データ処理アプリケーションのクラウドリソースと ローカルクラスタ間における負荷分散ミドルウェアの検討

豊島　詩織†　　山口　実靖††　　小口　正人†

† お茶の水女子大学　〒 112-8610 東京都文京区大塚 2-1-1
†† 工学院大学　〒 163–8677 東京都新宿区西新宿 1–24–2
E-mail: †shiori@ogl.is.ocha.ac.jp, ††sane@cc.kogakuin.ac.jp, †††oguchi@computer.org

あらまし　高度 IT 社会の進展に伴いデータの管理や IT コストの問題が深刻になっている．その問題に対して各々の組織でサーバを構築するのではなく，クラウドコンピューティングを利用することでシステムの構築コストを抑えることができる．コストパフォーマンスのよいデータ処理システムが望まれている中，手元のクラスタ使用状況を観察し，リソースが不足している場合は外部のクラウドリソースを動的に使用する，クラスタのスケーラブルな運用のためのミドルウェア構築を目指す．クラウドコンピューティングは不足している CPU パワーを補うことには有効であるが，本研究で議論を行うデータインテンシブアプリケーションは通常の計算処理とは異なる特徴がある．ミドルウェアにおいてはディスク I/O の状態により，投入されたジョブをローカルクラスタまたは商用のクラウドサービスである Amazon EC2 に振り分ける．データベースベンチマークを使用しミドルウェアの性能評価を行ったところ，クラウドサービスのコスト面において理想的な配置に近い，ジョブの振り分けの判断ができることを確認した．

Development of middleware for data processing load distribution using cloud computing resource

Shiori TOYOSHIMA†, Saneyasu YAMAGUCHI††, and Masato OGUCHI†

† Ochanomizu Univ　　2-1-1 Otsuka, Bunkyouku Tokyo 112-8610 JAPAN
†† Kogakuin University　　1–24–2 Nishi-shinjuku, Shinjuku-ku, Tokyo, 163–8677 Japan
E-mail: †shiori@ogl.is.ocha.ac.jp, ††sane@cc.kogakuin.ac.jp, †††oguchi@computer.org

Abstract　In recent years, data management and IT cost have become serious problem as the advanced IT society progresses. Cloud computing is introduced in order to save the construction cost of local cluster at each organization. This mechanism is suitable for supplying insufficient CPU power to a computing system. However, data-intensive applications have a totally different feature from CPU-intensive applications. Thus, execution of data-intensive applications making use of cloud computing resources is discussed in this paper.Since a data processing system with better cost/performance ratio is required, we have constructed middleware for load distribution among cloud computing resources and a local cluster. Scalable resource management is achieved by monitoring resource usage of a local cluster and insufficient resources are acquired dynamically from a cloud service. Our middleware allocates injected jobs optimally among the local cluster and commercial cloud service Amazon EC2, based on the result of the I/O disk load conditions. According to the evaluation using a data-intensive benchmark, our middleware has achieved an excellent job allocation, which is close to an ideal case in terms of the cost of the cloud service.

1. はじめに

高度 IT 社会の進展によりコンピュータシステムにおいて利用可能なデータの量が増大している近年，よりスケーラブルなリソース管理の実現が望まれている．そこで期待が高まっているのがクラウドコンピューティングである．クラウドコンピューティングにおいてユーザはリソースを利用するだけであるため，システムの運用コストが大幅に削減できる．そして必要なときに必要な機能を，必要な分だけ利用することが可能となる．本研究においてデータインテンシブアプリケーションのためのデータ処理システムを構築するにあたり，全てのシステムをクラウドで構築することも考えられるが元々自前のデータ処理シ

ステムを所有していることが考えられる他，運用面やコストの面でのリスクが懸念される．そのため上記で述べたクラウドコンピューティングのメリットを活かし，使用しているクラスタのシステム状況をモニタリングし，外部のクラウドリソースへ負荷分散するミドルウェアの構築を目指す．クラウドリソースを利用しているため，特に急激に大量のキャパシティが必要となる場合に有効になると考えられる．

ローカル環境における負荷が高い場合にネットワーク越しの遠隔リソースへ負荷を分散すること自体は，グリッドコンピューティングなどの枠組みで実現できるため，新しい考え方ではない．しかし遠隔リソースとしてクラウドを利用した場合，以下のような点で従来とは異なる特徴がある．まずユーザのニーズに応じてリソースを大幅に増減できることが期待される．またセキュリティポリシにより社外にデータを置けないユーザでも，データは社内に保存したまま，計算能力だけクラウドから借りることが可能になる．

さらに本研究ではデータインテンシブアプリケーションを対象負荷としている．クラウドコンピューティングにおけるロードバランスを議論している論文として [1] や [2] がある．しかしこれらの論文では CPU リソースの負荷分散を対象としており，データインテンシブアプリケーションは対象としていない．科学技術計算など計算処理が中心となるアプリケーションの場合は，各ノードの CPU 負荷により判断して適切な負荷分散を行うことができるが，データインテンシブアプリケーションの場合，CPU は I/O 待ちとなっていることが多く，CPU 負荷では適切な判断が行えない．そこで本研究では負荷の指標として，ディスクアクセス量を用いた．

このようにリソースとして伸縮性の高いクラウドを用いている点，そしてデータインテンシブアプリケーションを対象負荷として用いている点が，本研究の特徴である．

2. ローカルクラスタのシステム構成

2.1 仮想マシン PC クラスタ

IT リソースを効率的に活用するために仮想化が用いられる．自前のクラスタを効率よく運用するためクラスタシステムとしてワーカノードに仮想マシンを配置した PC クラスタを使用した．仮想化ソフトには Xen [4] を使用した．Xen は図 1 に示すように複数の OS を動かす為の基盤となるプラットフォームのみを提供することで仮想マシンのオーバヘッドを少なくし，物理マシンに近い性能が発揮されるよう工夫されている．オープンソースとしては非常に高性能で，現在ビジネスユースにおいても用いられるようになっている．仮想マシンモニタが仮想化のための土台となり，その上で Domain と呼ばれる仮想マシンが動作する．ホスト OS として動いているものが Domain0，ゲスト OS として動いているものが DomainU で，Domain0 は実ハードウェアへのアクセスやその他のドメインを管理する特権を持つ．クラウドコンピューティングでは仮想化が重要な構成要素となっており，仮想マシンを利用することでフレキシブルなリソースの利用が可能となっている．

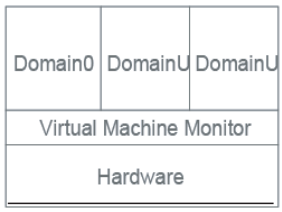

図 1　Xen の構造

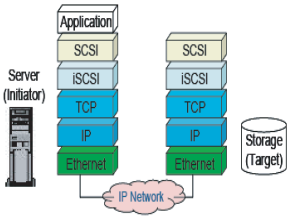

図 2　iscsi の階層構造

2.2 IP-SAN によるストレージ統合

ストレージネットワークには SAN を使用した．近年情報システムにおいて処理されるデータの量が膨大になってきたことから，ネットワークストレージ技術が発展し，PC クラスタのストレージに SAN を用いることが多くなっている．SAN は，分散したストレージをネットワークで統合し，ストレージの集中管理とディスク資源の効率的な活用を可能にする．特に IP-SAN は Ethernet インタフェースと TCP/IP 対応ネットワークさえあれば導入でき，また専用網も含め広範囲に IP ネットワークのインフラが整備されているため長距離接続が可能で，クラウドコンピューティングなどの枠組を用い，計算機リソースのアウトソーシングに利用されることが期待される．

IP-SAN のプロトコルとしては iSCSI (Internet Small Computer System Interface) [5] を使用した．iSCSI の構造を図 2 に示す.iSCSI は SCSI コマンドを TCP/IP パケットの中にカプセル化することでブロックレベルのデータ転送を行う．Gigabit Ethernet/10Gigabit Ethernet が広く普及していくであろうことを考慮すると，IP-SAN をバックエンドに持つ PC クラスタ多くが使用されるようになると考えられる．

2.3 データインテンシブアプリケーションのローカルクラスタ上の実行

データインテンシブアプリケーションでは CPU 負荷でジョブの実行状態を把握することができない．そこで I/O 負荷で把握することを試みた．図 3 は pgbench のそれぞれのクライアント数毎おけるディスク負荷である．この図の Disk-Read よりクライアント数 1〜7 において負荷が右肩上がりになっていることが確認された．この結果を元に，モニタリングで得られた I/O 負荷の値から今現在処理が行われているジョブ量を推測する．

User	Disk Read (kByte/sec)	Disk Write (kByte/sec)
1	3273	29000
2	3248	33666
3	3932	32000
4	4568	35000
5	5048	38000
6	5391	38333
7	5440	33000
8	5327	28666
9	5484	42500
10	7093	42000

図 3　ディスクアクセス負荷

3. ローカルクラスタからクラウドリソースを調達

3.1 概　　要

本研究ではデータインテンシブアプリケーション実行時に，ローカルクラスタの I/O 負荷をモニタリングすることでジョブ量を測定し，外部のクラウドリソースへ動的に負荷分散を実現するミドルウェアの構築を目指す．

その際システム構成として考慮すべきことは，まずローカルクラスタとクラウドには何ノードのマシンがあるのかということである．またデータインテンシブアプリケーションでは頻繁にデータベースアクセスが発生するため，サーバに対するデータの位置が実行時間に大きな影響がある．そのためデータ（ストレージ）とジョブ（サーバ）の位置，またローカルクラスタとクラウドにいくつずつジョブを投入するかが重要な考慮すべき点である．本実験では図 4 のように，ローカルクラスタにサーバが 4 台，クラウドリソースは無限にマシンが使用できるという環境である．このときマスタからサーバへジョブの投入が行われるが，CPU 負荷で測定したローカルクラスタの負荷が大きい場合はリソースの不足分を EC2 で補う．

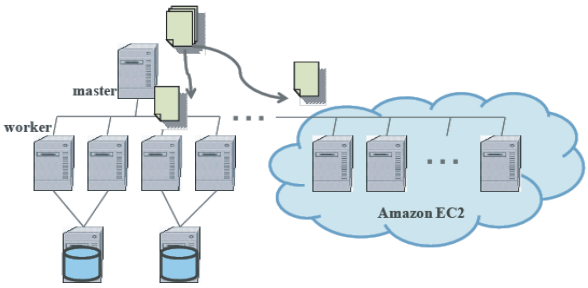

図 4　ローカルクラスタからクラウドリソースの調達

3.2 データの配置

本研究では自身のクラスタの負荷を一定間隔のモニタリングにより観察し，負荷が大きい場合は外部のクラウドコンピューティングリソースへ負荷分散を行うミドルウェアを構築しその評価を行う．負荷分散のためのクラウドコンピューティングとしては商用のクラウドサービスである Amazon EC2 [9] を用いる．ただし本研究では主にデータインテンシブアプリケーションの実行を想定しているため，リソースとしては，計算処理を行うサーバだけでなく，データおよびストレージをどのように取り扱うか考えなければならない．

リモートサイトのサーバに計算処理の一部をマイグレートする場合，処理を行うデータの配置については，大きく分けて以下の 3 つのケースが考えられる．

(a)　処理に必要なデータについても処理のマイグレート時にオンデマンドでリモートサイトにコピーする場合

(b)　ローカルクラスタでの処理中に遠隔バックアップが行われており，あらかじめリモートにデータが存在する場合

(c)　セキュリティポリシや，データが巨大すぎるなどの理由でリモートにデータを置くことができない場合

(a) については一般にリモートへのデータ転送はスループットが低いため，この方式はデータ量が多い場合には，性能低下を招く可能性がある．ただしコピーが終わったらリモートサイトにおいてデータへ高速アクセスが可能となるため，データ量が少ない場合やアプリケーション全体の処理時間が長い場合には有効であると考えられる．

(b) の場合にはデータアクセスに関する制約が無くなるため，積極的にリモートサイトのリソースを利用した方が性能面では有利になると考えられる．

(c) の場合には，計算処理のみリモートサイトのリソースを利用しながら，データはローカルに置きリモートサイトからアプリケーション実行時にアクセスする事が考えられる．例えば Google 社が提供するクラウドサービスである Google Apps においては Secure Data Connector という仕組みが提供されており，これを利用するとクラウドとローカルサイトの間にセキュアトンネルが構築され，クラウドからローカルサイトのデータに対し，安全にアクセスを行う事ができるようになる．このケースの場合には，リモートサイトから計算処理サーバだけ借りれば良く，容易に負荷分散のマイグレーションが実現できる．ただしリモートサイトとローカルクラスタの間の通信性能が全体の実行性能に大きな影響を与えるため，ネットワークの帯域幅が小さい場合やデータアクセス頻度が高いアプリケーションの場合には，性能の大幅な低下が予想される．また，リモートサイトからローカルのストレージへのアクセスには制限がある場合もあり，これらの問題をクリアしなければならない．このようにリモートサイトのリソースを利用して負荷分散を行う事を考える際，データインテンシブアプリケーションの場合には，データをどのように扱いどこに置いて実行するか考える事が重要である．さらに上記の 1 つ目と 2 つ目のケースのように，データをリモートサイトに配置して計算処理を実行する場合には，リモートサイトにおけるストレージについても，何台用いデータをどのように配置すべきかについて検討する必要がある．本研究では (b) 利用したいデータが既にリモートサイトに存在する環境での実験を行う．

4. データ処理アプリケーション最適配置ミドルウェア

4.1 最適配置の概要

本研究ではローカルクラスタの負荷を指標とし、負荷が大きい場合は外部のクラウドコンピューティングリソースへ自動的に負荷分散を行うミドルウェアを構築する．予備の実験により，pgbench 実行時のボトルネックはディスクアクセスであり，CPU やネットワークにはまだ余裕があるということを確認している．クラウドコンピューティングの一つの大きな特徴として，必要なときに必要な分だけリソースが使用可能であることがあげられる．利用するクラウドリソースを増やし，クラウドリソースへ多くの負荷を分散すればアプリケーションの実行時間が短くなることが期待されるが．しかしクラウドは従量制のコストがかかるということも特徴であるため，実行時間のみを考慮した場合コストが膨大になる可能性がある．そのため現実

にクラウドリソースを使用した負荷分散システムを使用する場合は，実行時間といったパフォーマンスとともに，コストパフォーマンスも考慮してリソースを配分することが必要である．そのため本研究では実行時間に制限が設けられた場合にローカルクラスタとクラウドコンピューティングリソース間でどのようにジョブを分けるのがいいのか，最もコストパフォーマンスが良くなるよう検討する．以下に最適配置の概要をまとめる．

【最適配置の概要】まずシステム環境ではジョブが順次投入されていく環境を想定しており，またマシン台数はローカルクラスタでは使用できる台数に制限があるという現実的な環境で，その不足分をクラウドリソースで補う．クラウド側は使用ノードの数の制限がない．

- 実行の制限時間をユーザが決定 (Limit time)
- 順次投入されるジョブを，ローカルクラスタに投入
- ローカルクラスタで Limit time を超える場合は EC2 に投入
- クラウド内において Limit time 内で最もコストが低くなるよう最適配置を判断　(クラウドコスト＝実行時間×台数)

まずアプリケーション全体の実行時間の制限時間をユーザが指定し，それを Limit time とする．アプリケーションの実行がスタートして次々投入されるジョブを，まずはローカルクラスタで実行していく．そしてローカルクラスタでジョブを実行する中で，ユーザが指定した Limit time を超える場合はクラウドリソースにジョブを投げ始め，クラウド内では Limit time 内で最もコストパフォーマンスが良くなるよう，クラウドから借りるマシンの台数やジョブの配置を決定する．

4.2　ミドルウェアの動作概要

データインテンシブアプリは CPU インテンシブアプリケーションとは大きく異なり，処理が I/O 待ちとなっていることが多い．従って CPU 負荷は割合が低く，CPU 負荷を見てもジョブ量の判断は行えないため適切な最適配置の判断を行うことは難しい．そのため上記の最適配置を実現するミドルウェアにおいてローカルクラスタの I/O 負荷をモニタリングすることにより，ジョブをクラスタと EC2 に最適配置を行うよう実装を行う．図 5 にミドルウェアの図を示す．

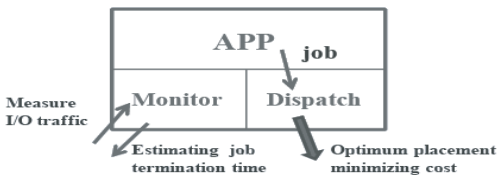

図 5　ミドルウェア図

- 定期的にローカルクラスタの I/O 負荷を測定
- I/O 負荷によりローカルクラスタのジョブ量をチェック
- ローカルの実行終了時間を見積もり，制限時間に基づきローカルとリモートジョブ振分けを決定
- リモートの分をコスト最小になるようノード数を決めて配置

dstat コマンドによりデータインテンシブアプリケーションアプリケーションでボトルネックの原因となる I/O 負荷を定期的に測定する．ミドルウェアはローカルクラスタをモニタリングし，図 3 の値を元に，現在何クライアント分相当のジョブが実行されているのかを判断し，制限時間超える場合は EC2 に振分ける．

このとき図 3 ではクライアント数が 7 以降の場合，処理が飽和状態となり，ジョブが増えてもモニタリングには現れていない．そのためクライアント数が 7 相当以上の場合にはローカルクラスタでの処理が飽和状態であると判断し，EC2 にジョブを投げることとする．

EC2 でのジョブの振分けについては多くのマシンを借りたほうが実行時間が短くなるが，その分コストが課金される．そのため最もコストパフォーマンスが良くなるようなマシンの台数を考える．このときコストは実行時間×台数で計算している．

上記のようにユーザから制限時間が与えられた場合，ローカルディスクの負荷状態から何クライアント分がローカルで実行されているかを判断し，それに基づいて EC2 に負荷分散を行うミドルウェアを使用した場合と，ローカルクラスタと EC2 における pgbench の実行時間より理想的なリソース配置を決めた場合との振舞いを比較し，ミドルウェアの精度を測定する．

5.　最適配置ミドルウェアの評価

5.1　評価実験の概要

クラスタの各ノードのスペックを表 2 に，実験環境を図 6 に示す. クラスタの各計算ノードには DomainU(virtual machine) を一つずつ配置した．Amazon EC2 のマシンスペックは High-CPU Medium インスタンスを使用した．

使用するリソースは、ローカルクラスタは限定（4 サーバ)、リモートのクラウド (High-CPU Medium) は台数無制限とする．ストレージについては，ローカル環境では iSCSI ストレージを使用し，EC2 ではローカルストレージを使用する．またデータについては上記の (b)「遠隔バックアップなどによりあらかじめリモートにデータが存在する」場合を考える．

表 1　Experimental setup：PCs

OS	Linux 2.6.18-128.el5(CentOS5.3)
CPU	Initiator：Intel (R) Xeon(TM) 3.6GHz
	Target：Intel (R) Xeon(TM) 2.66GHz
Main Memory	Initiator(DomainU)：1GB
	Target：8GB
iSCSI	Initiator：iscsi-initiator-utils
	Target：iSCSI-Enterprise-Target
Monitoring Tool	dstat

本研究ではデータインテンシブアプリケーションの動作を想定している．そのため評価アプリケーションとしては PostgreSQL のベンチマークである pgbench を用いた．データベースの大きさは 7.5GByte，サーバで実行するユーザ数 1 に対するトランザクション数を 1000 とした．

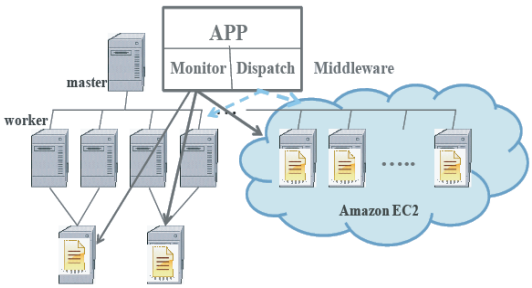

図 6　実験環境図

5.2　ミドルウェア評価方法

本実験においては，ジョブが一定間隔で追加投入されていくケースを想定し，ミドルウェアが現在のシステム状態を基に投入されたジョブの最適配置の判断を行う．5 秒ごとに 4 ユーザずつ合計 10 回投入し (図 7)，1 回毎にローカルクラスタと EC2 のどちらに投げるかを決定する．またシステムに負荷をかけ，より分かりやすくミドルウェアの評価を行うため，1 回の実行につき 10 回 pgbench を繰り返している．ジョブ投入間隔は全体の実行時間よりも十分に短い．またトータルの実行時間について，ユーザが指定した制限時間内に終了させることを制約条件とする．ローカルクラスタはコストが掛からず，クラウドは制限時間×台数でコストが決まるものとして，コストを最小にすることを目指す．

ローカルと EC2 におけるジョブの振り分けにおいて，ディスクアクセス量により自動で判断するミドルウェアを使用した場合と，ローカルのクラスタと EC2 における pgbench の実行時間とコストから求めた，最も理想的に振り分けた場合との振舞を比較し評価を行う．

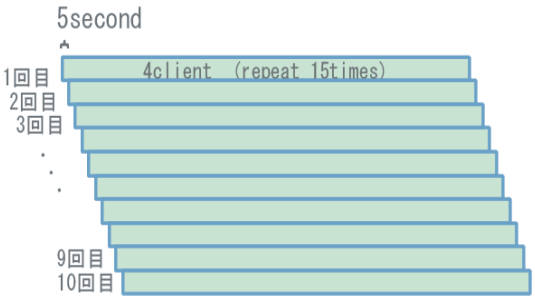

図 7　ジョブ投入イメージ

5.3　最も理想的な振分けをした場合

はじめにローカルクラスタと EC2 において pgbench を実行したときの実行時間と，EC2 のコスト概算から，最も理想的な振り分けと EC2 内でのリソース配置を求める．

まずローカルクラスタに 4client ずつ 5 秒毎に投げる繰り返しの数を変化させ，それぞれの実行時間を測定した (図 8)．この結果からユーザが指定する制限時間とローカルで実行する回数を決定する．

また図 9 は EC2 において 1 台で実行するクライアント数を変化させたときの実行時間である．ここから EC2 にジョブを振り分けたときの実行時間が概算できる．なおこの実行時間よりユーザが指定する制限時間は 264 秒から指定できるものとする．

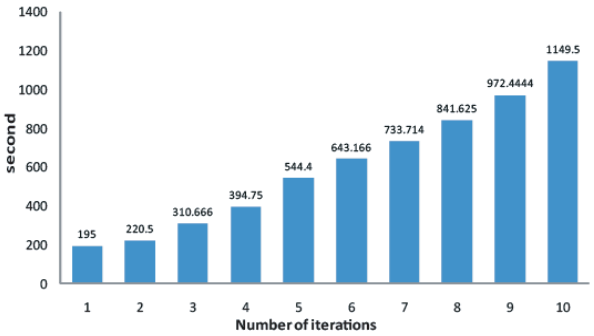

図 8　Indication of time limit

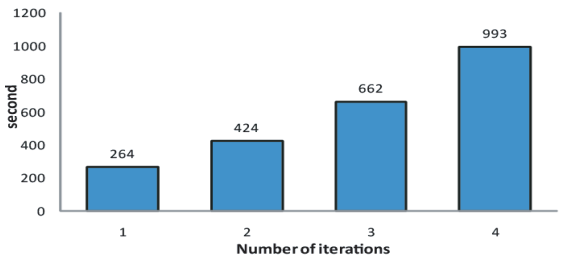

図 9　execution time on Amazon EC2

【例】

例えば制限時間が 600 秒と指定された場合は，ミドルウェア評価において 4 クライアントずつ 5 秒間隔で 10 回ジョブを投入するうち，図 8 より 5 回目まではローカルに投げることができるが，6 回目もローカルに投げると実行時間が 600 秒を超える．そのためローカルでは 5 回，残りの 5 回は EC2 で実行するのがよい．

EC2 内では 5 回× 4 クライアント＝ 20 クライアント分の投げ分けを考える必要がある．マシンの台数については 20 クライアントを全て別々のマシンに投入する場合の 20 台借りる場合から，全て同一のマシンで実行する場合の 1 台借りる場合が考えられる．ただ実行時間を考慮した場合，9 台以下になると 1 台につき 3 クライアント実行しなければならないマシンが発生し，実行時間が 600 秒以上になってしまう (図 8)．また実行時間×台数によりコストも考慮にいれると，10 台借りて 1 台につき 2 クライアントずつ実行するのが最もコストパフォーマンスがよい．

以上から制限時間が 600 秒と指定された場合には，10 回ジョブを投入するうち 5 回目まではローカルクラスタに投げ，残りの 5 回は 1 回ごとに 2 クライアントずつ 1 台に投入するのがよい．

5.4 実験結果と考察

図 10 にローカルクラスタと EC2 の投げ分けにおいて，ユーザが指定したそれぞれの制限時間における最も理想的な振り分けをした場合とミドルウェアを使用して振り分けた場合，10 回中何回ローカルクラスタと EC2 へ投げたかを示す．

EC2 にて pgbench を 1 クライアント実行したときの実行時間が 264 秒であるため (図 9) 制限時間は 264 秒以上から指定する．そのため理想的な振分けにおいて 10 回 pgbench のジョブを投入するうち 2 回ローカルで実行する場合から考える．また 7 回以上ローカルで実行するとモニタリングにおいてディスクアクセスが飽和となるため 8 回以上ローカルで実行することはない．

この結果から，ローカルで 5 回以上実行する場合，つまり負荷が多くかかる場合はモニタリング結果により正しくジョブの振り分けができていることが分かる．

Limit time(seconds)	264-310	310-394	394-544	544-643	643-733	733-
sorting of ideal	2 : 8	3 : 7	4 : 6	5 : 5	6 : 4	7 : 3
sorting using middleware	1 : 9	1 : 9	3 : 7	5 : 5	6 : 4	7 : 3

図 10　ローカルクラスタと EC2 への振分け比率

図 11 には EC2 内でのコストを表している．

ミドルウェアを使用した場合において，ローカルでの実行回数が 2〜4 の少ない場合では，正しく投げ分けができておらず EC2 で実行する回数が多くなっているため，理想よりもコストがかかっていることが分かる．またローカルでの実行回数が 5〜7 と多い場合は，正しい投げ分けができており，コストも理想と近くなることが確認された．pgbench はデータベースの状態等により，同じノードとジョブ配置で実行しても，実行時間には比較的大きなばらつきが生じる．図 11 でミドルウェア使用時に理想の場合よりコストが低くなっている部分があるのはそのためである．

ideal sorting	2 : 8	3 : 7	4 : 6	5 : 5	6 : 4	7 : 3
sorting of ideal	6144	5180	5136	2990	2224	1680
sorting using middleware	7668	7776	7140	2680	2144	1620

Cost= execution time * number of machines

図 11　EC2 におけるコスト

6. まとめと今後の課題

よりスケーラブルなデータ処理システムを実現するため，データインテンシブアプリケーション実行時，手元のクラスタ負荷をモニタリングし，負荷が大きい場合は外部のクラウドコンピューティングリソースへ負荷分散を実現するミドルウェアを構築した．実行のボトルネックになるディスクアクセスを負荷の判断指標としローカルクラスタと EC2 のジョブの振分けを行ったところ，ディスクアクセス負荷が大きい場合はディスクアクセスの値から正しく振り分けが行えていることが確認された．実行時間では理想に近い値が確認された．またローカルと EC2 への振分けについても理想に近い振分けができていることが分かった．現在はジョブの振分け指標としてローカルクラスタのディスクアクセス量を用いているが，今後は負荷が少ないところでも正しく投げ分け判断が行えるようモニタリングの精度を高めていく．

謝　　辞

本研究は一部，文部科学省科学研究費特定領域研究課題番号 18049013 によるものである．

文　　献

[1] G. Jung, K. R. Joshi, M. A. Hiltunen, R. D. Schlichting, and C. Pu: ”Generating Adaptation policies for Multi-Tier Applications in Consolidated Server Environments,” In Proc. 5th IEEE International Conference on Autonomic Computing (ICAC2008), pp.23-32, June 2008.

[2] E. Kalyvianaki, T. Charalambous, and S. Hand: ”Self-Adaptive and Self-Configured CPU Resource Provisioning for Virtualized Servers Using Kalman Filters,” In Proc. 6th International Conference on Autonomic Computing and Communications (ICAC2009), June 2009.

[3] pgbench: http://www.postgresql.jp/

[4] Xen : http://www.xen.org/

[5] iSCSI RFC: http://www.ietf.org/rfc/rfc3722.txt

[6] Asuka Hara, Kikuko Kamisaka, Saneyasu Yamaguchi, and Masato Oguchi: ”Analyzing Characteristics of PC Cluster Consolidated with IP-SAN using Data-Intensive Applications,” In Proc. 20th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS2008), No.631-042, November 2008.

[7] Masato Oguchi and Masaru Kitsuregawa:”Using Available Remote Memory Dynamically for Parallels Data Mining Application on ATM-Connected PC Cluster,”14th IEEE International Parallel and Distributed Processing Symposium (IPDPS2000), pp.411-420, May 2000.

[8] mpiBLAST:http://zach.asmallorange.com/suspended.page/

[9] Amazon Web Service: http://aws.amazon.com/

[10] Google Apps:http://www.google.co.jp/apps/intl/ja/business/inde

[11] Shiori Toyoshima, Saneyasu Yamaguchi, and Masato Oguchi: ”Storage Access Optimization with Virtual Machine Migration and Basic Performance Analysis of Amazon EC2,” In Proc. the Fourth International Workshop on Telecommunication Networking, Applications and Systems (TeNAS2010) in conjunction with the 24th IEEE International Conference on Advanced Information Networking and Applications (AINA2010), pp.905-910, Perth, Australia, April 2010.

[12] Aravind Menon,Alan L.Cox,Willy Zwaenepoel: ”Optimizing Network Virtualization in Xen,” 2006 USENIX Annual Technical Conference, pp.15-28, May 2006.

[13] Jose Renato Santos,Yoshio Turner,G.(John)Janakiraman,Ian Pratt:”Bridging the Gap between Software and Hardware Techniques for I/O Virtualization,” 2008 USENIX Annual Technical Conference, pp.29-42, June 2008.

[14] Tanimura Yusuke, Ogawa Hirotaka, Nakada Hidemoto, Tanaka Yoshio, Sekiguchi Satoshi: ”Comparison of Methods for Providing An IP Storage to A Virtual Cluster System,” IPSJ SIG Notes 2007, pp.109-114, March 2007.