Vol.2014-OS-130 No.21
2014/7/29

仮想化におけるCloudSuiteの性能解析

尾板 弘崇[1,a)]　山田 浩史[1]　谷本 輝夫[2]　小野 貴継[2]　佐々木 広[3]

概要：近年，クラウドサービスは，課金額に応じて必要な計算リソースを利用することができるため，設備投資をせずに大規模なデータ処理が可能となる．クラウドサービスを提供する企業 (クラウドサービスプロバイダ) はサーバに仮想マシン (VM) を使用していることが多い．一方，マルチコアプロセッサが広く普及してきており，ひとつのチップ上に搭載されるコア数は現在も増え続けている．マルチコアプロセッサを搭載したマシン上で VM を立ち上げる場合，VM に割り当てる仮想 CPU(vCPU) 数は適切に分配する必要がある．vCPU を多く与えてもアプリケーションの中には並列性が頭打ちになるものがあり，性能が上がるとは限らず，少なすぎてもアプリケーションの並列性を生かせず，性能が出ない．そのため，適切に分配しない場合，無駄な課金やマルチコアプロセッサの非効率な利用につながってしまう．本研究では，クラウド環境においてマルチコア CPU 上で動作する VM の挙動を解析し，マルチコア CPU を効率よく動作させるための VM の集約方法や仮想マシンモニタ (VMM) が提供すべき資源管理手法確率のための足がかりとする．解析は，VM に割り当てるコア数を変化させながら，クラウド環境で動作するワークロードを模したベンチマーク [1] を動作させ，ワークロードのリソース使用率を観察する．具体的には，機械学習 (Data Analytics)，キーバリュー型ストレージ (Data Serving)，グラフマイニング (Graph Analytics)，Web 検索 (Web Search) のワークロードを動作させる．実験の結果，クラウドサービスで実行されるワークロードには，vCPU 数を増やすともにスケールするもの，ある vCPU 数まではスケールするが，それ以降は変わらない，あるいは，逆に性能が下がってしまうもの，vCPU 数にかかわらずスケールしないものがあるということが明らかになった．

1. はじめに

クラウド環境では，課金額に応じて必要な計算リソースを利用することができるため，設備投資をせずに大規模なデータ処理が可能となる．クラウド環境を提供する企業 (クラウドサービスプロバイダ) は，ユーザに貸し出す資源として仮想マシン (VM) を採用していることが多い．これは，VM の数や構成で計算リソース使用量を簡単に調節できるからである．実際に VM を使用しているサーバに Amazon EC2 [2] などがある．また，クラウドサービスを提供するサーバ側では，サーバ統合を行っていることがある．これによって，1 台の物理マシンで複数の VM を立ち上げ，リソースを効率的に使用することができる．

一方でマルチコアプロセッサ技術の発達により 1 台のサーバ計算機が保持するコア数が増加しており，それに伴って VM が搭載する仮想 CPU 数 (vCPU 数) も多くなってきた．電力や半導体のサイズなどハードウェア面での問題によって処理速度を向上させるのは困難であるため，ひとつのチップ上にコアを複数搭載し並列処理を行うことによって処理の高速化を行うことが主流となっている．こうしたプロセッサを活かすために，10 以上の vCPU を搭載する VM(Manycore VM) も少なくなくなってきた．実際に Amazon EC2 では 32 個の vCPU を搭載するインスタンスが提供されており，1 台のサーバ計算機に搭載される物理コア数の増加は今後も続く傾向にあり，VM が搭載する vCPU 数も今後も続くと考えられる．

VM に与える vCPU 数は適切に決めなければならない．vCPU を多く与えてもアプリケーションの中には並列性が頭打ちになるものがあり性能が上がるとは限らないためである．また，少なすぎてもアプリケーションの並列性を生かせず性能が出ないためである．不適切な vCPU の割り当ては，無駄な課金やマルチコアプロセッサの非効率な利用につながってしまう．

本研究では，クラウド環境上で動作するワークロードが Manycore VM の恩恵を享受できるかを調査する．vCPU の数を変更していき，そのときの性能や挙動を解析する．これらの調査を通じて，マルチコア CPU を効率よく動作さ

[1] 東京農工大学
Tokyo University of Agricluture and Technology
[2] 富士通研
Fujitsu Laboratories
[3] コロンビア大学
Columbia University
[a)] oita@asg.cs.tuat.ac.jp

せるための VM の集約方法や仮想マシンモニタ (VMM) が提供すべき資源管理手法確立の足がかりとする．クラウド環境で動作するワークロードを模したベンチマークである CloudSuite[1] を動作させ，ワークロードのリソース使用率を観察する．今回はその第一歩として，機械学習 (DataAnalytics)，キーバリュー型ストレージ (Data Serving)，グラフマイニング (GraphAnalytics)，Web 検索 (Web Search) のワークロードを対象とした．OS は Linux 3.5.0，VMM は Xen 4.2.0[3]，サーバ計算機として 48 コア搭載するマシンを使用した．

実験の結果，vCPU 数を増やすともに性能がスケールするもの，ある vCPU 数まではスケールするが，それ以降は逆に性能が下がってしまうもの，vCPU 数にかかわらずスケールしないものがあるということがわかった．具体的には，Data Analytics は 30 コアまではコア数に応じてスケールし，3 コアのときに比べて 30 コアのときは 3 倍高速化した．Data Serving はコア数に応じたスケールは見られなかった．Graph Analytics は 12 コアまでコア数に応じてスケールし，1 コアのときに比べて 12 コアのときは 4 倍高速化したが，それ以降は性能が低下し，12 コアのときに比べて 48 コアのときは性能が半減してしまった．WebSearch は 18 コアまではスケールし，1 コアのときに比べて 30 コアのときは 10 倍高速化したが，それ以降は性能に大きな変化は見られなかった．

2. 関連研究

クラウド環境で動作するワークロードの挙動解析を行った研究がある．Ferdman らはクラウド環境で動作するワークロードを模したベンチマーク群を作成し，スケールアウトなワークロードにとって，現在のプロセッサの設計は，キャッシュミス率の高さ，命令，メモリレベルの並列性の低さ，キャッシュ，帯域幅の無駄遣い等の非効率な点があることを明らかにしている [1]．Ferdman らはハードウェア面に着目しており，ソフトウェア面には触れていない上，コア数による挙動の変化に着目していない．さらに，仮想化環境を考慮していない．

マルチコアプロセッサ上において，ソフトウェアのスケーラビリティに着目した研究がある．Boyd-Wicizer らは Linux のスケーラビリティを阻害する Linux の問題点を特定し，その解決法を提案している [4]．しかし，クラウドサービスで多く利用されている仮想化環境を考慮していない．また，Salomie らはマルチコアプロセッサ上でデータベース管理システム (DBMS) を実行，解析を行い，マルチコアプロセッサ上で DBMS を動作させる上での問題点を明らかにし，その解決法を提案している [5]．現在，DBMS だけでなく，さまざまなクラウドサービスが提供されている．本研究では DBMS だけに限らずさまざまなアプリケーションに対して研究を行う．

サーバ統合を行った場合のクラウド環境で動作するワークロードの挙動を解析した研究がある．Apparao らは，サーバ統合による性能の低下がコアの割り当てやキャッシュによることを明らかにしている [6]．また，Lv らは，性能の低下が不必要なコンテキストスイッチが多く発生し，メモリやキャッシュ量の増加によることを明らかにし，それを改善するためのスケジューラを提案している [7]．彼らの研究方針は性能の低下の原因を明らかにし，原因を改善することである．しかし，本研究の方針は，性能低下の原因の改善ではなく，性能低下を受けながらも全体的な性能を最大にする VM の集約方法や VMM の資源管理手法を提案することである．サーバ統合のリソースの競合における性能低下の対処法として，Novakovic らは，競合の発生を感知，競合の詳細を分析し，競合を解消するため，競合を起こしている VM を他の物理マシンにマイグレーションするシステムを提案している [8]．しかし，この手法は競合の対処法としてマイグレーションするのみであり，割り当てられているコア数を変化させるなどマルチコアプロセッサの利用に関して考慮されていない．

クラウド環境で動作するワークロードを対象にして仮想化環境におけるリソース管理手法を提案した研究がある．Vasic らは，負荷の変化を認識し，負荷の変化に合わせて割り当てるリソースを決定するリソース管理を高速化するシステムを提案している [9]．しかし，性能評価基準にサーバ統合のことを考慮されていない．

3. 提案

本研究では，クラウド環境における複数仮想 CPU を搭載する VM の挙動解析を行う．実際に仮想化環境を構築し，仮想マシンが備える vCPU 数を変化させながらクラウドサービスで動作するワークロードの性能とリソース使用量を計測する．コア数ごとの性能の変化からはワークロードのスケールの仕方を，リソース使用量からはワークロードの特徴を明らかにすることができるためである．もしワークロードがスケールしなかった場合，リソース使用量を分析すれば，ワークロードのスケールを阻害した要因を推測することも可能になる．例えば，リソース使用量より，CPU がボトルネックと明らかになれば，CPU の性能を良くすることにより，より高い性能を得られるとわかる．

このように VM の挙動を解析し，分析することにより，マルチコア CPU を効率よく動作させるための VM の集約方法や VMM が提供すべき資源管理手法を確立するための足がかりとすることができる．

4. 実験方法

4.1 ベンチマーク

本研究ではベンチマークとして CloudSuite[1] を用いる．CloudSuite とは，クラウド環境で動作するワークロー

ドを模したベンチマーク群であり，8 つのベンチマーク (Data Analytics, Data Caching, Data Serving, Graph Analytics, Media Streaming, Software Testing, Web Search, Web Serving) から構成される．本研究では，この 8 つのうち，計測しやすい 4 つのベンチマーク (Data Analytics, Data Serving, Graph Analytics, Web Search) を用いて実験を行う．この 4 つのベンチマークについて詳しく説明する．

4.1.1 Data Analytics

今日，ビックデータの重要性が広く認知されてきており，ビックデータを解析するクラウドシステムの需要が高まってきている．

Data Analytics は、機械学習を行うワークロードを模したベンチマークである．本研究では，Mahout[10] と Hadoop[11] を用いて，30GB の wikipedia の記事をもとに機械学習を行う．Mahout は機械学習を行うライブラリであり，Hadoop は大容量のデータの解析を高速に行うための分散処理システムである．

4.1.2 Data Serving

従来、DBMS には関係データベース管理システム (RDBMS) を用いるのが主流であった．しかし，クラウドサービスが広く使われるようになり，Facebook inbox や Google Earth のように大規模なデータを扱うようになった現在，サーバの台数を増やすことで簡単にスケールすることができる NoSQL が注目を集めている．

Data Serving は，Facebook inbox や Google Earth といったサービス内での DBMS へのアクセスを模したベンチマークである．クライアントは DBMS にリクエストを送り，サーバがリクエストにあったデータをクライアントに送信する．本研究では，DBMS に Cassandra[12] を，DBMS へのリクエストは書き込みと読み込みの比率が 95:5 のジップ分布に基づいてリクエストを送る Yahoo! Cloud Serving Benchmark(YCSB)[13] クライアントを用いる．また，データセットは 15GB の YCSB データセットを用いる．Cassandra は並列度が高く，与えたコアを効率よく使用するように設計されている．また，読み込みよりも書き込みの方が高速に実行できる．これは，メジャーな RDBMS や他の NoSQL と異なり，書き込みにライトスルー方式ではなく，ライトバック方式を採用しているからである．書き込みはディスクにアクセスする必要がないため，高速に行える半面，読み込みは大量のデータを読み込んでマージする必要があるため，低速になってしまう．

4.1.3 Graph Analytics

テキストデータを元に行うデータマイニングと異なり，グラフマイニングは分散処理グラフを用いて大規模なグラフでデータ解析を行う．グラフマイニングは Twitter や Facabook などのソーシャルネットワークワークの台頭により，より膨大な量のデータを分析する必要が生じている．しかし，膨大なデータから特徴的部分構造を見つける問題の多くが NP 完全問題であるなど，本質的計算困難性を有していることが知られている?．そこでグラフを用いて分析することにより，データの大まかな特徴を得ることができる．このような背景より，グラフマイニングの重要性は高まってきている．

Graph Analytics は、グラフマイニングを行うワークロードを模したベンチマークである．本研究では，データマイニングと機械学習を行う GraphLab[14] を用いて，5GB の Twitter のデータセットに PageRank アルゴリズムを実行し，Twitter のフォロワー数に基づく Twitter ユーザの影響についてグラフマイニングを行う．PageRank アルゴリズムは Web ページの重要度を決定するためのアルゴリズムである．また，このベンチマークはデータセットを読み込んでから PageRank アルゴリズムで計算しているが，性能は PageRank アルゴリズムでの計算部分のみで評価する．

4.1.4 Web Search

Google 検索や Microsoft Bing のような Web 検索エンジンは，クラウドサービスの典型的なものである．

Web Search は，Web 検索エンジンを模したベンチマークである．クライアントは検索ワードをサーバに送信し，サーバはワードを元に検索を行い，結果をクライアントに送信する．本研究では，分散処理に対応したオープンソース検索エンジンである Nutch/Lucene[15] を用いる．Nutch はオンラインソースから大量のデータを集め，インデックス化する．また，クライアントからリクエストを受けとると，各ノードにリクエストを送信し，レスポンスを回収，ソートし，クライアントにリプライを返す．Nutch では，ディスク I/O によるスループットや QoS の低下を避けるために，各ノードのメモリに合うようにサイジングされる．

4.2 実験手順

まず，それぞれの VM を立ち上げ，VM に割り当てる vCPU 数を変化させながらワークロードを実行し，性能の変化を観察する．同時にワークロードが使用しているリソース使用率も観察する．VM 内の CPU 使用率とメモリ使用量を top で，Domain-0 と VM の CPU 使用率を xentop で，ディスク I/O 量は iostat で、ネットワーク I/O 量は dstat で計測した．性能のぶれを考慮し，ワークロードを 5 回実行し，平均を求めてワークロードの性能とする．

なお，コア数を変更し，ワークロードを実行するときは，バッファキャッシュが性能に影響を与えないように，一度 VM をシャットダウンし，再度立ち上げ，コマンドによりバッファキャッシュをクリアしてから実行する．

4.3 実験環境

4.3.1 実験マシン

実験マシンは，AMD Opteron Processor 6300, 256GB

表 1　VM の構成

VM	vCPU	Mem	OS
Domain-0	1	32GB	Linux 3.5.0
Data Serving	1〜48	32GB	Linux 3.5.0
Data Analytics	1〜48	64GB	Linux 3.5.0
Graph Analytics	1〜48	64GB	Linux 3.5.0
Web Search	1〜48	32GB	Linux 3.5.0
Client	48	32GB	Linux 3.5.0

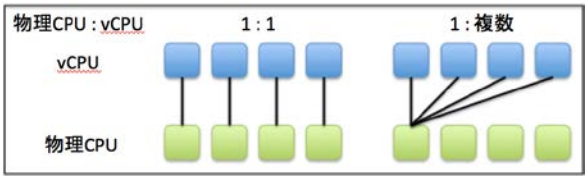


図 1　物理 CPU と vCPU の対応関係

メモリを搭載したマシンを 2 台搭載した PowerEdge C6145 を使用した．AMD Opteron Processor は 48 コア搭載している．それぞれのコアはキャッシュを保持している．HDD は 150GB と 2TB のそれぞれ 2 つを使用した．VM を構成するディスクイメージは 2TB の HDD に保存されている．

4.3.2　仮想化環境の設定

VMM には Xen 4.2.0[3] を用いた．VM の設定を表 1 に示す．vCPU は，Domain-0 には 1 コア，クライアントの VM に 48 コア割り当てる．Cassandra，Mahout, GraphLab, Nutch は 1 コアから 48 コアまで動的に vCPU 数を変化させる．また，メモリは，Domain-0 には，Mahout, GraphLab を実行するときは 12GB，Cassandra，Nutch を実行するときは 32GB を割り当て，Mahout, GraphLab の VM に 64GB，Cassandra, Nutch, クライアントの VM にそれぞれ 32GB 割り当てる．OS は，Domain-0 は Ubuntu 12.10, VM は Debian 7.0 であり，Linux カーネルはどれも Linux 3.5.0 を使用する．

物理 CPU と vCPU の対応関係は図 1 のように 2 種類あるが，本研究では 1 対 1 の対応を採用する．また，VM の vCPU 数を動的に変化させても常にひとつの物理 CPU を Domain-0 と VM が共有するように Domain-0 に物理 CPU を割り当てる．サーバ統合時には，I/O 用 VM も Domain-0 と同様に物理 CPU を割り当てる．

5.　実験と考察

5.1　Data Analytics

まず，図 2 にコア数ごとにワークロードを実行したときの実行時間をグラフ化したものを示す．値が低い方が性能が高いことを示している．3 コアのときは 11000 秒，12 コアのときは 6000 秒，24 コアのときは 5000 秒とコア数が増えるにしたがって性能が向上した．しかし，30 コア以上の場合は 4000 秒前後と，コア数が増えても性能は変わらなかった．

次に、リソース使用率を示していく。なお、リソース使用率に関して 48 コア分示すと量が多くなってしまうのでそれぞれのベンチマークで 4 コア分のみ示す．また，リソースには CPU，メモリ，ディスク I/O，ネットワーク I/O 等さまざまあるが，これらも全て示すと量が多くなってしまうため，コア毎の CPU 使用率とボトルネックなど注目すべきリソースのみを示すものとする．CPU 使用率は usr+sys 時間，idle 時間，I/O wait 時間の 3 つについて示している．また，CPU に関する割合は 100%のとき与えられた CPU を完全に使用していることを意味する．DataAnalytics では，12，24，36，48 コアのときの CPU 使用率とディスク I/O 発行量を図 3 に示す．CPU 使用率を見ると，12，24 コアのとき，mahout はほぼ 100%CPU を使用しているのに対し，36，48 コアのときは CPU を 100%使用できておらず，idle 時間とわずかに I/O wait 時間が生じている部分があるのがわかる．そこで，ディスク I/O 発行量を見ると 12，24 コアのときに比べて 36，48 コアのときはディスク I/O 発行量が多く発生しているのがわかる．これより，30 コアからスケールしなくなったのは I/O wait と 30 コア以降 CPU がうまく割り当てられていないことが原因であると考えられる．

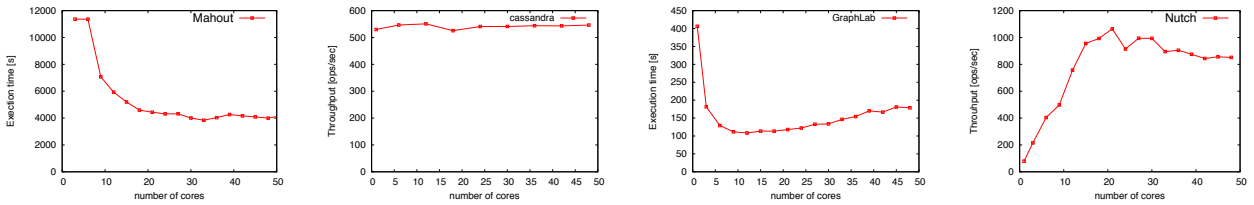
5.2　Data Serving

まず，図 2 にコア数ごとにワークロードを実行し，得られたスループットをグラフ化したものを示す．スループットは 1 秒間にいくつの命令を実行できたかを表しており，値が高い方が性能が高いことを示している．図からわかるように，どのコアの時も 520ops/sec~560ops/sec を推移しており，スケーラビリティは見られなかった．

次に、リソース使用率を示していく。DataServing では，1，12，24，48 コアのときの CPU 使用率とディスク I/O 発行量を図 4 に示す．CPU 使用率を見ると，idle 時間が高く，コア数が増えても CPU をほとんど使えていないのがわかる．また，実行を開始してから暫くの間，usr+sys 時間よりも I/O wait 時間の方が高くなっている．そこでディスク I/O 発行量を見ると読み込みを大量に行っていることがわかる．これより，Cassandra にスケーラビリティが見られないのは I/O wait と CPU がうまく割り当てられていないことが原因であるが原因であると考えられる．

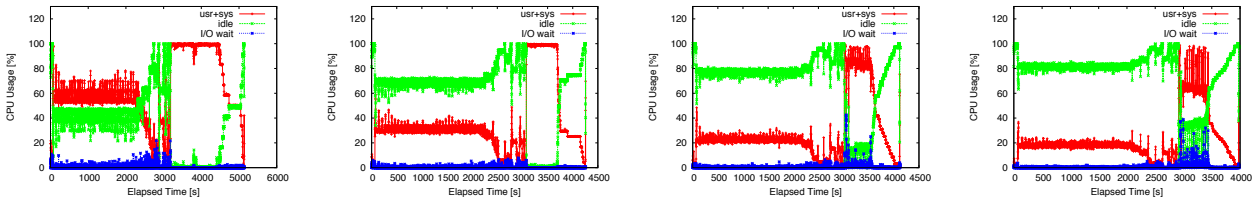
5.3　Graph Analytics

まず，図 2 にコア数ごとにワークロードを実行したときの実行時間をグラフ化したものを示す．値が低い方が性能が高いことを示している．1 コアのときは 400 秒，6 コアのときは 125 秒，12 コアのときは 100 秒とコア数が増えるにしたがって性能が向上した．しかし，24 コアのときは 125 秒，48 コアのときは 175 秒と，12 コア以上の場合，コア数が増えるにしたがって性能が低下していった．



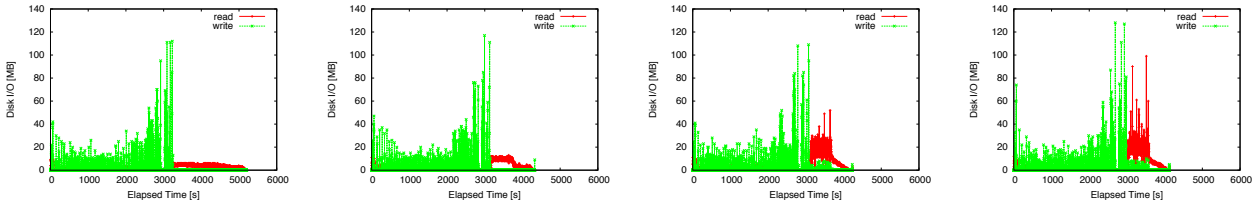
(a) DataAnalytics (b) DataServing (c) GraphAnalytics (d) WebSearch

図 2 各ワークロードの実行結果



(a) 12 コア (b) 24 コア (c) 36 コア (d) 48 コア

CPU 使用率



(e) 12 コア (f) 24 コア (g) 36 コア (h) 48 コア

ディスク I/O 発行量

図 3 Data Analytics の資源使用率

次に, リソース使用率を示していく．GraphAnalytics では，図 5 に 1，12，24，48 コアのときの CPU 使用率を示す．図 5 より，GraphLab はコア数にかかわらず，ほぼ 100%CPU を使用しているのがわかる．他にボトルネックになるようなリソースが見られないことから，12 コア以降性能が低下したのは，コア数が増えるにしたがって不必要な処理に CPU を使用しているからではないかと考えられる．

5.4 Web Search

まず，図 2 にコア数ごとにワークロードを実行し，1 秒間にどれだけ検索を行ったかを示すスループットをグラフ化したものを示す．値が高い方が性能が高いことを示している．1 コアのときは 100ops/sec，12 コアのときは 800ops/sec，21 コアのときは 1100ops/sec と 21 コアまでは性能が向上したが，それ以降は 24，36 コアのときは 900ops/sec，48 コアのときは 800ops/sec と性能が低下してしまっていた．

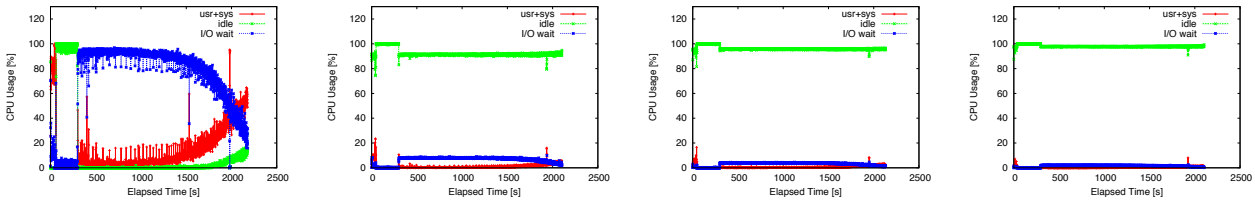
次に，リソース使用率を示す．WebSearch では図 6 に 1，12，24，48 コアのときの CPU 使用率とネットワーク使用量を示す．CPU 使用率を見ると usr+sys 時間の割合がコア数が増えるにしたがって低下しているのがわかる．よって，CPU がうまく割り当てられていないことがわかる．また，微妙に発生している I/O wait はネットワーク I/O である．

5.5 まとめ

実験より，クラウド環境で実行されるワークロードには，ある一定のコア数まではスケールするもの (Data Analytics, Graph Analytics, Web Search)，コア数に関係なくスケールしないもの (Data Serving) があることが確認できた．ある一定のコア数まではスケールするものにはその VM にスケールしていく最大のコア数を，コア数に関係なくスケールしないものには 1 コアだけを与え，余ったコアを他の VM に与えることで効率的に資源を使用することが可能となる．

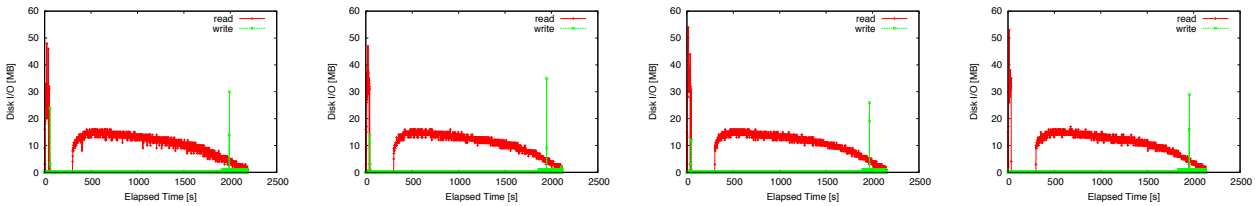
6. おわりに

本研究では，クラウド環境においてマルチコア CPU 上で動作する VM の挙動を解析した．実験の結果，クラウドサービスで提供されるようなワークロードには，一定数のコアまでスケールするもの，全くスケールしないものに分けられることが明らかになった．これらの結果から，スケーラビリティによって VM に割り当てる vCPU 数を動的に変更する，あるいは，スケーラビリティに応じて VM を配置することによって計算リソースを効率的に利用する



(a) 1 コア　(b) 12 コア　(c) 24 コア　(d) 48 コア

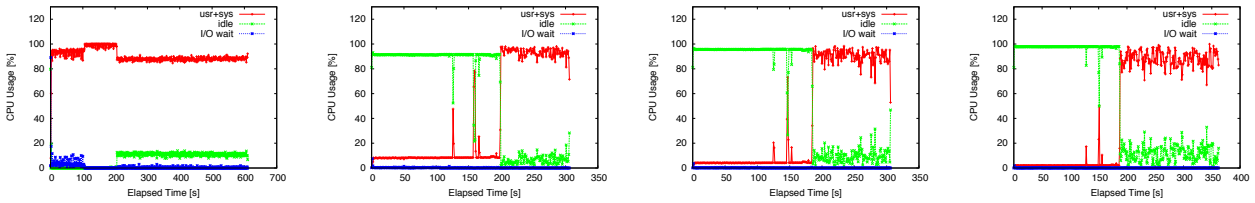
CPU 使用率



(e) 1 コア　(f) 12 コア　(g) 24 コア　(h) 48 コア

ディスク I/O 発行量

図 4　Data Serving の資源使用率



(a) 1 コア　(b) 12 コア　(c) 24 コア　(d) 48 コア

CPU 使用率

図 5　Graph Analytics の資源使用率

ことができると考えられる．

参考文献

[1] Ferdman, M., Adileh, A., Kocberber, O., Volos, S., Alisafaee, M., Jevdjic, D., Kaynak, C., Popescu, A. D., Ailamaki, A. and Falsafi, B.: Clearing the Clouds: A Study of Emerging Scale-out Workloads on Modern Hardware, *Proceedings of the seventeenth international conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '12)*, pp. 37–48 (2012).

[2] Amazon: Amazon Elastic Compute Cloud (EC2) - Scalable Cloud Hosting. `http://aws.amazon.com/`.

[3] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I. and Warfield, A.: Xen and Art of Virtualization, *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP '03)*, pp. 164–177 (2003).

[4] Boyd-Wickizer, S., Clements, A. T., Mao, Y., Aleksey Pesterev, M. F. K., Morris, R. and Zeldovich, N.: An Analytics of Linux Scalability to Many Cores, *Proceedings of the 9th USENIX conference on Operating systems design and implementation (OSDI '09)*, pp. 1–8 (2010).

[5] Salomie, T.-I., Subasu, I. E., Giceva, J. and Alonso, G.: Database engines on multicores, why parallelize when you can distribute?, *Proceedings of the sixth conference on Computer systems (EuroSys '11)*, pp. 17–30 (2011).

[6] Apparao, P., Iyer, R., Zhang, X., Newell, D. and Adelmeyer, T.: Characterization & analysis of a server consolidation benchmark, *Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments (VEE '08)*, pp. 21–30 (2008).

[7] Lv, H., Dong, Y., Duan, J. and Tian, K.: Virtualization Challenges: A View from Server Consolidation Perspective, *Proceedings of the eighth ACM International Conference on Virtual Execution Environments (VEE '12)*, pp. 15–25 (2012).

[8] Novakovic, D., Vasic, N., Novakovic, S., Kostic, D. and Bianchini, R.: DeepDive: Transparently Identifying and Managing Performance Interference in Virtualized Environments, *Proceeding of the 2013 USENIX conference on Annual Technical Conference (USENIX ATC '13)*, pp. 219–230 (2013).

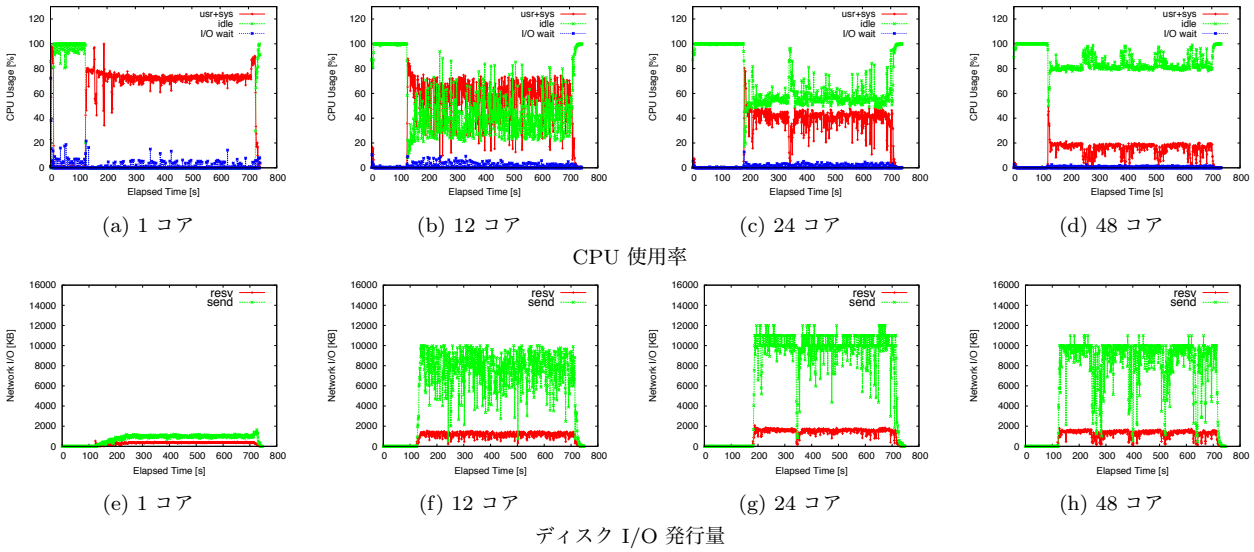
[9] Vasic, N., Novakovic, D., Miucin, S., Kostic, D. and Bianchini, R.: DejaVu: Accelerating Resource Allocation in Virtualized Environments, *Proceedings of the 17th international conference on Architectural Support for Programming Languages and Operating System (ASPLOS '12)*, pp. 423–436 (2012).

[10] Mahout: Apache Mahout: Scalable machine learning and data mining. `https://mahout.apache.org/`.

[11] Hadoop: Welcome to Apache Hadoop. `http://hadoop.apache.org/`.

[12] Cassadra: The Apache Cassandra Project. `http://cassandra.apache.org/`.

[13] Cooper, B. F., Silberstein, A., Tam, E., Ramakirishnan, R. and Sears, R.: Benchmarking cloud serving systems



(a) 1 コア (b) 12 コア (c) 24 コア (d) 48 コア

CPU 使用率

(e) 1 コア (f) 12 コア (g) 24 コア (h) 48 コア

ディスク I/O 発行量

図 **6** Web Search の資源使用率

with YCSB, *Proceedings of the 1st ACM symposium on Cloud computing (Socc '10)*, pp. 143–154 (2010).

[14] GraphLab: GraphLab - Large-Scale Machine Learning on Graphs. `http://graphlab.org/`.

[15] Nutch: Welcome to Apache Nutch. `http://nutch.apache.org/`.