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

理学専攻　情報科学コース　豊島 詩織　　(指導教員:小口 正人)

1　はじめに

情報システムにおいて扱われるデータ量が爆発的に増加していることから，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，いきなり全ての処理を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．このようにリソースとして伸縮性が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としている点が本研究の特徴である．

2　ミドルウェアの評価について

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，コストも考慮してリソースを配分することが必要である．そのためミドルウェア全体のコストは下式の Total cost として表され，この値の最小になるところがミドルウェアによる最適な負荷分散であると考えられる．

$$Totalcost = a \times Max(T_L, T_R) + b \times T_R \times N_R$$

T_L : Execution time on Local site
T_R : Execution time on Cloud
N_R : Number of Machines

第一項は実行時間，第二項がクラウドの従量制コストを表す．最適化については後述するが，定数 a,b についてはサーバの性能や，サーバからのストレージの位置，実行時間とコストのバランスに関する方針などに基づいて定まる値であり，具体的な数値の決定は本論文の議論の対象外とする．

3　データ処理アプリケーション最適配置ミドルウェア

上記で示した Total cost を最小にするためには，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通常の計算処理と異なり，CPU は I/O 待ちとなっていることが多い．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る．
図 1 は実行するジョブ量を変化させた場合の，アプリケーションの実行時間と 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」状態とし，そのマシンへのジョブの投入を終了する．

以下に最適配置のアルゴリズムをまとめる．ジョブは連続的に投入されていくことを想定している．

(1) ジョブの連続投入開始.
(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マシン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) 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 を測定し，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コストの両方の最小化を目指す．

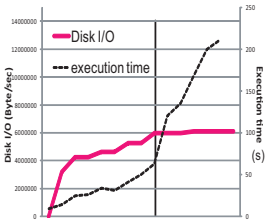

図 1: 実行時間と Disk I/O の関係

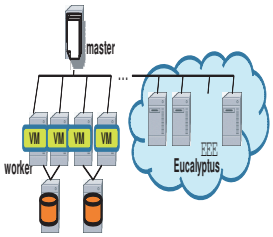

図 2: システム環境

4　ミドルウェア評価実験

4.1　実験環境

図 2 は想定するシステム環境である．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．クラウドには東京大学生産技術研究所が構築した Eucalyptus クラウド (以下 Cloko) を使用した．Eucalyptus は米 Amazon.com 社が提供しているクラウドである Amazon EC2 (Amazon Elastic Compute Cloud) 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 で構築したプライベートクラウドに移すことが可能である．使用できるマシン

台数は，ローカルでは制限があるという現実的な設定とし，一方クラウドでは使用できるマシン台数に制限がないものとした．

ストレージについては，ローカルでは iSCSI ストレージを用いる．クラウドではインスタンス内のストレージを用いた実験 (実験 1) と，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 (実験 2) を行う．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状況が想定される．

以下に実験 1,2 を示す．実験にはデータベースベンチマークの pgbench を使用た．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，これを連続的に投入する．

4.2 実験 1:インスタンス内ストレージ

実験 1 ではクラウドにおけるストレージとしてクラウドのインスタンス内のストレージを用いる．図 3 にジョブを 50 回目まで投入した際の結果を示す．実行が開始されるとまずはローカルクラスタでジョブの実行が始まる．そして負荷が高くなるとクラウドに負荷を分散し，またジョブが終了してローカルクラスタが空くとローカルで実行される，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．

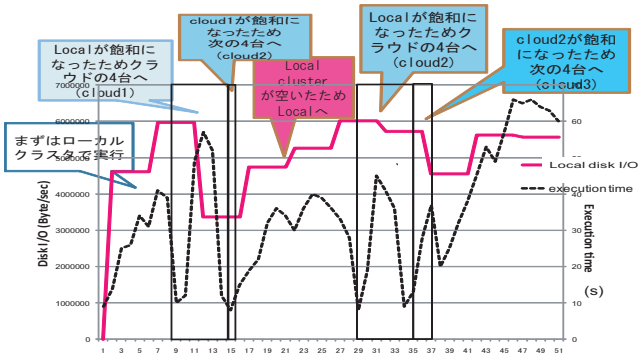

図 3: 実験結果 1(インスタンス内ストレージアクセス)

4.3 実験 2:ローカルストレージへの遠隔 iSCSI アクセス

実験 2 ではクラウドにおけるストレージはローカルサイトのストレージを用い，クラウドから iSCSI 遠隔ストレージアクセスを行う．図 4 にジョブを 50 回目まで投入した際の結果を示す．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ことが分かる．しかし実験 1 とは異なり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．

5 最適配置ミドルウェアの評価

本研究では，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

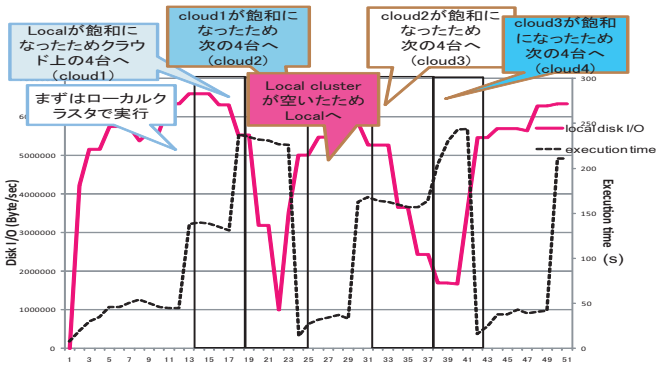

図 4: 実験結果 2(ローカルストレージへの遠隔 iSCSI アクセス)

投入先を別のマシンに切り替えている．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 5，6)．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コスト両方が A より低くなることが基本的に無かったことから，A の場合が最も実行時間とクラウドのコストを両立していると言える．

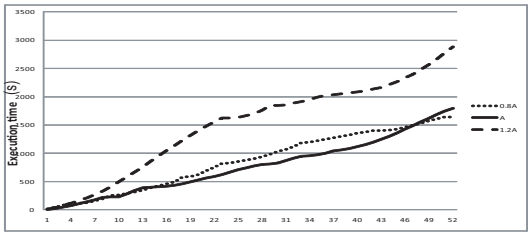

図 5: 実行時間累計

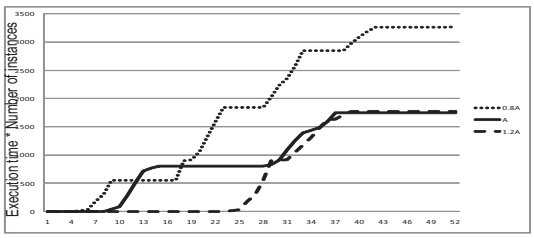

図 6: クラウドコスト累計

6 まとめと今後の課題

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．その結果 Disk I/O に応じて適切にクラウドに負荷分散できていることが確認された．今後はこの結果や Total cost の式をもとに，ミドルウェアの評価を行う．