DBSJ Journal, Vol.10, No.1
June 2011

データインテンシブアプリケーション実行時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

Middleware for Load Distribution among Cloud Computing Resource and Local Cluster used in the Execution of Data-Intensive Application

豊島 詩織♦ 山口 実靖♠ 小口 正人♥

Shiori TOYOSHIMA Saneyasu YAMAGUCHI
Masato Oguchi

コンピュータシステムにおいて利用可能な情報量が爆発的に増大し，それに伴うITコストも大きな問題となっている．こうした状況から，リソース使用状況に合わせてスケーラブルなシステム構築が可能なクラウドコンピューティングへの期待が高まっている．しかしそのメリットを考えても既にクラスタシステムを所持していたり，そこに大半のデータを置いている企業は全ての処理をいきなりクラウドに移行することは難しいと考えられる．そのため本研究においてはデータインテンシブアプリケーションを対象とし，ローカルシステムの使用状況から判断してリソースが不足している場合はスケーラブルに増減させた外部のクラウドリソースへ動的に負荷分散を行うミドルウェアを構築する．

1. はじめに

クラウドコンピューティングにおいて，ユーザはコンピュータリソースを従量制のコストにより利用でき，様々な種類の環境を必要な時に必要な分だけ利用することが可能となる．クラウドが持つメリットにより，システムの利用者が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場合には，これを有効に活用したい．また全ての処理をクラウドに任せるのは，運用面やコスト面でのリスクが懸念される．そのため本研究では，上述のクラウドコンピューティングのメリットを活かし，使用している自前のシステム状況をモニタリングして，負荷が高い場合のみ外部のクラウドリソースへ動的に負荷分散を行うミドルウェアを構築した．

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，例えば[1]や[2]などがある．しかしこれらの論文ではCPUインテンシブなアプリケーションを対象負荷としており，データインテンシブアプリケーションは考慮していない．ある種の科学技術計算のように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負荷に基づいて判断し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，判断基準として使うことは難しい．

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されるものであり，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．まずクラウドの高伸縮性により，ユーザのニーズに応じてリソースを大幅に増減することが可能となる．その一方でクラウドでは従量制のコストがかかることから，単に性能向上のみを目標として負荷分散を行うことはできず，コストパフォーマンスが良くなることを目指す必要がある．さらにはセキュリティポリシ等により社外のクラウドにデータを置けないユーザが，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ようなケースも想定される．本研究では，クラウドならではのこれらの点を考慮した上で，負荷分散ミドルウェアの構築とそれを用いた評価を行った．

このように，リソースとして伸縮性は極めて高いが従量制のコストがかかる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取り上げ，ジョブの最適配置ミドルウェアを構築している点が本研究の特徴である．

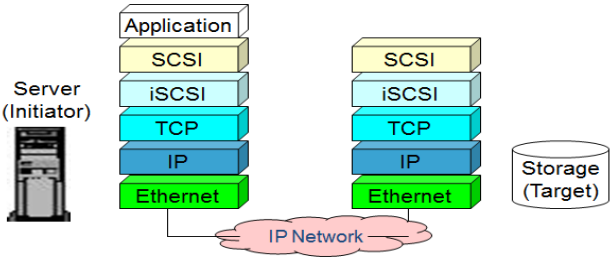
2. 仮想マシン PC クラスタ

本研究においてローカルサイトのシステムはクラスタのワーカノードに仮想マシンを配置したPCクラスタとし，仮想化ソフトにはXen[3]を使用した．

2.1 IP-SAN

ストレージアクセスができるだけ高速になるよう，ストレージネットワークには SAN を使用した． 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 iSCSI (Internet Small Computer System Interface)[4]を使用した．iSCSI の構造を図 1 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．

図 1 iSCSI の階層構造



♦♥ お茶の水女子大学大学院人間文化研究科
shiori@ogl.is.ocha.ac.jp
oguchi@computer.org
♠ 工学院大学
sane@cc.kogakuin.ac.jp

3. クラウドコンピューティング

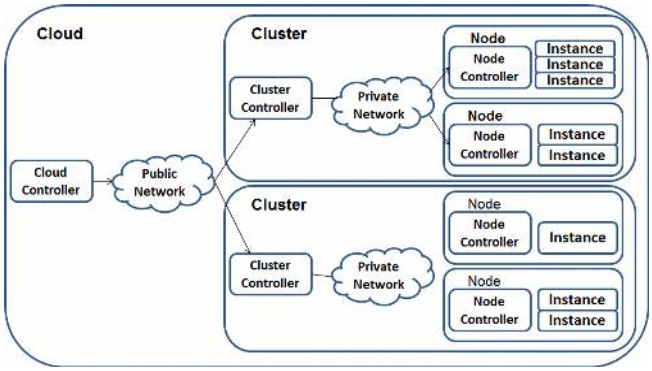
3.1 クラウドコンピューティングの概要

現在注目を集めているクラウドコンピューティングは,インターネットを介してアクセスするだけで,ネットワーク上に存在するサーバが提供するハードウェアやソフトウェアを利用できるサービス形態であり，計算機リソースにおいては HaaS (hardware as a service)モデルが知られている．ハードウェアをクラウドから利用する場合,ユーザは実機を買い揃える必要がないため運用・管理コストの削減が可能になるほか，予めシステム規模が予測しづらいときなど,現在の使用状況に合わせてキャパシティを増減できるといったメリットがある.本研究ではこの特徴を利用し,クラスタで急激に大量のリソースが必要となった場合にクラウドリソースへ負荷分散を行なう.

3.2 Eucalyptus

本研究ではクラウドシステムに，東京大学生産技術研究所に構築されたEucalyptus[5]クラウドを使用した.Eucalyptusはオープンソースのクラウド基盤構築ソフトウェアである.米Amazon.com社が提供しているクラウドサービスであるAmazon EC2 (Amazon Elastic Compute Cloud)[6]のAPIと互換性を持っており，Eucalyptusを使用することで，Amazon EC2上のサービスをそのままEucalyptusで構築したクラウドに移すことが可能である．既存研究で我々はAmazon EC2をクラウドリソースとして使用した実験を行ったが[7]，本研究ではEucalyptusを用い，既存研究を発展させた．またEucalyptusは，Amazonが提供するストレージサービスであるAmazon Simple Storage Service (Amazon S3)やAmazon Elastic Block Store (Amazon EBS)とも同等の機能を備えており，それぞれWalrus，Block Storageと呼ばれている.Eucalyptusは以下の3つのコンポーネントで構成されている(図2).CLCからCCまでの上位層をPublic Network，NCまでの下位層をPrivate Networkとして扱う.

図 2　Eucalyptus の構成



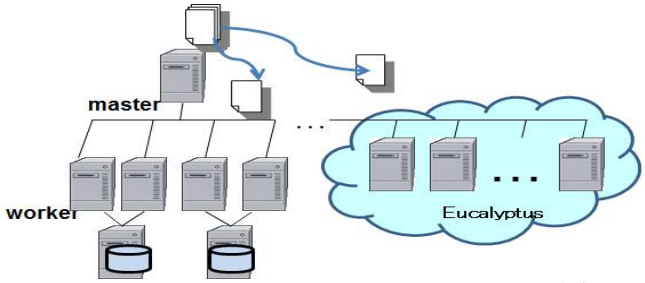
- Cloud Controller (CLC)
クラウド全体の情報を管理．Amazon EC2互換インタフェースを備え，利用者に対してAPIやWeb管理画面を提供.
- Cluster Controller (CC)
インスタンスの仮想ネットワークの管理やNode Controllerとインスタンスの状態管理を行う.
- Node Controller (NC)
実際にインスタンスを動作させたり，制御を行う．複数のインスタンスを稼働させるため，Node Controller上では仮想化ソフトウェアが稼働する.

4. データインテンシブアプリケーション最適配置ミドルウェア

4.1 システム環境と最適配置ミドルウェアの概要

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,ローカルクラスタとクラウドリソースのDisk I/Oをモニタリングすることで実行されているジョブ量を推定し,投入されたジョブを最適配置するミドルウェアを構築した．すなわち図3に示すように,データ処理アプリケーションのジョブが連続的に投入されている状況において,ローカルクラスタとリモートのクラウドリソースのどこにジョブを配置するのが最もコストパフォーマンスが良くなるかをミドルウェアが決定し,そのノードにジョブを送り込む．システム環境は，ローカルクラスタでは使用できるマシン台数に制限があるという現実的な状況を想定し,必要に応じて不足分をクラウドリソースで補う形とした．クラウド側は使用ノード数の制限がないものとした.

図 3　システム構成



ミドルウェアはローカルクラスタのマスタノード上で動作しているシェルスクリプトとC言語のプログラムで，ローカルクラスタおよびクラウドリソースのDisk I/Oを測定するMonitor部とジョブを振り分けるDispatch部から成る.Monitor部では,dstatコマンドを利用して定期的にDisk I/Oの情報を収集している．一方Dispatch部では，投入されたジョブを受け取り，収集したDisk I/Oの情報を基に，ローカルクラスタまたはクラウドリソースへジョブを配置する.この時ジョブの最適配置を決めるアルゴリズムについて，次節で紹介する.

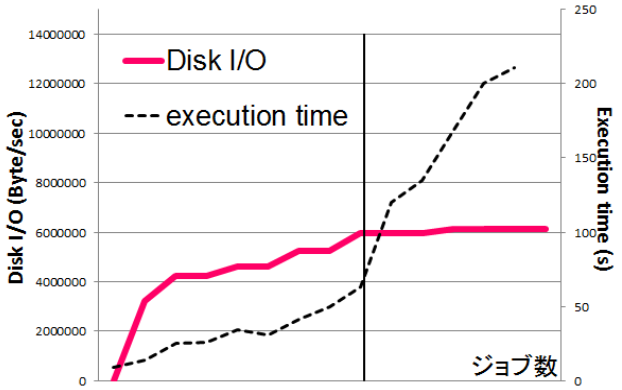
4.2 ミドルウェアの動作アルゴリズム

ローカル処理をクラウドに負荷分散することを考えた場合,クラウド側で多くのインスタンスを用い,多くの処理を並列して実行すれば，全体の実行時間が短くなることが期待される.しかしクラウドは従量制のコストが発生するため,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．従ってクラウドリソースを使用した負荷分散システムを実現する場合は，実行時間などのパフォーマンスとともに，従量制コストも考慮してリソースを配分することが重要である.これを実現するためにはまずはローカルシステムを有効な範囲で使い切ることが必要となる.本研究で対象としているデータインテンシブアプリケーションでは,CPUはI/O待ちとなっていることが多いため,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る.

図4は各ノードで実行するジョブ量を変化させた場合のアプリケーション実行時間とDisk I/Oの値である．図のように,実行時間はジョブ量が増えるに伴い長くなるが，Disk I/Oはある一定の値を超えると飽和となり,ジョブが増えても値として増加しない状態に陥る．DiskI/Oが飽和に達すると，アプリケーションの実行時間が極端に長くなる.そこで本ミドルウェアにおいてはこのように飽和した状態を「リソースを使い切った」状態とみなし，そのノードへのジョブの投入を終了する．Disk

I/0が飽和したとミドルウェアが判断する方法は，ジョブの実行が行われるノードにおいて定期的にDisk I/0を測定し，ピークの値がある一定の回数以上変わらなければI/0が飽和したものとしている．

図 4　実行時間と Disk I/O の関係



以下に最適配置のアルゴリズムをまとめる．ジョブはユーザから連続的に投入されていることを想定している．

1. 連続投入されたジョブを受け取る．
2. ローカルクラスタが飽和していなければローカルクラスタで実行して(1)へ．飽和していれば(3)へ．
3. クラウドにおいて実行優先順位が高いインスタンスから順次飽和しているか調べ，飽和していないものが見つかり次第実行して(1)へ．見つからなければ(4)へ．
4. クラウドにおいて借りるインスタンスを増やし，そこで実行し，(1)へ．

まずはローカルクラスタ，次はクラウドの優先順位が高いものから Disk I/O の状態を調べ，空いているところから実行を行うことで，アプリケーションの実行時間とクラウドの従量制コストの両方がバランス良く最小となるようにしている．

5. クラウドにおけるストレージアクセスとその性能

5.1 クラウド利用時のデータ配置に関する議論

本研究のように，ローカルサイトにおけるデータインテンシブアプリケーションの実行の一部を，クラウドなどリモートサイトのサーバで実行させる場合，処理を行うデータの配置については，以下の 3 つのケースが考えられる．

A) 処理の遠隔実行開始時に，処理に必要なデータについてもオンデマンドでリモートサイトにコピーする
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いるなどにより，あらかじめリモートサイトにデータが存在する
C) セキュリティポリシや，データが巨大すぎる，コストがかかるなどの理由でリモートサイトにデータを置くことができない

(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ればリモートサイトから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[8] においては Secure Data Connector[9] 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，データコピーなどを考慮せずに負荷の遠隔実行が実現できる．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にデータアクセス頻度が高いアプリケーションを実行すると，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，これらの問題を考慮する必要がある．

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(A)と(B)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など,性能やコストについて検討する必要がある．

本研究においては，上記の(B)と(B)のケースに基づいた評価を行う．すなわちデータが既にリモートのクラウド上に存在する場合と，データは常にローカルシステムにのみ置かれている場合を想定し，リモートへのデータ転送やそのコストについては考慮しない．

5.2 実験環境

図 5　実験環境

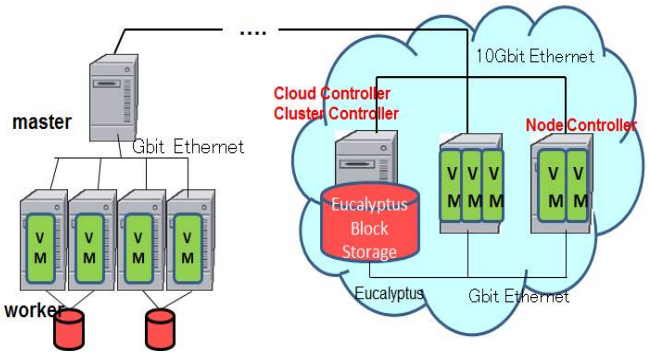


図 5 に本研究の実験環境を示す．ローカルシステムはワーカノードに仮想マシンを配置した仮想マシン PC クラスタとし，お茶の水女子大学に設置した．アプリケーションの実行はそれぞれの仮想マシン上で行われる．ローカルサイトの各ノードのスペックを表 1 に示す．アプリケーションが動作するのは実マシン上に構築した仮想マシンで，メモリは 1GB としている．

一方リモートのクラウドリソースは，東京大学生産技術研究所に設置されたクラスタ上に，Eucalyptus を用いてクラウドシステムを構築した．このシステムの物理マシンのスペックを表 2 に，アプリケーションが動作する仮想マシンのスペックを表 3 に示す．CLC と CC は同一のマシン上で動作し，

Eucalyptus Block Storage は CLC のローカルストレージである．Eucalyptus Block Storage とインスタンスは Gigabit Ethernet で接続され，それぞれのマシンは外部のネットワークと 10Gigabit Ethernet で接続されている．

表 1　Local site

OS	Cent OS Linux 2.6.18-128.el5xen
CPU	Intel (R) Xeon(TM) 3.60GHz
Main Memory	Initiator：4GB Target：8GB

表 2　Cloud site (Physical machine)

OS	Debian Linux 2.6.26-2-xen-amd64
CPU	Intel(R) Xeon(R) CPU E5530 @ 2.40GHz total 8 core (4 core * 2 socket)
Main Memory	24GByte

表 3　Cloud site (Virtual machine)

OS	Debian Linux 2.6.24-19-xen
Main Memory	1GByte

ストレージについては，ローカルサイトでは Linux iSCSI を用いた． クラウドにおけるストレージについて，インスタンス内のストレージを用いた実験 1，ローカルサイトのストレージを使用し，クラウドから遠隔 iSCSI アクセスする実験 2 を検討した． それぞれの実験の概念図を図 6,7 に示す.通常クラウドでのストレージには，Amazon EC2 と組み合わせて利用できるブロックデバイスである Eucalyptus Block Storage を使用するのが望ましいと思われる．インスタンス内のイメージはマシンをシャットダウンすると消えてしまうが，Block Storage を用いることでデータの中長期保存が可能となる．そのため実験 1 については参考として測定を行った．実験 2 についてはセキュリティポリシなどによりデータをクラウド上に出すことができない企業などが，計算処理のみリモー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遠隔アクセスする環境が想定される．

また前節の「クラウド利用時のデータ配置に関する議論」について，実験 1 は(B)を，実験 2 は(C)を想定している．

アプリケーションをどこで実行する場合も，処理するデータは常にローカルサイトの iSCSI ストレージに保持している．

図 6　実験 1：インスタンス内ストレージ

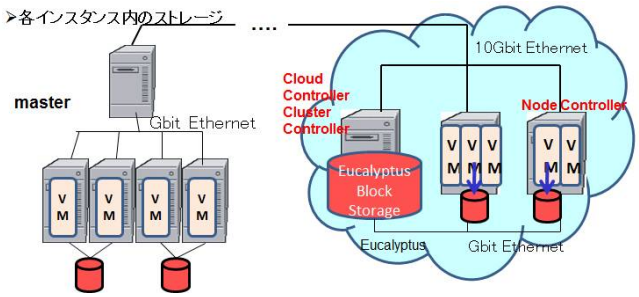
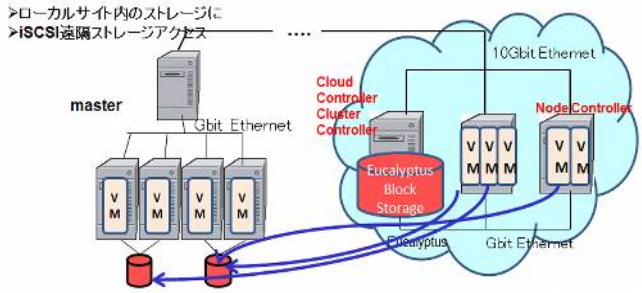


図 7　実験 2: ローカルストレージへの遠隔 iSCSI アクセス



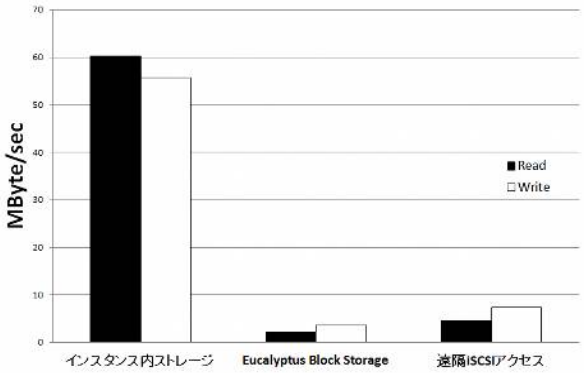
5.3 ストレージアクセス性能の測定（キャッシュなし）

前節で紹介した 2 つのストレージアクセスについて，ディスクキャッシュを排除した際の性能を測定した．（図 8）．参考として Eucalyptus Block Storage を用いた場合の性能も示す.ディスクアクセス性能の測定には dd コマンドを用いた．

その結果クラウドインスタンスからインスタンス内のストレージにアクセスする方法が極めて早いという妥当な結果となった．インスタンスと Block Storage は Gigabit-Ethernet で接続されているにも関わらず，あまりよい性能がでていないことが分かる．Eucalyptus における Block Storage の実体は CLC に置かれるディスクイメージファイルである．このことから Block Storage の実装自体を改善したり，インスタンスからのネットワーク帯域をさらに上げる必要があると言える結果となった．

iSCSI アクセスによりクラウドインスタンスからローカルサイトのストレージへ遠隔アクセスする方法は，遠隔アクセスを行っている割には悪くない性能が出ていることが分かる．アプリケーション実行時にはディスクキャッシュも期待できるため，ネットワーク帯域がある程度大きい場合には，リモートのクラウドからローカルサイト上のストレージに遠隔 iSCSI アクセスをする方式は現実的といえる．

図 8　ディスクアクセス性能(ディスクキャッシュ効果なし)



5.4 ストレージアクセス性能の測定（キャッシュあり）

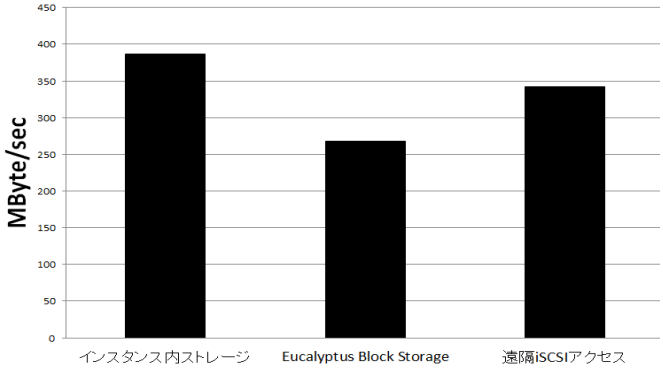
次に dbench を動作させ，ディスクキャッシュの効果が現れる実験を行った（図 9）．こちらも Block Storage の性能を合わせて示す．

ディスクキャッシュがない場合と同様，クラウドインスタンスからインスタンス内のストレージにアクセスする方法が一番早いという結果となったが，どの場合もディスクキャッシュの効果がよく表れている．特に iSCSI 遠隔ストレージアクセスを行う場合は，インスタンス内ストレージを用いる場合に迫る性能が出ている．

以上の 2 つの基礎実験から，リモートのサーバからローカルクラスタのストレージに遠隔ストレージアクセスをする方法

は, 実際のアプリケーションにおいてディスクキャッシュの影響が期待でき, またローカルクラスタのストレージリソースを増設すれば, さらに良い性能が出ることが期待される.

図 9 ディスクアクセス性能(ディスクキャッシュ効果あり)



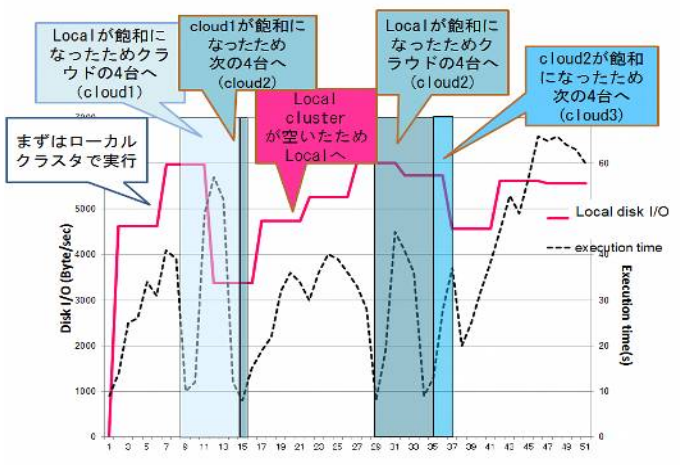
6. ミドルウェア評価実験

以下の実験では, データインテンシブアプリケーションの例として, PostgreSQL と共に配布されているデータベースベンチマークの pgbench を使用した. 評価する際に分かりやすいよう一度に投入する量を 4client と固定し, これを 1 秒間隔で連続的に投入する. 以下にそれぞれの実験結果として, ジョブを 50 回目まで投入した際の結果を示す.

6.1 実験 1: インスタンス内ストレージ使用時

図 10 に実行結果例を示す. 実行が開始されるとまずローカルクラスタでジョブの実行が始まり, 実線で示されるようにローカルクラスタに負荷がかかっている. そして負荷が高くなり, Disk I/O が飽和になるとクラウドに負荷を分散し, またローカルクラスタが空くとローカルで実行が行われる, ということが繰り返されている. ローカルクラスタの Disk I/O が飽和したらクラウドへ処理が分散され, またジョブの実行時間が Disk I/O と対応して制御できていることが分かる.

図 10 実験結果 1 結果

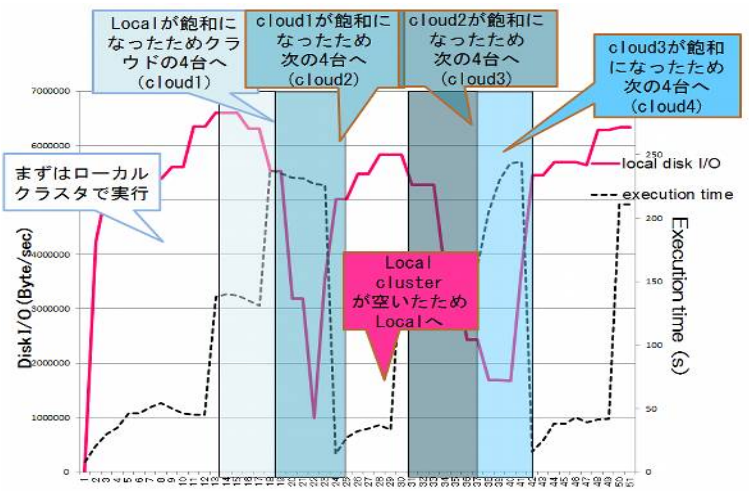


6.2 実験 2: ローカルストレージへの遠隔 iSCSI アクセス実行時

図 11 より, 実験 1 と同様に, Disk I/O と対応して, ジョブの投げ分けが行われている. しかしストレージアクセスがローカルのストレージに集中するため, ローカルクラスタが空いて再度ローカルでの実行が行われても, すぐに Disk I/O が飽和となりクラウドへの投入が再開されている. 図 9 に示されたように遠隔 iSCSI アクセスの性能は割合高く, さらにディスクキャッシュの効果もあり, 全体の実行速度はかなり速くなっている. ただしローカルサイトに設置した iSCSI ターゲットへ多くのサーバからのアクセスが集中し, ここがボトルネックになっている. しかしこれはローカルクラスタにおけるストレージのリソース量の問題であり, クラウドにおいてもストレージにフロントエンドノード上で一括管理される EBS などを用いた場合には同様の問題に直面する.

むしろ実験 2 の実行時間が, 遠隔ストレージアクセスであるにも関わらず, 参考として測定した実験 1 の結果と比べて高々 3〜4 倍程度に収まっている点に注目すべきであると考えられる.

図 11 実験結果 2 結果



7. 最適配置ミドルウェアの評価

このシステム全体のコストは, 実行時間とクラウドの従量制コストの両面から決定される. この両者はトレードオフの関係で, 実行時間を短くするためには大量のインスタンスをクラウドから借りれば良いが, そうするとクラウドの従量制コストが極めて高くなり, 逆に極力クラウドからインスタンスを借りないようにすれば従量制コストは抑えられるが, 実行時間が非常に長くなる. 従って, この両方の値が共にそれを下回るようなケースが他に無い場合を, 最適な状態であると考えることができる.

本研究において, 「Disk I/O が飽和した」とミドルウェアが判断する方法は, ジョブの実行が行われるマシンにて定期的に Disk I/O を測定し, ピークの値がある一定の回数以上変わらなければ I/O が飽和したと判断し, ジョブの投入先を別のマシンに切り替えている. 本ミドルウェアで設定したデフォルトの回数を A とした場合, この値を増減させて例えば 0.8A と 1.2A とし, ジョブを次々投入していった場合のジョブの実行時間とクラウドでのコストの累積をグラフに示す(図 12,13).

クラウドにおける総コストの計算方法は様々に考えられるが, ここではクラウドでジョブを実行する時間が短ければ短いほどよいとし, 実行時間×インスタンス数で計算を行った. 一方, クラウドの従量制コストは実行時間が実行時間×ノード数である.

その結果, 多少のばらつきはあるが, A の値を増減させても実行時間とクラウドの従量制コスト両方が A より低くなることは基本的に無かった.

このことから, A の場合が実行時間とクラウドの従量制コストの両方を最もバランス良く最小にしており, 最適な負荷分散を実行できたと考えられる.

図 12　実行時間累計

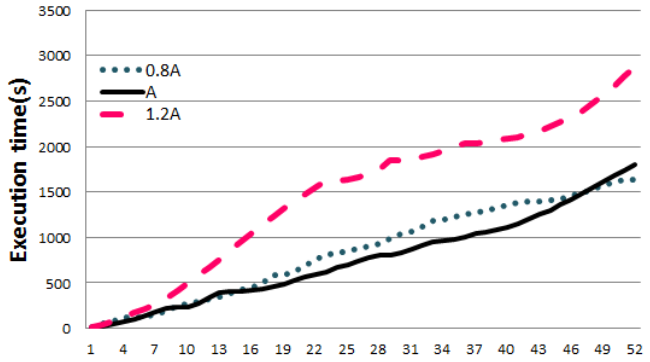
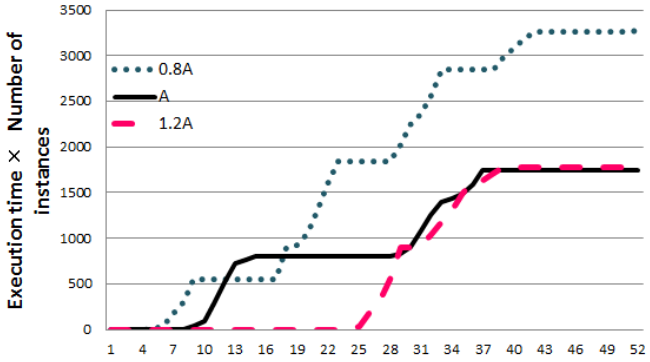


図 13　クラウドコスト累計



7. まとめと今後の課題

本稿ではデータインテンシブアプリケーションを対象とし, ローカルシステムを有効に使いながら, リソースが足りない場合はスケーラブルに増減させたクラウドリソースへ負荷分散を行うミドルウェアを構築した. 負荷の判断には Disk I/O を用い, クラウドにおいてストレージアクセス方法を変化させて実験を行った. その結果 Disk I/O からシステム状態を判断し, それぞれのマシンの負荷が高くなった場合にジョブの投げ分けを制御できていることが確認された.

前述のように本システムの Total cost は, 実行時間とクラウドの従量制コストから成る. そのため今後は, 2 つの指標に基づき本稿で行った実験について評価を行う. またクラウドでのストレージアクセスについて, データを長中期的に保存することが可能な Eucalyptus Block Storage を用いた実験も行う.

本実験では処理に必要なデータが常に処理を行うサーバ上にあることが前提だったが, 今後は処理のアウトソーシングと同時にデータのコピーを行ったり, またクラウドでのインスタンスの起動時間等も考慮にいれるなど, より現実的な環境での実験を行っていく.

[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] Xen：http://www.xen.org/

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

[5] Eucalyptus： http://aws.amazon.com/jp/ec2/

[6] Amazon EC2：http://www.eucalyptus.com/

[7] 豊島 詩織, 山口 実靖, 小口 正人：「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」並列/分散/協調処理に関するサマー・ワークショップ (SWoPP2010), CPSY-6, 金沢, 2010 年8月

[8]GoogleApps :http://www.google.co.jp/apps/intl/ja/business/index.html

[9]GoogleSecureDataConnector :http://code.google.com/intl/ja-JP/securedataconnector/

[10]Aravind Menon,Alan L.Cox,Willy Zwaenepoel : "Optimizing Network Virtualization in Xen",USENIX Annual Technical Conference,2006年

[11JoseRenatoSantos,YoshioTurner,G.(John)Janakiraman,Ian Pratt : "Bridging the Gap between Software and Hardware Techniques for I/O Virtualization",USENIX Annual Technical Conference,2008年

豊島 詩織 Shiori TOYOSHIMA

2011年お茶の水女子大学大学院人間文化創成科学研究科博士前期課程修了, 理学修士. 2009年同大学理学部情報科学科卒業. 大規模分散の研究に従事. 現在は日本電信電話株式会社に勤務.

山口 実靖 Saneyasu YAMAGUCHI

工学院大学工学部情報通信工学科准教授.
2002 年東京大学大学院工学系研究科電子情報工学専攻博士課程修了, 博士(工学). 基盤ソフトウェア, ストレージシステムに関する研究に従事. 日本データベース学会, 情報処理学会, 電子情報通信学会会員各正会員.

小口 正人 Masato OGUCHI

お茶の水女子大学人間文化創成科学研究科教授. 1995 年東京大学大学院工学系研究科電子工学専攻博士課程修了, 工学博士. ネットワークコンピューティング・ミドルウェアに関する研究に従事. IEEE, ACM, 日本データベース学会, 情報処理学会, 電子情報通信学会各正会員.