M-012

ミドルウェアを用いた広域的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負荷分散方式とその評価

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]．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]．広域的に分散したシステムでは、システムを構成するサイトへの要求の到着ピークは異なるため、余っている計算資源をより有効に利用することができる．そのためには、サイト間でのオブジェクトの移動と，移動先決定アルゴリズムにより、サイトの過負荷状態と安定的性能確保を可能とするミドルウェアが重要となる[5]．

前論文[8]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到着分布を正規分布としたシミュレーションを行い評価した．そして，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．

本論文では，到着分布及び移動制御方式の変化に伴うパフォーマンス特性の変化について報告する．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．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を用いたアプローチで提案手法を評価する．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ネットワークを介して接続されている．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求められた処理を行い，結果を返却する．

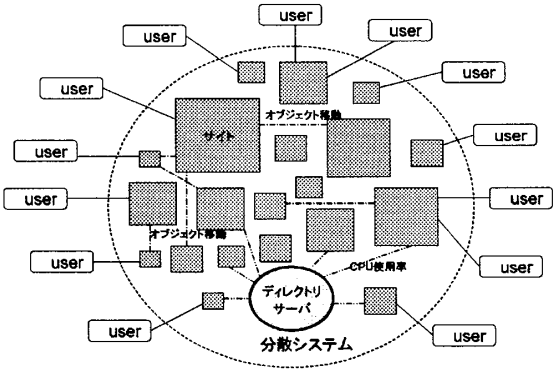

図 1. 対象となるシステムモデル

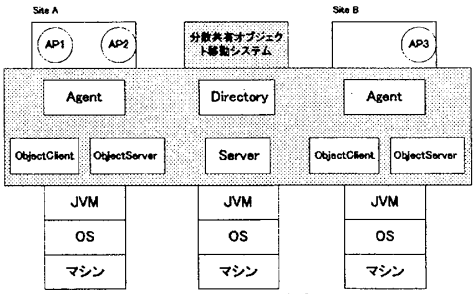

図 2. システム構成図

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

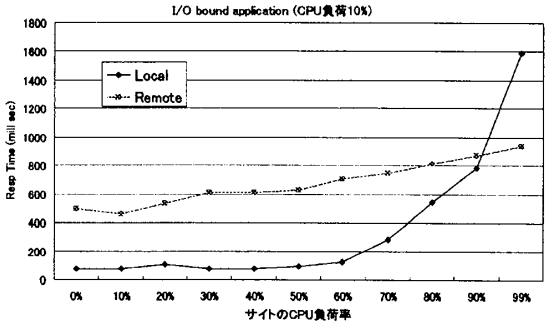

図 3. 応答時間の変化（I/0 バウンド時）

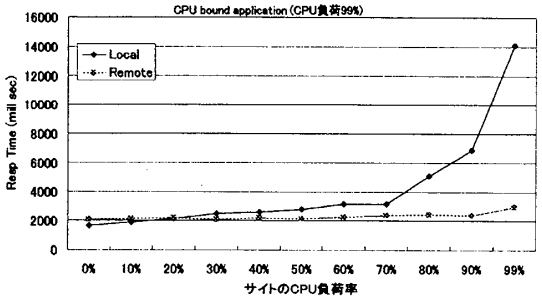

図 4. 応答時間の変化(CPU バウンド時)

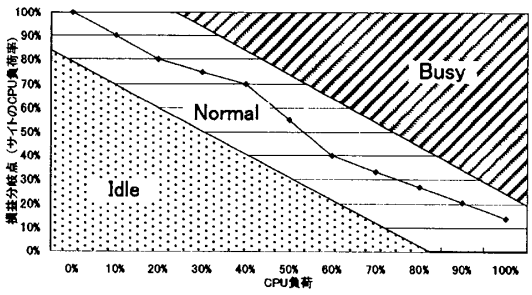

図 5. CPU 負荷に伴う損益分岐点

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，それともローカルでオブジェクトを利用するべきかを判断できる．

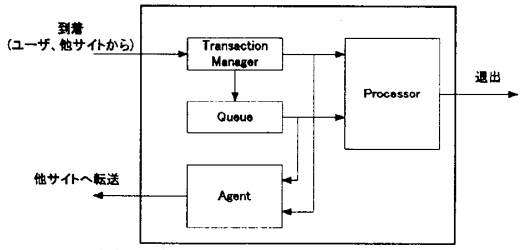

図 6. サイトのシミュレーションモデル

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[8]．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と観測対象となるデータについて述べる．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]．

以下にモデルのパラメータを示す

1) サイト数
分散システムの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
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，値が大きくなると移動しやすく，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
トランザクションが移動する際に発生する，トランザクション転送遅延時間

観測するデータは発生，待ち，転送，サービス開始時間，サービス終了時間，移動回数，移動試行回数，参照回数，移動失敗回数である[8]．これらのデータを集計して以下の結果を得た．また，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た．

- 平均レスポンスタイム
  レスポンスタイムの平均
- 平均スループット
  サイトの単位時間あたりの処理数の平均
- 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
- 平均参照回数
  移動先サイトを見つけるために必要な通信回数
- 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

3.4. シミュレーション結果

到着の分布と移動制御方式による変化

移動制御アルゴリズムを用いる場合と，用いない場合の差について実験した．制御アルゴリズムは以下の 3 種類の方法を用いる．

- <ランダム制御>
  移動先をランダムに選択する
- <CPU パワー制御>
  CPU の性能の高い順番に選択する
- <負荷状況制御>
  サイトの中で負荷状況が最も軽いサイトを選択する

システム全体への到着分布は一様分布に近い分布とした．システム全体への到着を図 7 に示す．表 I ～IVに投入トランザクション数を変化させた場合の結果を示す．これらを図 8～11 のグラフにプロットした．レスポンスタイム（図 8）では，移動を行わない場合には 2500 個のトランザクションを投入すると平均のレスポンスタイムが 17.07 となり，あっかしはじめる．移動を行うと場合には，移動先決定の制御方式により結果は異なるが，<ランダム制御>では 4000 トランザクションの負荷で 13.7 であり，<CPU のパワー>と<負荷状況>を基にする制御方式ではトランザクション負荷 4500 で 11.49，11.19 となる．スループット（図 9）では移動の有無，またすべての制御方式でトランザクション 4000 の負荷が最も高いスループットである．その中でも<負荷状況>に応じた制御方式が最も高く 6.99(個/unit time)となった．制御アルゴリズムでの平均参照回数（図 10）は<負荷状況>に応じて移動先を選択する制御方式が最も回数が多い．応答時間と参照回数については以下の式で表せる．

$$\text{応答時間} = \text{処理時間} + \text{参照回数} \times T$$

ここで，T はメッセージ通信の時間である．制御方式によって，平均の参照回数が異なる．応答時間は<負荷状況>がもっとよく，次に<CPU パワー>，そして<ランダム>という結果になる．しかし，参照回数が少ないのはランダム，CPU パワー，そして<負荷状況>が最も回数が多い．棄却率が大きくなると，トランザクションの移動を行わない方が有益であることが分かっている[8]．棄却率 40%を超えると応答時間が悪くなっていることが確認できる．

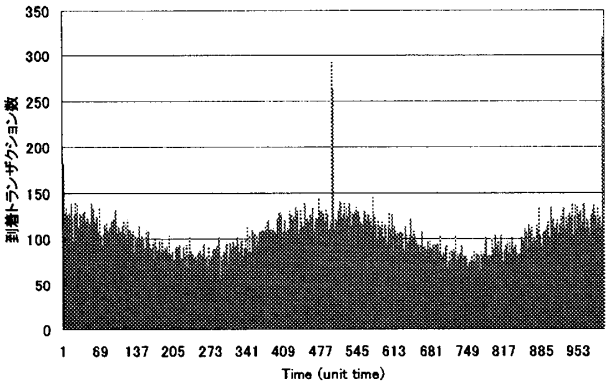

図 7. システムへ到着（総トランザクション数 2000）

表 I． レスポンスタイム

応答時間(unit time)	移動無し	ランダム	CPUパワー	負荷状況
1000(traffic)	1.04	1.20	1.21	1.20
1500	1.21	1.35	1.37	1.35
2000	4.02	1.50	1.52	1.50
2500	17.07	1.63	1.65	1.63
3000	38.36	1.80	1.76	1.73
3500	61.72	3.97	1.88	1.83
4000	87.17	13.79	3.06	2.78
4500	128.83	37.58	11.42	11.19
5000	162.02	83.72	42.41	42.21

表Ⅱ．スループット

スループット	移動無し	ランダム	CPUパワー	負荷状況
1000(traffic)	1.00	1.20	1.19	1.20
1500	1.49	2.02	1.98	2.02
2000	1.91	2.98	2.87	2.98
2500	2.32	4.02	3.83	4.03
3000	2.66	4.92	4.82	5.11
3500	3.25	5.41	5.87	6.21
4000	3.54	5.72	6.55	6.99
4500	3.14	4.76	6.48	6.69
5000	3.19	4.47	5.34	5.36

表Ⅲ．平均参照回数

参照回数	ランダム	CPUパワー	負荷状況
1000(traffic)	1.29	2.05	40.14
1500	1.52	3.77	31.53
2000	2.26	6.11	28.70
2500	2.76	8.87	27.90
3000	3.23	11.74	27.54
3500	3.81	14.92	27.38
4000	4.55	17.79	27.61
4500	5.80	20.13	28.05
5000	6.43	21.12	29.64

表Ⅳ．棄却率

棄却率	ランダム	CPUパワー	負荷状況
1000(traffic)	0.00%	0.00%	0.00%
1500	0.47%	0.00%	0.00%
2000	0.81%	0.21%	0.12%
2500	1.44%	0.59%	0.43%
3000	10.54%	0.89%	0.65%
3500	34.74%	1.80%	1.30%
4000	51.87%	19.12%	12.02%
4500	71.04%	39.97%	37.32%
5000	81.04%	68.19%	67.79%

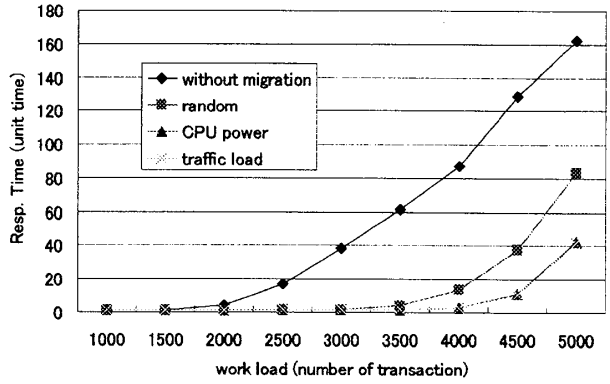

図 8. レスポンスタイムの変化

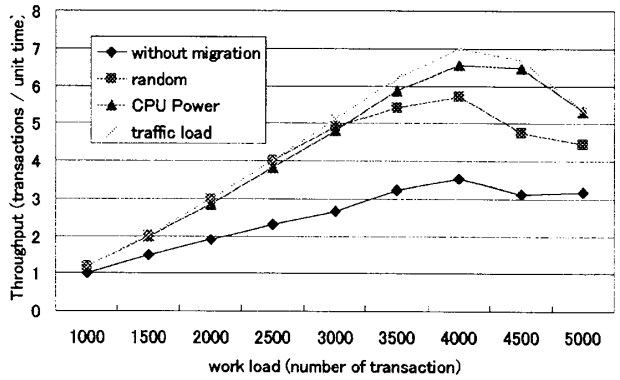

図 9. スループットの変化

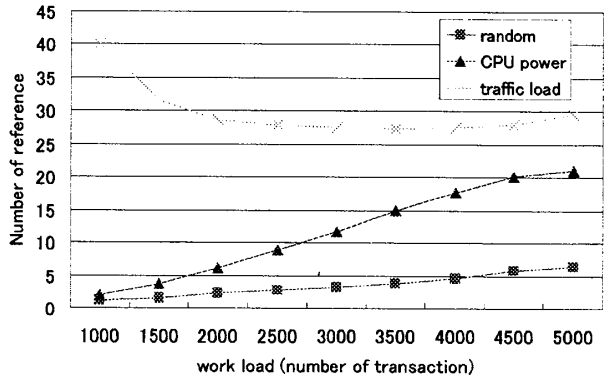

図 10. 移動 1 あたりの平均参照（メッセージ通信）回数の変化

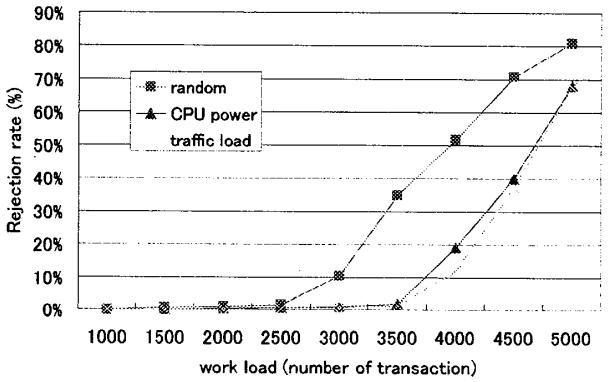

図 11. 棄却率の変化

4. おわりに

本論文では，オンラインシステムにおける負荷に対応して安定的な性能を発揮させるための，広域的大規模疎結合 PC クラスタの特性について評価した．すで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済みであり，このミドルウェアの特性は実際の動作を監視，測定することで行っている．これらの実測データを基にして，広域的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．シミュレーションから以下が観察された．

- 負荷状況制御や CPU パワー制御に基づいた移動方式を用いるとランダム制御より多くの負荷に対応できる
- 移動制御を行うと参照による通信回数が増える
- 制御方式を複雑にするほど通信回数も増加する

以上の結果から，ランダムに移動先を決定するよりは制御アルゴリズムを用いて移動先を決定するほうがレスポンスタイムやスループットの面からも有効的であることが確かめられた．第 3.4.1 節の計算式に示したように，参照回数が増加すれば，応答時間は変化する．これらのトレードオフを考慮した制御方式を完成させる必要がある．低負荷状況では<負荷状況制御>が有効であり，高負荷状況では<CPU パワー制御>有効となる．

移動先を見つけるための参照回数は制御アルゴリズムを用いることによって発生するオーバーヘッドである．今後は，参照回数を抑えつつ最適な移動先を発見するためのアルゴリズムについて研究を行っていく．

参考文献

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

[2] Haifeng Yu and Amin Vahdat: The costs and limits of availability for replicated services, ACM TOCS, Vol.24, No.1, pp70-113 (2006).

[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).

[8] K.Sakamoto, M.Yoshida: Design and Evaluation of Large Scale Loosely Coupled Cluster-based Distributed Systems, IFIP International Conference on Network and Parallel Computing-PAMA 2007 (2007).

[9] Alonso R, Cava L, Sharing Job Among Independent Owned Processor, Proc. Of 8th ICDCN, (1988).