M-011

大規模疎結合分散システムにおける安定的性能確保のための ミドルウェアシステムの特性とその評価

阪本　憲司†　　吉田　誠‡
Kenji Sakamoto　　Makoto Yoshida

1.　はじめに

近年，コンピュータネットワークによるオンラインシステムは社会を支える重要な基盤となっており，高い信頼性と，ユーザの要求に応えることができる性能が常に要求されている[1, 6]．一方，オンラインシステムの性能は，システム負荷（トラヒック）の影響を大きく受ける．例えば，処理しなくてはいけない要求がバースト的に到着し，システムの想定する負荷を超えてしまうと，システムがダウンしてしまう危険性も存在する．しかしながら，これらの負荷を事前に予測することは難しい．このような問題や要求に応えるための手法として PC クラスタとグリッドコンピューティングがある[1,3,4]．

一般のシステムにおいては，非常に多くの負荷（要求）が同時に発生する場合には，システムを構成するマシンの性能を要求到着のピークに耐え得る程度に向上させる手段が必要となる．ハードウェア強化による方法もあるが，この方法を用いる場合コンピュータの価格は非常に高価なものとなってしまう[6]．ピーク時のためだけにコンピュータリソースを増やすのは無駄が多く，コストが余分にかかるという問題が生じる．一方，分散システムでは仕事をしていないアイドル状態のマシンの計算資源を利用するという手法が考えられる[1,5]．この方法であれば，他のサイトが要求を処理しきれなくなった場合に，計算資源を貸せば良いだけなので性能の特別高いハードウェアを用いてシステムを構築することなく，要求されるシステムの性能を全体的に満たすことが可能となる．

本論文では，多数の PC をネットワークで接続した大規模疎結合分散システムを取り上げ，そのパフォーマンス特性を観察・評価する．サイト間でオブジェクトを移動させることにより，サイトが過負荷状態になることを防ぎ，安定的性能の確保を可能とするミドルウェアの特性とその評価について記述する．著者らは，まずミドルウェアのプロトタイプシステムを実装し，オブジェクト移動のオーバーヘッドを測定した[7]．そして，その測定結果をもとに大規模疎結合分散システムモデルを作成し，シミュレーションにより評価した．

2 章では現実に対象となるシステムのモデルを示し，3 章ではシミュレーションにより分散システム特性を観察する．最後に 4 章で本論文のまとめと今後の課題について述べる．

2.　モデル

本章では，実際にオブジェクト移動を可能とする分散システムのモデルについて説明する．図 1 にその構成図を示す．対象となるシステムは複数台のサイトによって構成される分散システムであり，分散システム内では各サイトが

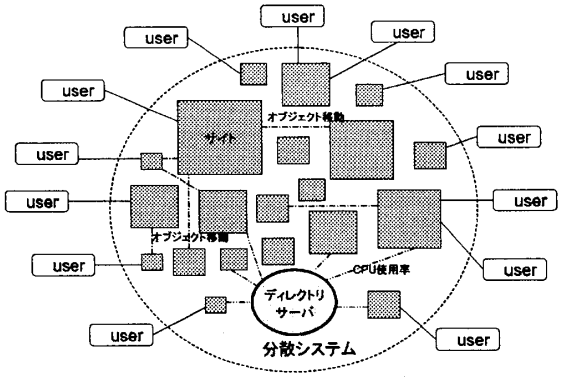

図 1.　分散システム環境モデル

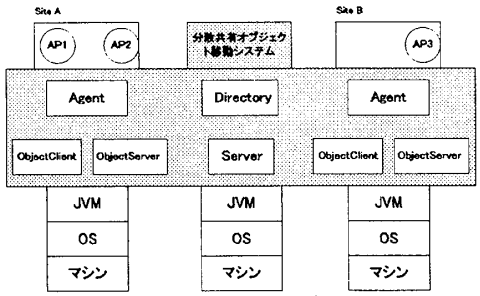

図 2.　システム構成図

ネットワークを介して接続されている．本モデルはメッシュ型のネットワークトポロジとしている．実際のシステムとしては大規模疎結合 PC クラスタを想定している．PC クラスタには，各種処理能力の異なるコンピュータが多数存在し，それらがクラスタを構成している．図 1 に示す分散システムにはユーザ（人，もしくはアプリケーション）からの要求が送られ，この要求を受け取ったサイトは分散システム内で求められた処理を行い，結果を返却する．

3.　シミュレーション

著者らは実際に計算資源を共有するためのミドルウェアのプロトタイプを実装した[7]．そして，その実測データを基にシミュレーションモデルを構成し，オブジェクト（トランザクション）移動のシミュレーションを行った．図 2 に実装したシステムのシステム構成図を示す．以下，計算資源を共有するためのミドルウェアについて説明し，シミュレーションを行う際に用いるシミュレーションモデル，モデルのパラメータと観測データ，そして実行結果とその考察についてそれぞれ述べる．

3.1.　プロトタイプシステム

分散オブジェクトを用いる分散システムでは，オブジェクトの配置はシステムの性能に重要な役割をはたす．オブジェクト配置を動的に変化させるミドルウェアについて説明する．

†岡山理科大学大学院工学研究科
‡岡山理科大学工学部情報工学科

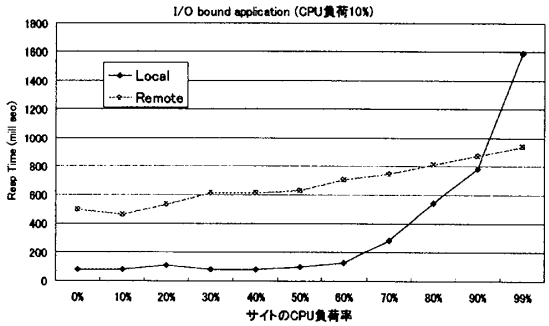

図 3. 応答時間の変化（I/0 バウンド時）

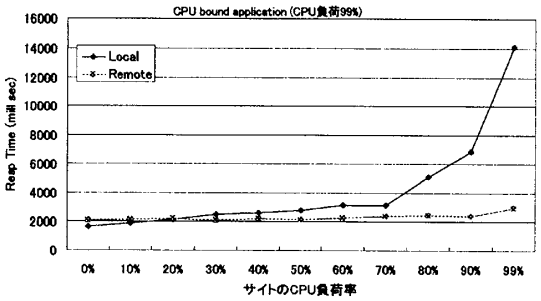

図 4. 応答時間の変化(CPU バウンド時)

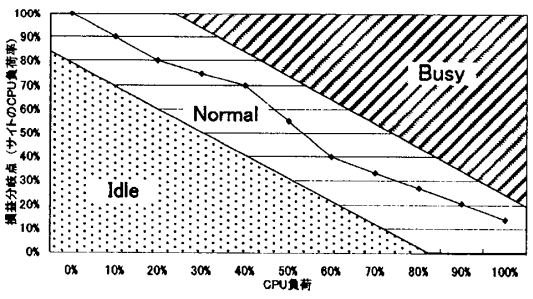

図 5. CPU 負荷に伴う損益分岐点

著者らは実装したプロトタイプシステムを使用して，ローカルと，リモートにオブジェクトを利用する場合の比較実験を行い，その利得が逆転する損益分岐点を計測した．この実験では一つのサイトに着目し，マシンに負荷(CPU 負荷)をかけた状態で，オブジェクト(アプリケーション)の特性を変化させることにより損益分岐点を観測した．オブジェクトが I/O バウンド（CPU 負荷 10%）である場合の結果を図 3 に，オブジェクトが CPU バウンド（CPU 負荷 99%）である場合の結果を図 4 に示す．

図 3 が示すように，I/O バウンドの場合はサイトの CPU 負荷が 90%を超えなければリモートにオブジェクト移動を行う利点が認められない．一方，CPU バウンドであればサイトの CPU 負荷が 20%を超える程度でオブジェクト移動の利点を確認できる．図 5 は，CPU 負荷を変化させてその損益分岐点をプロットした図である．

図 5 のグラフの下の部分がローカルでのオブジェクト利用が有利である領域，上の部分がリモートでのオブジェクト利用が有利である領域であることを示している．サイトの負荷状況（縦軸）と対象となるオブジェクトの特性（横軸）が解っていれば，図 5 を基にして動的にオブジェクトを移動させて，リモートに利用するべきか，それともローカル

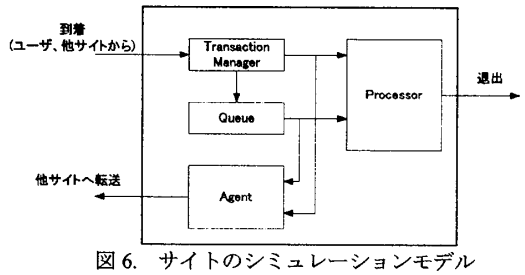

図 6. サイトのシミュレーションモデル

でオブジェクトを利用するべきかを判断することが可能となる．

3.2. シミュレーションモデル

シミュレーション環境を以下とした．分散システムはネットワークで接続された多数の処理能力の異なるコンピュータによって構成されている．

各サイトのシミュレーションモデルを図 6 に示す．ユーザから投入されたトランザクションはプロセッサが空いていればサービスを受ける．空いていない場合は待ち行列でプロセッサが空くのを待つ．損益分岐表を参照し，移動を行う場合はエージェントが他のサイトと通信を行い，受け入れ可能なサイトがあれば，そのサイトにトランザクションを転送する．

本モデルでのオブジェクト移動は，プロトタイプの実測から得られた図 5 に示す損益分岐表をもとに制御されている．今回の実験では 2 重閾値方策を用いて負荷分散制御を行っている．図 5 に示すように，サイトの CPU 使用率が高い状態を「BUSY」，CPU 使用率が低い状態を「IDLE」，その中間を「NORMAL」としている．BUSY，もしくは NORMAL 状態のサイトにトランザクションが到着した場合は，IDLE もしくは NORMAL 状態のサイトにトランザクションを転送し，リモートサイトで実行を行う．移動できるサイトが存在しない場合，BUSY 状態のサイトではトランザクションは待ち行列に格納され，現在実行中のトランザクションの終了を待ち，NOMAL 状態であれば到着したサイトで実行する． IDLE 状態のサイトのトランザクションが到着した場合には，トランザクションは到着サイトでローカルに実行される．また，BUSY 状態のサイトでは一定時間間隔で他のサイトを検索し，IDLE か NORMAL 状態のサイトが存在すれば，待ち行列にあるトランザクションを他のサイトに移動させている（図 6 参照）．

3.3. パラメータと観測データ

シミュレーションモデルで用いられる各種のパラメータついて記述する．シミュレーションはイベントドリブン方式を使用した[2]．

以下にシミュレーションモデルのパラメータを示す．

1) サイト数
分散システムを構成するサイトの数
2) 平均到着率
単位時間あたりにサイトに到着する平均のトランザクション数
3) 平均サービス率
サイトの単位時間あたりのトランザクション処理能力，各サイトの処理能力により変化する
4) 平均の処理能力
サイトに与える処理能力の平均，この値を変化させると

システム全体の性能が変わる

5) 閾値の大きさ
トランザクション移動条件．閾値の幅が大きくなると移動条件が緩やかになり，小さくなると移動しにくくなる

6) メッセージ通信（移動問い合わせ）遅延
移動を試行する場合の他サイトへの通信問い合わせの遅延時間

7) 移動遅延時間
オブジェクトが移動する際に発生する，オブジェクト移動時間

すべてのトランザクションに対して，発生時間，待ち時間，移動時間，サービスの開始時間，終了時間，移動回数，移動失敗回数，移動履歴，を観測した．また，観測データから，トランザクションの観測結果をサイトごとにまとめて，各サイトで以下の結果を算出した．

- 平均応答時間：応答時間の平均
- 平均スループット：サイトの単位時間あたりの処理の平均
- トランザクションの移動割合：移動操作を行ったトランザクションの割合
- トランザクションの移動失敗率：移動試行が失敗する割合

なお，サイトに到着したトランザクションは他のサイトでサービスを受けても，最終的には到着サイトに返却されるようにしている．

3.4. シミュレーション結果

3.4.1. オブジェクト移動に伴う応答時間の変化

トランザクションの移動を行った場合と行わなかった場合の応答時間の比較を行った．トランザクション数の変化に伴う応答時間の変化を図 7，図 8，及び表 1 に示す．各サイトの CPU パワーの大きさは正規分布に従うものとし，最大の処理能力を持つサイトと最小の処理能力を持つサイトの性能の比率を 2：1 とした．閾値の幅は２０％とし，図５に示す損益分岐点の前後 10%としている．

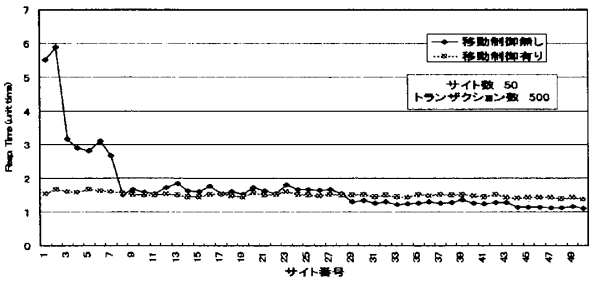

図 7. 応答時間の変化（トランザクション数 500）

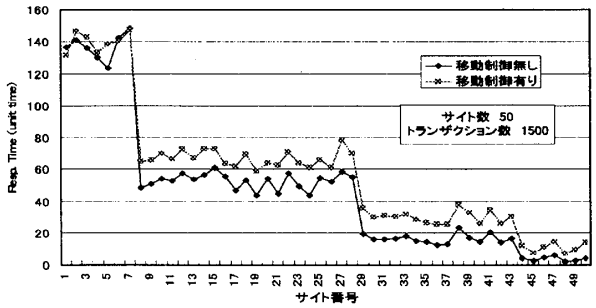

図 8. 応答時間の変化（トランザクション数 1500）

各サイトに到着するトランザクションは正規分布とした．図ではサイト番号が大きいほどCPUパワーが大きくなっている．

システムへのトランザクションの投入数が少なく，かつ移動を行う場合，応答時間は CPU パワーの低いサイト群で大幅に向上している（図 7 参照）．システム全体としても，平均応答時間は１４％（表１の応答時間から算出）向上している．CPU パワーの大きなサイト群では移動を行わなかった方が良好な結果を示している．これはパワーの低いサイトから高いサイトへトランザクションが移動したことによって，負荷の均一化が行われたためと考えられる．

表 1. 応答時間とスループット

要求数（個）	移動無し		移動有り	
	応答時間 (unit time)	スループット (個 / unit time)	応答時間 (unit time)	スループット (個 / unit time)
500	1.76	0.51	1.51	0.73
1000	6.77	1.01	3.01	1.35
1500	46.70	1.49	58.28	1.66

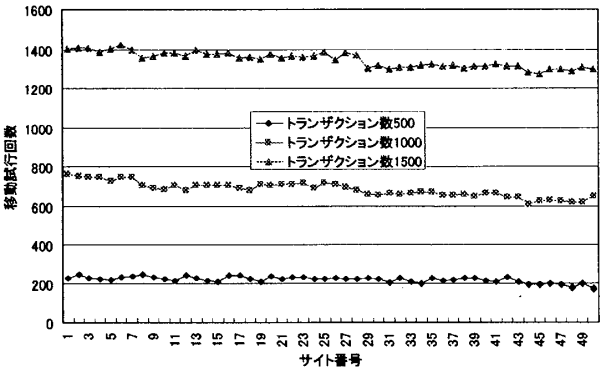

図 9 移動試行回数

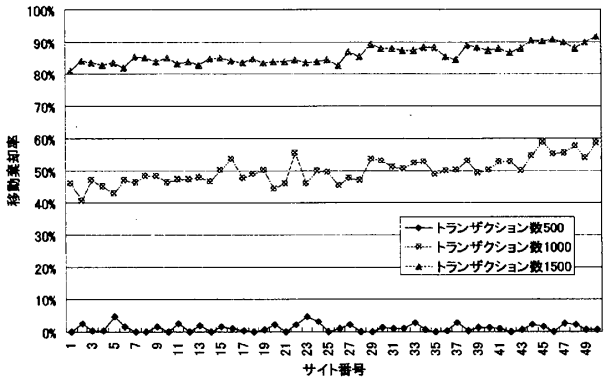

図 10. 移動棄却率

処理するべきトランザクションが非常に多い場合は，移動処理を行わない方が良い結果となっている（図 8 参照）．平均応答時間も２５％悪化（表 1 から算出）している．これはシステム全体の処理能力に対して，トランザクション全体の負荷の方が大きくなったためで，各サイトが移動を行おうとしても移動が行えなく，通信のオーバーヘッドだけが余分にかかっているためと考えられる．

図 9 にトランザクションの移動試行回数，図 10 にトランザクション移動棄却率（移動要求をして拒否された割合）を示す．

到着システムへの負荷が大きくなればなるほど，移動試行回数は増えていくことがわかる．移動棄却率はシステム全体が輻輳状態になると急激に増えていることが観察される．これらを総合的に考えれば，システム全体に対して投入されるトランザクションの負荷が大きくなりすぎると，移動試行による通信のオーバーヘッドだけが余分にかかってしまい，安定したレスポンスタイムが期待できないことがわかる．よって，システムの負荷状態に応じて適応的に，オブジェクトを移動させるかどうかを判断し，制御する機構が必要となる．例えば，棄却率が一定値を超えると，移動試行を一定期間行わないという操作を行う必要がある．

3.4.2. オブジェクト移動に伴うスループットの変化

オブジェクト移動を行う場合と，行わない場合のスループットの比較を行った．トランザクション数の増加によるスループットの変化を図 11，図 12 に示す．トランザクションの移動を行う場合の方が行わない場合よりもスループットは全体的に良くなっている．スループットは，トランザクション負荷が大きくなると減少傾向にあるが，平均スループットはオブジェクト移動しない場合に比べてほとんどの場合において向上している．表 1 から１１%から４３%向上していることが観察される．また，移動を行う場合でも性能の優れているサイトのスループットの方が一層高くなる傾向が観察される．

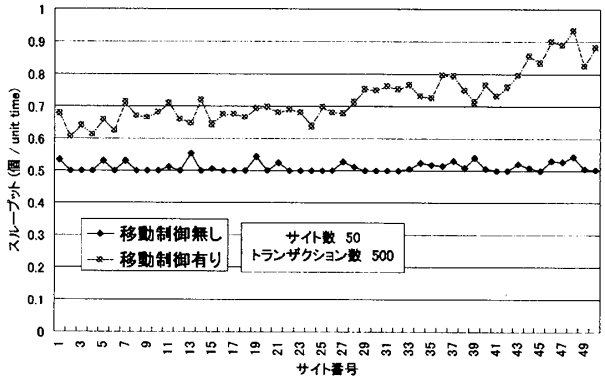

図 11. スループットの変化（トランザクション数 500）

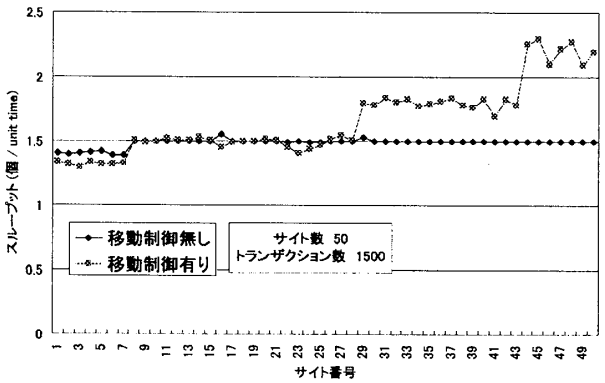

図 12. スループットの変化 （トランザクション数 1500）

4. おわりに

本論文では，オンラインシステムにおいて大規模疎結合分散システム上で安定的性能を確保するためのミドルウェアシステムの特性についてシミュレーションを行い評価した．まず，計算資源を共有するためのミドルウェアのプロトタイプを実装し，オブジェクトの転送・実行時間，オブジェクト移動の効果などシミュレーションに必要な実際のデータを取得した．オブジェクト移動のための損益分岐表を実データから作成し，オブジェクト移動のためのシミュレーションモデルを構築した．そして，大規模システムを構築した際のシステムの振る舞いをシミュレーションにより評価した．シミュレーション結果から以下が観察された．

- オブジェクト移動による応答時間の有益性は負荷状況により異なる．負荷の状況により，平均応答時間が５０%以上向上する場合もあるが，急激な負荷が生じた場合には，１０%低下となる場合もある．平均スループットは，オブジェクト移動を行った方が常に良い．
- 過負荷状態ではオブジェクト移動を行わない方が良い．
- 移動制約条件と応答時間は比例関係にある．制約が緩いと応答時間は悪くなる．
- 棄却率が一定値を超えると，移動試行を一定期間行わない方が良い．

大規模 PC クラスタ上での過負荷状態に対して，他のサイトにオブジェクトを移動させることで，未然に過負荷状態を防ぎ，安定的な応答を獲得できる制御方法について考察した．

今後は，ネットワークトポロジが動的に変化するようなグリッドコンピューティング，モバイルコンピューティングへの適用を考えるとともに，動的にネットワークを形成する仕組みについて検討していく予定である．

参考文献

[1] Andrew S.Tanenbaum, Maarten Van Steen: Distributed Systems Principles and Paradigms 2nd Edition, pp17-31, Pearson Prentice Hall (2007).

[2] Sukumar Ghosh, Distributed Systems an Algorithmic Approach, pp303-310. Chapman&Hall/CRC(2007).

[3] Srikumar Venugopal, Rajkumar Buyya, Kotagiri Ramamohanarao: A Taxonomy of Data Grids for Distributed Data Sharing, Management, and Processing, ACM Computing Surveys, Vol.38, (March 2006).

[4] Stephanos Androutsellis-Theotokis, Diomidis Spinellis: A Survey of Peer-to-Peer Content Distribution Technologies, ACM Computing Serveys, Vol.36, No.4, pp335-371 (December 2004).

[5] Marta Patiño-Martínez, Ricardo Jiménez-Peris, Bettina Kemme, and Gustavo Alonso: MIDDLE-R: Consistent database replication at the middleware level, ACT TOCS, Vol.23, No.4, pp375-423 (2005).

[6] Valeria Cardellini, Emiliano Casalicchio, Michele Colajanni and Philip S. Yu: The state of the art in locally distributed Web-Server systems, ACM Computing Surveys, Vol.34, No.2, pp263-311 (2002).

[7] 阪本憲司，吉田誠 ： オブジェクト移動を可能とするミドルウェアの開発と評価，電気・情報関連学会中国支部連合大会 講演論文集，pp240-241 (2006).