RL-005

分散デスクトップコンピューティングにおける安定的性能確保のための負荷分散制御方式の設計と評価

阪本　憲司†　　吉田　誠††

Kenji Sakamoto　　Makoto Yoshida

1.　はじめに

コンピュータネットワークを用いたオンラインシステムは重要な社会基盤として機能しており，高い信頼性，ユーザの要求を満たす高速な処理能力などが常に要求されている[1, 2]．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）の影響を強く受ける．例えば，ユーザからの処理要求がバースト的に到着し，システムが想定していた負荷を超えてしまうと，システムがダウンする危険性も存在する．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．

一般のシステムにおいては，非常に多くの負荷（要求）が同時に発生することが想定される場合は，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[4]．また，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]．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対象とし，負荷の分散を行うことにより，システムパフォーマンスの向上と安定的動作を行うことを目的とする．本研究成果はグリッドコンピューティングシステムだけではなく，P2P システムへの適用も可能である[2, 3]．

著者らは，まず各マシンの計算資源を共有するためのミドルウェアプロトタイプを実装した[5]．次に，プロトタイプが負荷分散を行う場合のオーバーヘッドを測定した[5]．こ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，このプロトタイプを適用した計算資源共有による，負荷分散の有用性について，シミュレーションにより評価を行った[6, 8, 9, 10]．本論文では 4 つの負荷分散制御方式を比較する．

2 章ではこれまでの研究の過程を示し，3 章では対象となるシステムのモデルや負荷分散制御方式について説明する．4 章でシミュレーションの結果とそれについての考察を述べる．最後に 5 章で本論文のまとめと今後の展望について述べる．

2.　プロトタイプシステム

分散オブジェクト技術を用いて構築されたシステムでは，仕事を行うオブジェクトを移動させることにより，仕事を他のマシンに振分けることが可能となる．このオブジェクト移動を利用した負荷分散を行うためのミドルウェアプロトタイプを試作した．そして，プロトタイプを用いたオブジェクト移動の特性を実測し，確認した[5]．プロトタイプのシステム構成図を図 1 に示す．

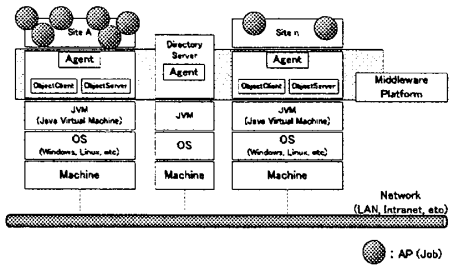

図 1.　システム構成図

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，応答時間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．オブジェクトの CPU 負荷が 10%である場合の結果を図 2 に，CPU 負荷が 99%である場合の結果を図 3 に示す．

図 2 が示すように，CPU 負荷率 10%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 負荷率 99%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 4 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 4 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 4 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することが可能となる．

3.　シミュレーション

本章では，実際にオブジェクト移動を可能とする大規模疎結合分散システムのモデルについて説明する．

3.1.　システムモデル

図 5 にオブジェクト移動を用いる，システムの構成図を示す．対象となるシステムは数十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．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

* †　岡山理科大学大学院工学研究科情報工学専攻

†† 岡山理科大学工学部情報工学科

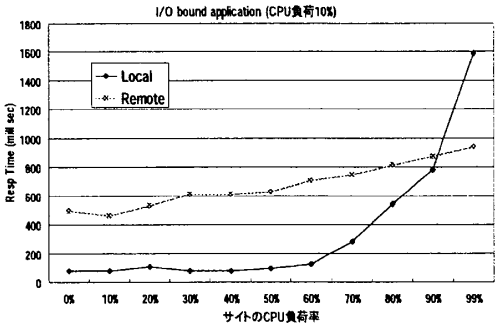

図 2. 応答時間の変化

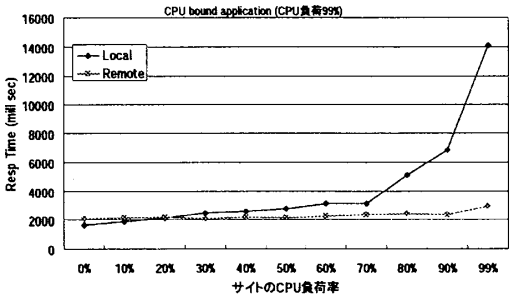

図 3. 応答時間の変化

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 5 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．

3.2. 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．状態マップを参照し，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．

3.3. 状態マップ

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 4 に示す状態マップ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 4 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

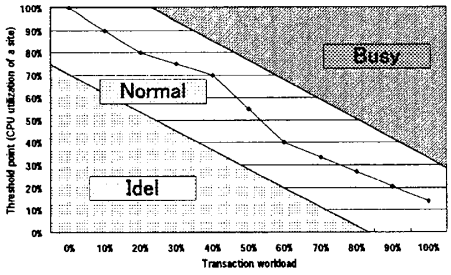

図 4. CPU 負荷に伴う損益分岐点

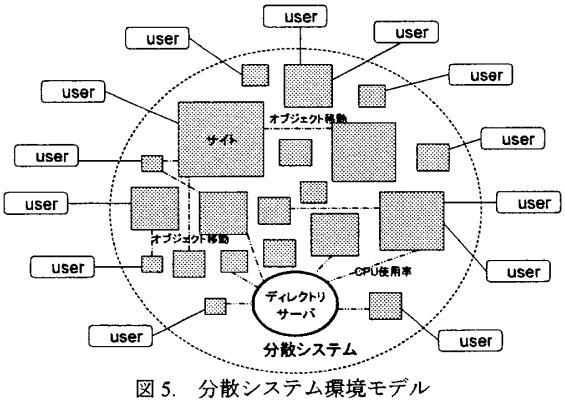

図 5. 分散システム環境モデル

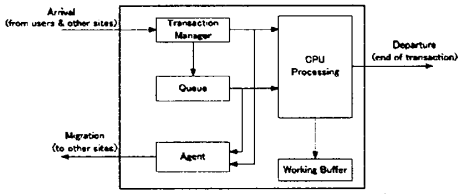

図 6. サイトのシミュレーションモデル

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）．

3.4. 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．

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) 到着分布 ：到着する要求が従う分布，”正規分布”，”一様分布”，”ポアソン分布”を設定する
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) 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に発生する，他サイトへの通信問合せによる遅延時間
6) 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．

表 A. レスポンスタイムの変化

トランザクション数	移動無し	ランダム	CPUパワー	負荷状況	WB
1000	1.00	1.03	1.03	1.03	1.03
1500	1.00	1.09	1.09	1.08	1.08
2000	1.01	1.19	1.16	1.16	1.15
2500	1.03	1.33	1.25	1.24	1.25
3000	1.32	1.59	1.34	1.33	1.37
3500	2.56	2.31	1.44	1.42	1.48
4000	6.27	3.43	1.54	1.51	1.61
4500	11.81	7.26	5.11	4.32	3.90
5000	20.77	16.90	14.50	14.32	12.62
5500	32.55	29.80	28.30	27.43	25.23
6000	45.98	46.33	44.04	44.12	40.53

表 B. スループットの変化

トランザクション数	移動無し	ランダム	CPUパワー	負荷状況	WB
1000	1.98	2.02	2.03	2.02	2.02
2000	3.06	3.62	3.52	3.51	3.51
3000	4.15	5.93	5.46	5.45	5.62
4000	5.22	7.10	7.68	7.65	8.13
5000	6.23	7.51	7.78	7.79	9.20
6000	7.09	8.04	8.34	8.33	10.27

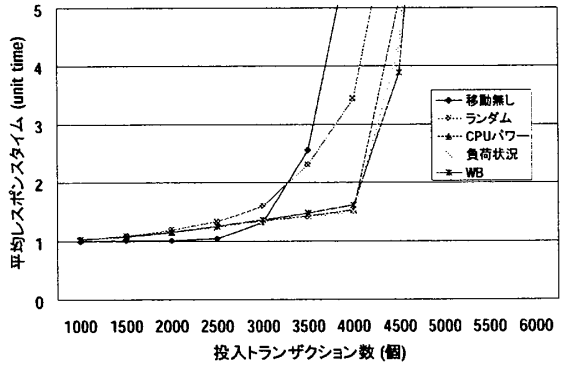

図 7. レスポンスタイムの変化

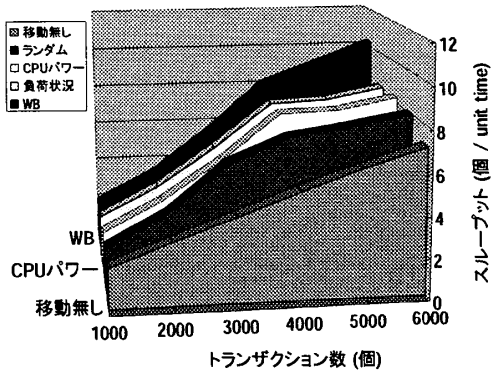

図 8. スループットの変化

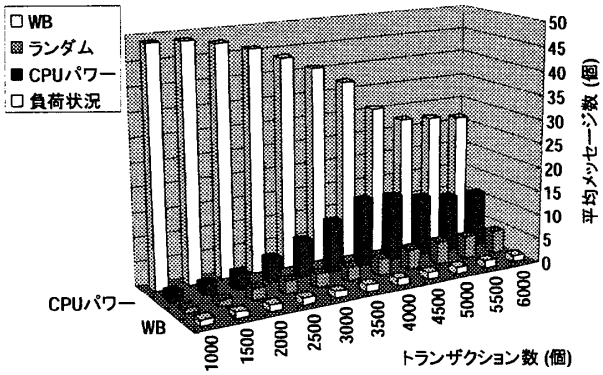

図 9. 通信メッセージ数の変化

- 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

3.5. 移動制御方式

本研究で検討を行ったそれぞれの移動制御方式を以下に示す[10].

(1) <ランダム制御>
移動先をランダムに選択する

(2) <CPU パワー制御>
CPU の性能の高い方を優先して選択する

(3) <負荷状況制御>
サイトの中で最も仕事がないサイトを選択する

(4) <ワーキングバッファ制御>
論理的な作業領域 (Working Buffer) を設定し，作業領域が最も空いているサイトを選択する

4. シミュレーション結果

以下にシミュレーションの結果を示す. サイトの数は 50, 各サイトの処理能力は正規分布に従い，最も処理能力の高いサイトは，最も低いサイトの 3 倍の処理能力である．各サイトへの到着も正規分布に従って到着する.

4.1. オブジェクト移動と制御方式

オブジェクト移動の有無と 4 種類の制御方式を使用したシミュレーションを行った．各サイトへ到着するトランザクションの数を変化させた場合の応答時間，スループットとの変化を表 A と表 B に示す．レスポンスタイムの変化を図 7 に，スループットの変化を図 8 に，通信メッセージ数の変化を図 9 にそれぞれ描画する.

応答時間に関しては，投入される要求が少ない場合は移動操作を行わない方が良い結果となっている. しかし，投入される要求の数が多くなってくると移動を行うほうが良い結果となる. 制御方式については CPU パワーを優先, 負荷状況を優先, ワーキングバッファを使用する方法が有効な結果を示している．しかしながら，負荷状況に基づいて移動先を決定する場合は，各サイトの負荷状況を動的に随時把握する必要がある．負荷状況を把握するには非常に多くのメッセージ交換が必要となる．それに対して，WB 制御では，ほとんど 1 回の通信回数で移動を行うことが可能となっている（図 9）. また，スループットに関しては，ワーキングバッファを使用する方式が有効であるから，ワーキングバッファを使用する方法が最も現実的かつ効果的な方法である.

4.2. Working Buffer サイズ

6 種類のバッファサイズでシミュレーションを行った，表 C にバッファサイズの条件を示す．応答時間の結果を表 D 及び，図 10 に示す．ベストケースとワーストケースを表 E にまとめた．投入される要求の数が 3000 を超えると <WB3>が最良の結果となる．反対に<WB6>が最悪の結果となった．<WB6>の特徴として，他の制御に比べて平均バッファサイズが最も小さくなっている.

表 F と図 11 に移動率の結果を示す．これは各サイトに到着した要求が，リモートサイトで処理される割合である．各制御方式のバッファの大きさが大きく反映された結果と

表 C. バッファサイズ

	min size	max size	average
WB1	10	35	21.7
WB2	5	30	16.7
WB3	10	18	13.56
WB4	5	15	12.52
WB5	10	13	8.56
WB6	5	10	7.52

表 D. レスポンスタイム

負荷	BEST	WORST
1000	WB4	WB5
2000	WB4	WB2
3000	WB6	WB2
4000	WB3	WB6
5000	WB3	WB6
6000	WB3	WB6
7000	WB3	WB6
8000	WB3	WB6

表 E. レスポンスタイム

トランザクション数	WB1	WB2	WB3	WB5	WB4	WB6
1000	1.03	1.03	1.03	1.03	1.03	1.03
2000	1.16	1.16	1.16	1.16	1.15	1.15
3000	1.37	1.38	1.37	1.37	1.37	1.37
4000	1.72	1.73	1.61	1.65	1.62	3.20
5000	14.26	13.59	12.62	15.23	12.83	22.73
6000	44.72	43.52	40.53	45.19	42.71	55.37
7000	81.02	81.19	76.86	82.75	79.75	94.91
8000	119.33	119.85	116.48	123.77	122.17	137.53

表 F. 移動率 (リモートサービス率)

トランザクション数	WB1	WB2	WB3	WB5	WB4	WB6
1000	0.03	0.03	0.03	0.03	0.03	0.03
2000	0.16	0.16	0.16	0.16	0.15	0.15
3000	0.35	0.35	0.37	0.37	0.37	0.36
4000	0.50	0.50	0.56	0.56	0.57	0.47
5000	0.54	0.52	0.47	0.35	0.47	0.35
6000	0.52	0.48	0.44	0.31	0.42	0.32
7000	0.51	0.46	0.42	0.27	0.39	0.29
8000	0.52	0.45	0.41	0.25	0.37	0.27

なった．バッファサイズを小さく設定すると移動率は低くなり，大きくすると高くなった．

5. おわりに

本研究では，オンラインシステムを対象とした大規模結合分散システムにおける，負荷分散について述べた．負荷分散を実現するミドルウェアプロトタイプを実装し，シミュレーションに必要な実際のデータ（転送時間など）を取得した．オブジェクト移動の指針となる状態マップを実際のデータから作成し，オブジェクト移動を取り入れた大規模疎結合システムのモデルを構築した．そして，実際に大規模システムを構築した場合の，システムの振舞いと各移動制御方式について，シミュレーションを行うことで評価した．シミュレーション結果から以下が観察された．

システムへの要求の到着が少ない場合は，移動を行うと移動のオーバーヘッドによりレスポンスタイムが多少悪化した．しかし，要求の数が増えるにつれて移動を行うほうが，レスポンスタイム，スループット共に向上した．スループットの向上は，余っている計算資源を有効に利用できたためである．各制御方式は，レスポンスタイム，スループット共に以下のような結果となった．

{WB > {負荷状況, CPU パワー} > ランダム}

WB 制御においては，バッファサイズの設定が重要となる．また，システムの状態ごとに最適なバッファサイズが存在する．システムの構成やシステムが処理するアプリケーションの特性に応じて，バッファサイズをチューニングする必要がある．

本研究では，大規模疎結合分散システム上において，他のサイトに負荷を分散させることにより，過負荷状態を防ぎ，安定的な性能を獲得する手法について評価した．本研究はデスクトップグリッドコンピューティングシステム[7]に加えて，オーバーレイネットワーク上の P2P アプリケーション[3]に適応可能である．

現在は，ディレクトリサーバを中心とした，集中制御により，負荷分散を行っている．今後の展開として，各サイトによる自律的な負荷分散制御を行った場合のシステムの振舞いについても研究していく．

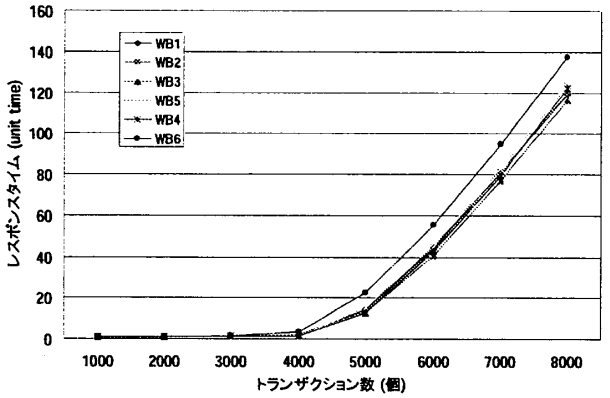

図 10. レスポンスタイム変化

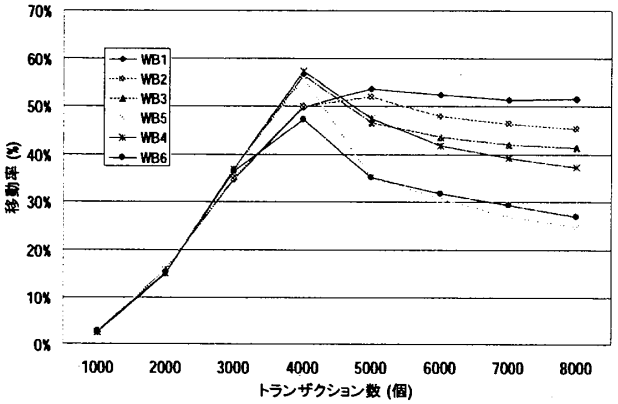

図 11. 移動率の変化

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] S.Venugopal, et al, A taxonomy of Data Grids for distributed data sharing, management, and processing, ACM Computing Surveys, Vol.38, No.3 (2006).

[3] 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).

[4] 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).

[5] 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第 57 回連合大会講演論文集, pp.240-241, 2006.

[6] 阪本憲司, 吉田誠:「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」, FIT2007 第 6 回情報科学フォーラム, pp155-158, 2007.

[7] R.Shah, et al, On the Design of Adaptive and Decentralized Load-Balancing Algorithms with Load Estimation for Computational Grid Environments, IEEE Trans. On Parallel and Distributed Systems, Vol.18, No.12, pp.1675-1685 (2007).

[8] S.Choi, et al, A Taxonomy of Desktop Grids and its Mapping to State-of-the-Art Systems, Technical Report, GRIDS-TR-2008-3 (2008).

[9] K.Sakamoto, M.Yoshida, "Design and Evaluation of Large Scale Loosely Coupled Cluster-based Distributed Systems", IFIP International Conference on Network and Parallel Computing Workshops, pp.572-577, 2007.

[10] M.Yoshida, K.Sakamoto, "Code Migration Control in Large Scale Loosely Coupled Distributed Systems", Proc. of the 4th International Conference on Mobile Technology, Applications and Systems, pp32-38, 2007.