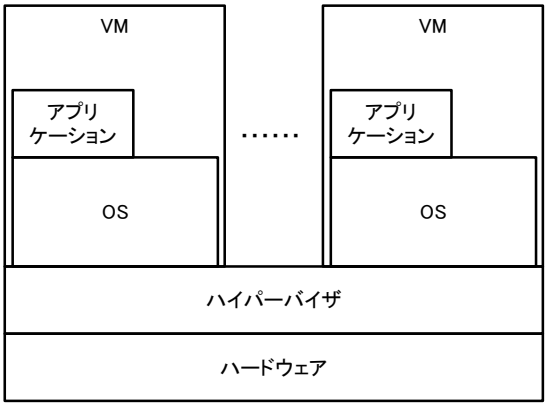

図 1 ハイパーバイザ型仮想化の構成

2.2 コンテナ型仮想化

通常の OS 上でアプリケーションを隔離した環境で動作させる（図 2）．ハイパーバイザが動作していないのでハイパーバイザ型仮想化における VM をスケジューリングするオーバヘッドが解消される．ハイパーバイザ型仮想化において VM でアプリケーションを動作させる場合よりも

高いパフォーマンスが得られる．Linux カーネルの機能を使ってコンテナの管理を行う Linux Containers[1] やコンテナの作成だけではなくイメージや複数ホストマシンの管理を行うことができる Docker[2] などが開発されている．

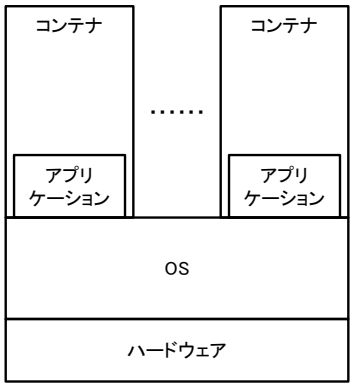

図 2　コンテナ型仮想化の構成

2.3　軽量 OS を用いた VM

コンテナ型仮想化とは異なりゲスト OS のスケジューリングを削減するために図 3 のように VM 上で軽量 OS を動作させ，その上で単一のアプリケーションを動作させるというクラウドの構成法が提案されている [3]．VM 上でのプロセスのスケジューリングオーバヘッドが削減できるが，軽量 OS 上のアプリケーションは使用する OS によって実装言語やライブラリが制限されるのでユーザが自由にアプリケーションを実装することはできない．

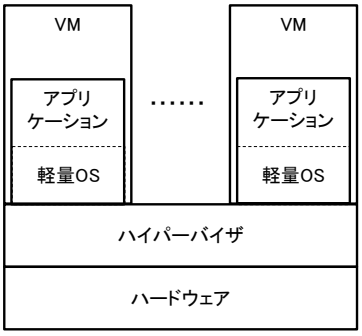

図 3　軽量 OS を用いた VM の構成

3　仮想化基盤の構成を変更する IaaS クラウド基盤

本研究では，アプリケーションの動作状況を考慮して VM 構成を動的に変更できる IaaS クラウド基盤を提案する．本節では，その概要について述べる．本 IaaS クラウド基盤の特徴は，アプリケーションの監視を行う部分と，コンテナや VM の構成を動的に変化させる部分である．

3.1　アプリケーションの監視

ハイパーバイザの機能やトレースツールを使って動作状況を取得する．軽量 OS を使った VM では動作するアプリケーションは単一で，さらにトレースツールを動作させることはできないので軽量 OS を使った VM の動作状況はハイパーバイザの機能を使って取得する．トレースツールは，通常の OS 上で動作させることができ，通常の OS 上やコンテナで動作するアプリケーションの動作状況を取得することができる．例えば，ハイパーバイザの機能を使うことで，各 VM の CPU の使用率やディスクやネットワークの I/O の状況が取得でき，トレースツールではスレッドの多重度やプロセススケジューリングの待ち時間を取得することができる．

3.2　VM 構成の変更

動作しているアプリケーションの特性に応じてパフォーマンスがより高くなる仮想化環境で動作させる．コンテナで動作させる方がパフォーマンスが高い場合にはコンテナで動作させ，軽量 OS を使った VM で動作させた方が高い場合には VM で動作させる．

また，動作させるアプリケーションによっては複数の VM やコンテナを使って構成されるので，パフォーマンスが向上する配置になるように移動する．図 4 はコンテナの配置を変更している例である．並列度の高いアプリケーションが 1 つのホストマシンが集中してパフォーマンスが低下している場合には複数の物理マシンを使ってアプリケーション構成し並列度を上げる．しかし，コンテナ間で通信を行っているようなアプリケーションは，同一ホストマシン上で動作させることでアプリケーションの並列度は低下するが，通信のオーバヘッドが小さくなりパフォーマンスが向上する場合がある．

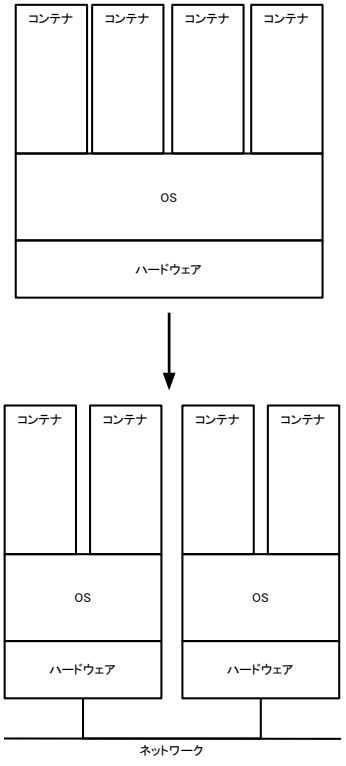

図 4　分散させて動作させる

4　プロトタイプシステムの実装

提案したシステムの有効性を確認するために，プロトタイプシステムを実装した．Xen ハイパーバイザ [5] の機能を使って各 VM の CPU 使用率と I/O の状況を取得するツールを作成した．ハイパーバイザからは VM 上のアプリケーションの動作状況を取得することができないが，単一のアプリケーションを動作させる軽量 OS を使った VM の動作状況を取得した場合には，CPU を多く使うアプリケーションや I/O を多く行うアプリケーションが動作している VM であることが推測できる．

また，proc ファイルシステムからアプリケーションの動作状況を取得するトレースツールを作成した．proc ファイルシステムからはスレッドの多重度，I/O の待ち時間，スケジューリングの待ち時間といったアプリケーションの動作状況を取得することができる．通常の OS 上やコンテナで動作しているアプリケーションの動作状況を取得することで，多重度の上がっているアプリケーションや待ち時間の長いアプリケーションを判断できる．

今回は，コンテナや VM の構成を動的に変化させる機構は実装していない．Unikernel には，1 つのアプリケーションコードから通常の OS 上やコンテナで動作する実行ファイルと軽量 OS を用いた VM のイメージファイルを作成するコンパイラが実装されているものがある．このコンパイラを利用して，動的に構成を変化させる機能を実装できると考えている．

5　実験

アプリケーションの特性毎にどちらの仮想化環境で動作させることでパフォーマンスが高くなるかを明らかにする必要がある．また，コンテナや VM をどのような配置で動作させることでパフォーマンスが高くなるかを明らかにする必要がある．そこで，アプリケーションのパフォーマンスがより向上する仮想化環境と配置を知るために実験を行なった．

5.1　実験環境

表 1 に示される PC4 台を実験に使用した．仮想化ソフトウェアに Xen と Docker を使用し，動作させる通常の OS に Ubuntu，軽量 OS に OSv[6] を使用した．Xen を用いる場合の VM はすべて表 2 に示す同じ仕様とした．Docker を用いたコンテナはすべて表 3 に示す同じ仕様とした．

5.2　Spark を用いた並列計算

CPU を多く使うアプリケーションの動作を確認するために Spark を用いた並列計算を行うアプリケーションを動作させた．

表 1　PC の仕様

CPU	Intel®Core™i7-4770 3.4GHz
コア数	4 コア 8 スレッド
メモリ	16GB
HDD 容量	1TB
OS	Ubuntu Server 14.04.5 64bit
ネットワーク	1000BASE–T

表 2　VM の仕様

CPU コア数	2 コア
メモリ	2GB
HDD 容量	32GB
OS	Ubuntu Server 16.04.1 64bit OSv 0.24
ハイパーバイザ	Xen 4.7.1

表 3　コンテナの仕様

CPU コア数	2 コア
メモリ	2GB
コンテナ	Docker 1.6.2

5.2.1　実験内容

Java のバージョン 1.8.0_111 と Apache Spark のバージョン 2.0.1 を用いてモンテカルロ法で円周率の近似値を求める並列計算を行うアプリケーションを動作させた．各計算ノードは互いに通信しながら計算を行う．通常の OS が動作する VM を用いる場合とコンテナを用いる場合と軽量 OS を用いた VM を用いる場合とで計算時間を比較した．Spark は計算処理をオンメモリで行うが，今回動作させるアプリケーションに必要なメモリは約 730MB で，どの仮想化環境で動作させた場合でもメモリは不足していない．

5.2.2　実験結果

1 ホストマシンあたりの VM またはコンテナ数を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．その実験結果を図 5 に示す．

1 ホストマシンあたりのサーバ数に関係なく軽量 OS を使った VM でアプリケーションを構成し処理を行う場合の計算時間が他の 2 つの場合より短くなっている．しかし，サーバ数が 4 台以上では，物理的な並列度が上がらないので処理時間が短くならない．

5.3　Web サーバを用いた実験

ネットワーク I/O を行うアプリケーションの動作を確認するために Tomcat を用いた実験を行った．

5.3.1　実験内容

Apache Tomcat のバージョン 8.0.32 を用いて Web サーバを動作させる．Web サーバのベンチマークソフトである

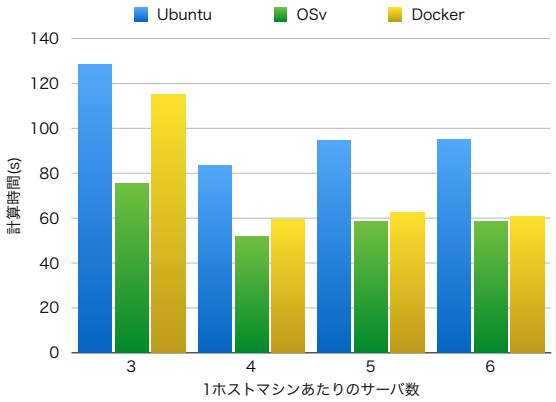

図 5　並列計算アプリケーションの実験結果

httperf[7] を使って負荷を掛け処理能力を計測する．Web サーバは，リクエストに対して静的で軽量な Web ページを返している．通常の OS が動作する VM を用いる場合とコンテナを用いる場合と軽量 OS を用いた VM を用いる場合とで結果を比較した．

5.3.2　実験結果

Web サーバでは，1 ホストマシンあたりの VM またはコンテナ数を変化させ 1 秒間あたりのレスポンス数を測定した．その実験結果を図 6 に示す．

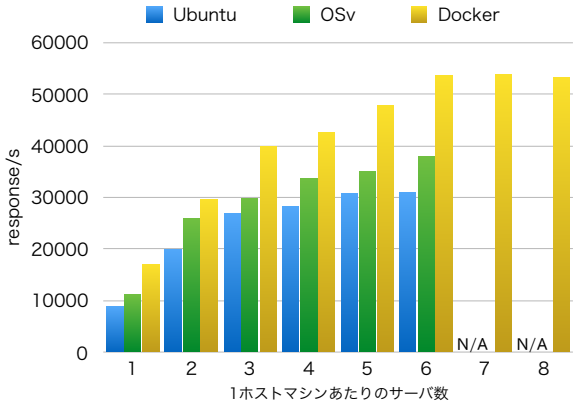

図 6　Web アプリケーションの実験結果

コンテナを用いて構成した場合のパフォーマンスが 3 つのうち最も高くなっている．しかし，サーバ数が 6 台以上では物理的な並列度が上がらないので，パフォーマンスが向上しなくなる．一方，VM を用いる場合，メモリの制限でサーバ数は 6 台までしか動作させられないが，サーバ数が増えるとパフォーマンスが向上しなくなる傾向が見られる．

5.4　考察

CPU を多く使うアプリケーションでは軽量 OS を用いた VM で構成することでパフォーマンスが向上し，ネットワーク I/O の多いアプリケーションは，コンテナを用いてアプリケーションを構成することでパフォーマンスが向上することがわかる．また，動的にコンテナまたは VM の構成を変更する際には，ホストマシンの物理的な並列度を超えるような並列度でアプリケーションを動作させないように配置しなければならない．

6　まとめ

動作しているアプリケーションの特性に応じて適切な仮想化環境を動的に選択することができる IaaS クラウド基盤を提案した．本 IaaS クラウド基盤によって，アプリケーションの動作基盤をコンテナまたは軽量 OS を使った VM に動的に変更することで，アプリケーションの動作パフォーマンスを向上することができる．実験により，CPU を多く使うアプリケーションを軽量 OS を用いた VM で構成した場合にアプリケーションの動作パフォーマンスを改善でき，ネットワーク I/O の多いアプリケーションをコンテナで構成する場合にアプリケーションの動作パフォーマンスを改善できることが分かった．

今後は，アプリケーションを動的に別の仮想化環境で動作させる機構について検討する．

参考文献

[1] “Linux Containers,” https://linuxcontainers.org/, 2017/2/6 accessed.

[2] “Docker - Build, Ship, and Run Any App, Anywhere,” https://www.docker.com/, 2017/2/6 accessed.

[3] Anil Madhavapeddy, et al., “Unikernels: Library Operating Systems for the Cloud,” in *Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS ’13. New York, NY, USA: ACM, 2013, pp. 461–472.

[4] D. R. Engler, et al., “Exokernel: An Operating System Architecture for Application-level Resource Management,” in *Proceedings of the Fifteenth ACM Symposium on Operating Systems Principles*, ser. SOSP ’95. New York, NY, USA: ACM, 1995, pp. 251–266.

[5] Paul Barham, et al., “Xen and the Art of Virtualization,” *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 164–177, oct 2003.

[6] Avi Kivity, et al., “OSv: Optimizing the Operating System for Virtual Machines,” in *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference*, ser. USENIX ATC’14. Berkeley, CA, USA: USENIX Association, 2014, pp. 61–72.

[7] “GitHub - httperf/httperf: The httperf HTTP load generator,” https://github.com/httperf/httperf, 2017/2/6 accessed.