岡山理科大学紀要 第46号 A pp. 29-38(2010)

分散デスクトップコンピューティングにおけるコードマイグレーションを利用したワークロード管理の設計とその制御

吉田　誠・小島　一峰*

岡山理科大学工学部情報工学科
*岡山理科大学大学院工学研究科修士課程情報工学専攻

(2010年9月17日受付、2010年11月9日受理)

1. はじめに

近年，PC／WSの低コスト化・高性能化，ブロードバンドネットワークの普及，Java／Eclipseなどのオープンな分散開発環境の整備が急速に進んでいる．一方，大規模で高速な計算，情報共有を必要とする需要や要求も急速に拡大し，多様化している．これらの技術や要求にともない，グリッドコンピューティング[1) 2)]，クラウドコンピューティング[15)]，およびP2P[3) 4)]などの分散システムが注目されている．分散システムでは，ネットワークに存在する各種資源(CPU，データ，等)を有効に利用し，効果的な処理を行う必要がある．そのためには，拡大化・多様化したいろいろなアプリケーションのワークロード特性を分析し，適切な負荷分散制御を実現する必要がある[5)]．

著者らは，分散コンピューティングシステムとしてデスクトップPC／WSに着目し，負荷分散制御のための実験モデルを開発した[11)]．そして，その実験モデルによる実データをもとに，数百サイト程度から構成される，分散デスクトップコンピューティング環境を，モデル化した．また，各種シミュレーションを実施し，各々のワークロードの制御方式を比較評価した[8) 9) 10) 12) 13) 14)]．

本論文では，各種ワークロード環境のもとでの，コードマイグレーションを利用した負荷分散制御方式の比較評価についてまとめる．そして，それらの評価結果をもとに，ワークロード制御のための設計スキーマを提案し，その設計スキーマに基づいたワークロード制御の実現方式について提案した．

2. 適用領域

2.1 損益分岐点

マイグレーションの有効性とその適用領域を把握するために，図1に示すミドルウェアプロトタイプシステムを実装した[11)]．そして，実装したミドルウェアを用いて，オブジェクトマイグレーションの時間的特性（オーバーヘッド）を測定し，その損益分岐点を観測した．ローカルサイトとリモートサイトで，オブジェクトを実行した時のレスポンスタイムを測定し，ローカルサイトでの実行とリモートサイトでの実行の利得が逆転する点を損益分岐点とした．損益分岐点は，測定するオブジェクトの負荷（AP 負荷率）と他アプリケーションの負荷（CPU 使用率）を変動させながら観測された．結果を図2(a)に示す．図中の右枠内の数値はAP負荷率を示している．Rはリモートを意味し，リモートで実行されたことを示している．図 2(a)の四角形のプロット点が損益分岐点である．点線で囲った部分は，ローカルサイトで処理をした方が良い領域である．

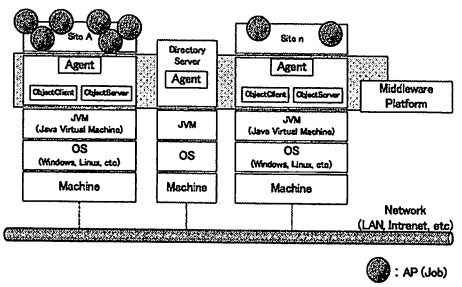


図1　システム構成図

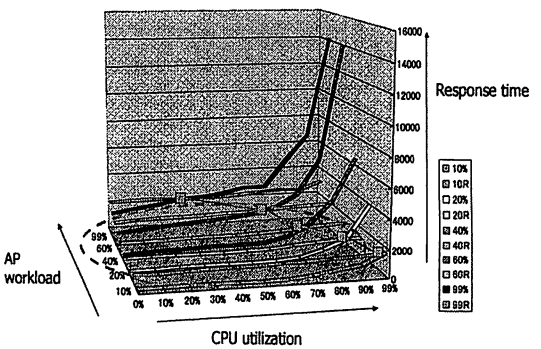


図 2(a)　損益分岐点

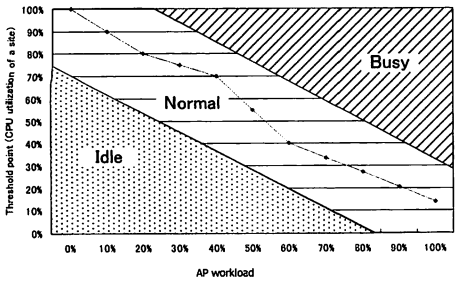


図2(b) マイグレーション状態テーブル

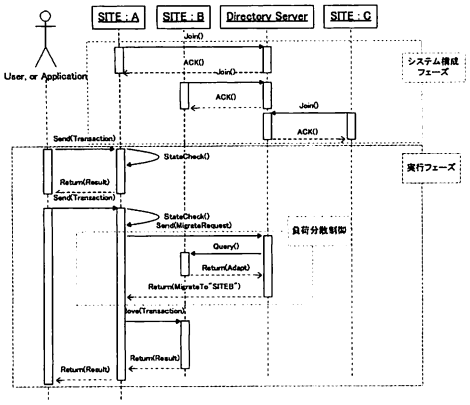


図3 集中制御方式のシーケンス図

2.2 マイグレーション状態テーブル

著者らは，図 2(a)から図 2(b)のマイグレーション状態テーブルを作成し，以降のシミュレーションにおいて転送決定ポリシーの判定に使用した．図 2 (b)の状態テーブルは，2 重閾値方策[7]を用いて， 3 つの状態に分類されている．CPU 使用率が高い状態を BUSY 状態，CPU 使用率が低いサイトを IDLE 状態，その中間を NORMAL 状態と定義している．図 2(b)の横軸はオブジェクトの CPU 負荷率，縦軸はサイトの CPU 使用率（当該計算以外に使用している CPU の割合）を示している．

BUSY 状態のサイトにトランザクションが到着した場合には，IDLE もしくは NORMAL 状態のサイトに当該トランザクションを転送し，リモートサイトで実行される．IDLE 又は NORMAL 状態のサイトにトランザクションが到着した場合には，当該トランザクションはローカルサイトで実行される．

3. 負荷分散制御方式

負荷分散を行う場合には，位置設定ポリシーに基づき負荷の受入が可能なサイトを見つける必要がある．また，複数のサイトが受入可能である場合は，それらのサイト群から最適なサイトを決定しなければならない．本章では，ディレクトリサーバ（情報サーバ）を用いる集中型制御方式と，各サイト間で合意を得て自律的に負荷分散を行う分散型制御方式について説明する．

3.1 集中型制御方式

集中型制御方式では，各サイトは位置決定情報に関わるすべてのサイトの情報を集中的に管理するディレクトリサーバを介して負荷分散を行う[8][10]．図3に集中型制御方式におけるシステムの動作を示す．システムは2つのフェーズに分けられる．1つはシステム構成フェーズであり，もう1つは実行フェーズである．システム構成フェーズにおいて，各サイトはディレクトリサーバに対してシステムへ参加することを宣言する．実行フェーズにおいては，各サイトに対してユーザもしくはアプリケーションからマイグレート要求が行われる．各サイトは，自身のマイグレーション状態テーブルを確認しながら動作し，自身の状態が悪化すると，サーバにマイグレーション要求を送信する．要求を受取ったサーバは，以下に示す各種方式にもとづいて，受入可能であるかを確認するメッセージを送信する．受入可能であった場合は，マイグレーション要球を送信したサイトに，受入可能なサイトのアドレスを返信する．受入可能でなかった場合，他に受入可能であると考えられるサイトが存在するならそのサイトに対して同様のメッセージを送信し，存在しないならマイグレーション要求を棄却し，受入サイトが存在しないことを通知する．以下に，集中型での受け入れサイトを決定する 4 つの負荷制御方式を示す．

3.1.1 RB 方式（Random Based）

RB方式は，ディレクトリが保持しているサイト情報の中から，受入可能なサイトをランダムに選択する制御方式である．

3.1.2 CB 方式（CPU-Power Based）

CB方式は，ディレクトリが保持する各サイトのCPU性能情報に基づき，性能の高いサイトを優先して選択する制御方式である．CPU性能についての情報は静的（static）な情報であり，サイトがシステムへ参加する時点でディレクトリに登録され，以降変更されることはない．

3.1.3 TB 方式（Traffic Based）

TB方式は，各サイトの負荷状況を確認して，負荷状況の軽いサイトを優先して選択する制御方式である．各サイトの負荷状況についての情報は，時々刻々と変化していく動的（dynamic）な情報である．各サイトにブロードキャスト通信を行い，それぞれのサイトの負荷状況を確認する必要がある．

3.1.4 WB 方式（Working-Buffer Based ）[12]

WB方式は，各サイトに予め設定されたWB値（他サイ

```
SITE i (Si)
  SENDER Process
    Send (MigrateRequest) to all_member;
    Receive(Ack_with_Timestamp Txo) from members;
      /* Txo stands for the timestamp T for object o on site x/
    Select the site_X that has the earliest timestamp Txo from Received messages;
    Send (Move_Transaction with Txo) to the members;
    Receive(Result);

  RECEIVER Process
    if  Receive(MigrateRequest) then
        Statecheck(); /* Check Migration State Table/
        if state=idle then
                Reserve&Enque_Request with timestamp Tio;
                        /*Tio stands for the Timestamp T for object on site i/
                Return(Ack_with_TimestampTio);
    if Receive(Move_Transaction) then /*Compare the timestamps/
        if (It has sent the Ack with Timestamp Tio) & ( Tio= Txo) Then
                        Migrate_Code & Execute_Code;
                        Return (Result);
        Else            Cancel&Deque the request that has the timestamp Tio;
```

図4　分散制御アルゴリズム

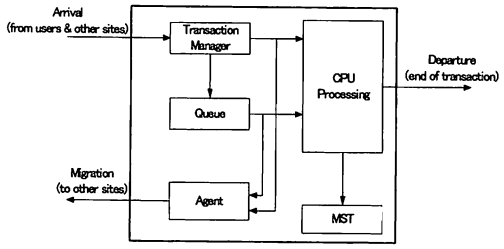


図5　各サイトのシミュレーションモデル

トから利用可能なマイグレーション数）の最も大きいサイトを優先して選択する方式である．WB値は，各サイトが過去の履歴情報や自サイトの状況に応じて各々設定できる論理的な値である．WB値は，自サイト及び他サイトからマイグレートされるトランザクションを処理することにより動的に変動する．各サイトは，このWB値が正でない場合は他のサイトからの要求を受取ることはできない．また，ユーザから直接受取った要求も処理できない．WB値を適切に設定することで，サイトの能力に応じた効果的な数の要求を処理することができる[12]．

3.2 分散型制御方式（Distributed Control）[9]

DC方式では，それぞれのサイトが自律的に自身の状態を確認しつつ，他のサイトとメッセージを交換し，負荷分散制御を行う．図4に，各サイトの動作を示す負荷分散アルゴリズムを示す．基本的な動作としては，送信されてくる各種のメッセージに応じた処理を各サイトで自律的に行っている．本研究で対象とするシステムの適用対象としては，メッシュ型のネットワークを想定している．また，サイト間の通信としてブロードキャスト（Broadcast）通信を採用している．状態が悪化した，もしくは悪化し始めたサイトは，システムを構成するすべてのサイトに対して，ジョブの受入が可能であるかを問い合わせる．

表 1　シミュレーションパラメータ

パラメータ	値	
サイト数	50，500	システムを構成するサイトの数
到着分布	正規分布 一様分布 ポアソン分布	トランザクションの到着分布パターン
シミュレーション時間	10000（unit time）	シミュレーションでシミュレートする時間
投入トランザクション数	1000 ～ 15000	システムへ投入される要求の数
サイトの処理能力	平均:2（transaction / unit time）	各サイトが1つのトランザクションを処理する能力，最も性能の良いサイトは悪いサイトの3倍
通信遅延	1（unit time）	メッセージ通信にかかる時間
マイグレーション遅延	10（unit time）	要求のマイグレーションに必要な時間

表 2　観察データ

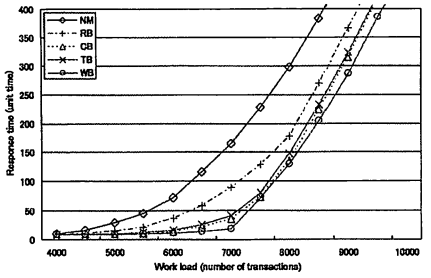
観測データ	
レスポンスタイム	トランザクションが到着してから処理されるまでの時間
スループット	サイトが単位時間当たりに処理したトランザクション数
マイグレーション率	到着トランザクションが，到着サイト以外で処理された割合
通信回数	各サイトの送受信メッセージ数
ディレクトリの通信（集中制御のみ）	集中制御を用いる場合に，ディレクトリサーバが行う通信の回数
CPU使用率	各サイトのプロセッサの動作状況
マイグレーション受入数	各サイトが受入れた要求の数
待ち行列長	プロセッサが空くのを待っている要求の数と，マイグレーションするのを待っている要求の数の合計

応答がない場合は要求を受入可能なサイトは存在しない．複数のサイトから応答がある場合は，タイムスタンプが最も若い応答メッセージが採用される．

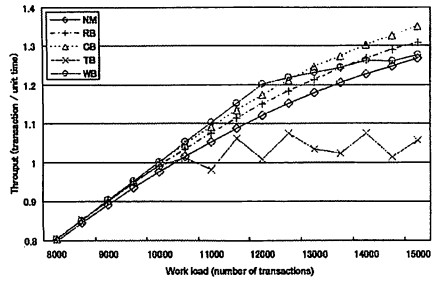
4. シミュレーション環境

4.1 シミュレーションモデル

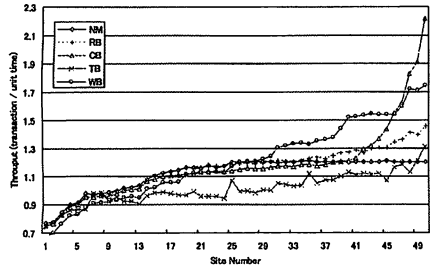
シミュレーション環境は以下のように設定されている．分散システムは，ネットワークで相互接続されたせいぜい数百程度の処理能力の異なるコンピュータにより構成される．各サイトのシミュレーションモデルを図5に示す．ユーザから投入されたトランザクションは，CPUが空いていれば，サービスを受ける．空いていない場合は，待ち行列でCPUが空くのを待つ．また，CPUが空くのを待たないで，マイグレーションを行う場合もある．マイグレーションを行う場合，各サイトは自身のマイグレーション状態テーブル(MST)を参照し，マイグレーションが必要と判断すると，ディレクトリサーバや他のサイトと通信を行い，受入可能なサイトを見つける．



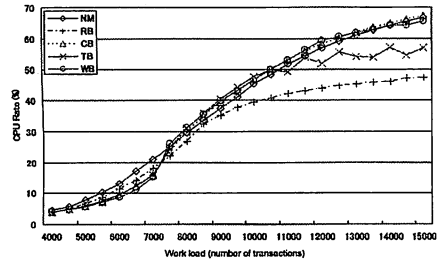
(a) レスポンスタイム



(b) スループット



(c) 各サイトのスループット



(d) CPU使用率

図6 集中型制御方式

4.2 シミュレーションパラメータ

本実験で設定するシミュレーションパラメータを表1に示す．ユニット値は，プロトタイプで実測した値に基づきモデル化された値となっている．

4.3 観察データ

表2にシミュレーションの観察データを示す．

5. シミュレーション結果

各種集中型制御方式と分散型制御方式の特性を，トランザクション数，到着分布パターン，サイト数の規模，ネットワーク遅延などを変化させて観察した．

5.1 集中型制御方式

本実験では，サイト数を50とし，到着する要求トランザクションを正規分布に従って到着させ，各サイトに投入する要求の数を4000から15000まで変化させて各種特性を観測した．

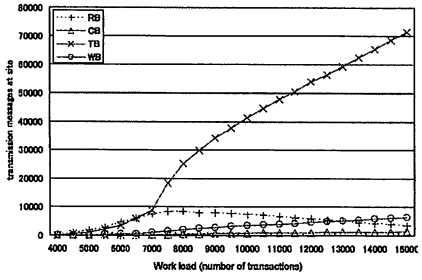
5.1.1 レスポンスタイムとスループット

図6（a）にレスポンスタイム，図6（b）にスループット，図6（c）に投入トランザクション数が12000の場合の各サイトの平均スループット，図6（d）に平均CPU使用率の結果を，それぞれ示す．

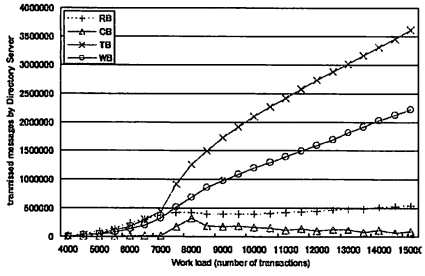
NM方式（マイグレーションを行わない方式であり，Non-Migrationの省略形である）では，レスポンスタイムが投入トランザクション数4500から急激に悪化し始める．次に，トランザクション数が6000でRB方式，7000でTB方式とCB方式，最後に7500でWB方式のレスポンスタイムが悪くなるという結果が確認できる．トランザクション数が7000の場合，NM方式のレスポンスタイムが，165.37（unit time）に対して，WB方式では18.67（unit time）となり，WB方式の方がNM方式よりも8.85倍性能が良い．スループットについては，投入トランザクション数が12000の場合，NM方式では1.12（transaction / unit time）であり，WB方式では1.20となり，WS方式が7%良くなっている． 図6（c）はCPU能力の高い順に各サイトを並びかえて，各サイトのスループットをプロットした図である．NM方式のスループットは一定値に収束しているが，CB方式ではCPUの高いサイトが多ければ多いほどスループットは良くなることがわかる．RB方式とWB方式では，CPU能力に比例してスループットが良くなっている．図6（d）に示すCPU使用率は，ほぼ一様にある時点（トランザクション数7000）から急激に高くなっている．NM方式の場合，レスポンスタイムとスループットは良くないのに対して，CPU使用率は高くなっている．これは効果的にリソースを使用できていないことを意味している．RB方式も同様に，トランザクションを適切にマイグレーションしていない結果を示しているといえる．

5.1.2 通信メッセージ数

図7に，集中型制御でマイグレーションを行った場合の，通信メッセージ数を示す．図7（a）に各サイトが送信したメッセージ数の平均とトランザクション数の関係，図7（b）にディレクトリサーバが送信したメッセージの総数を示す．

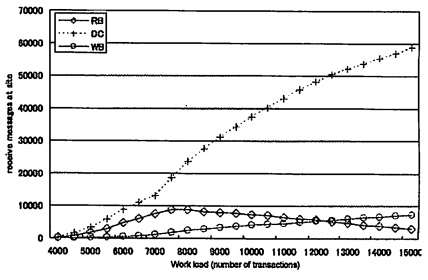


(a) 各サイトの通信メッセージ数

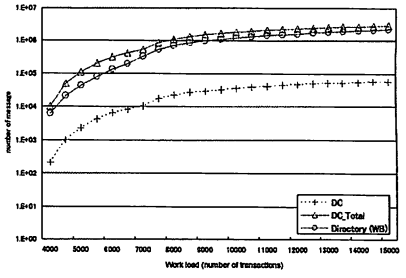


(b) 通信メッセージ数（ディレクトリサーバ）

図7　集中制御における通信メッセージ数

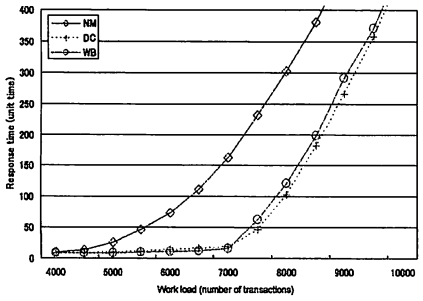


(d) 転送メッセージ数

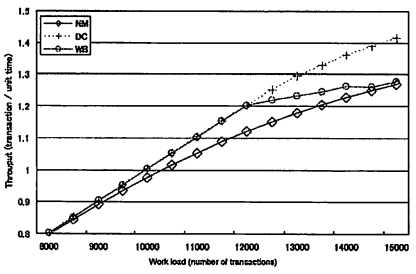


(e) 転送メッセージ数比較

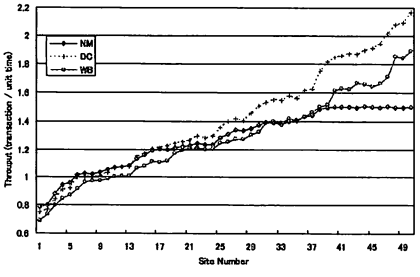
図8　分散制御結果



(a) レスポンスタイム



(b) スループット



(c) 各サイトのスループット

TB方式が最も多くメッセージを送受信し，かつその数は急激に増加していることが確認できる．到着トランザクション数が15000の場合は，TB方式の送信メッセージは71330で，CB方式では1386であり，CB方式とTB方式ではおよそ50倍のメッセージ数の相違が観察される．集中型制御方式においては，ディレクトリサーバを介して負荷分散を行うため，各サイトからのメッセージが集中し，ディレクトリサーバが最大のボトルネックとなっていることが観察される．

5.2 分散型制御方式

集中型制御方式でレスポンスタイムが最良であるWB方式，NM方式，と分散型制御方式であるDC方式を比較した．DC方式では，各サイトは，3.2節で示した分散アルゴリズムに従って，負荷分散を行っている．

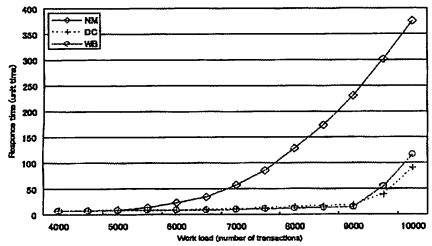
図8（a）に平均のレスポンスタイム，図8（b）にスループット，図8（c）に各サイトのスループット，図8（d）に各サイトの通信メッセージ数，図8（e）にWB方式でのディレクトリの送信メッセージとDC方式でのサイトの送信メッセージの比較を，それぞれ示す．

レスポンスタイムについては，WB方式よりもDC方式が良好な結果を示した．投入トランザクションが7500の場合，WB方式では62.90（unit time），DC方式では46.95（unit time）であり，DC方式の方がWB方式より34%良くなっている．スループットについても，投入トランザクション数が15000の場合，WB方式では1.277（transaction / unit time），DC方式では1.415（transaction / unit time）でありDC方式の方がWB

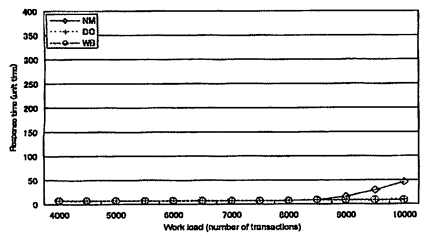
方式より11%良くなっている．また，図8（c）に示した各サイトのスループットを観察すると，WB方式では性能の良くないサイトのスループットはNM方式と較べて低下しているが，DC方式では低下しておらず，常にNM方式より良くなっている． DC方式は，それぞれのサイトが自身の処理能力を最大限に使用しているといえる．通信メッセージについては，送受信ともに，DC方式が他方式よりも最も多いという結果になっている．トランザクションが少ない場合は，性能の悪いサイトだけがブロードキャストによる問い合わせを行うのに対して，トランザクションが多くなると，各サイトの状態が悪化し，それぞれのサイトが他のすべてのサイトに対して，受入可能であるかを問い合わせているのが原因である．しかしながら，図8（e）に示すように，DC方式の各サイトの送受信メッセージは集中型に較べると十分に少ないといえる．各サイトのメッセージを合計した数DC-Totalは，ディレクトリサーバのメッセージ数と大きな差はない．

5.3 到着分布パターンの相違による変化

トランザクションの到着分布パターンを変化させて実験を行った．



(a) レスポンスタイム（ポアソン分布）



(b)レスポンスタイム（一様分布）

図9 到着分布による変化

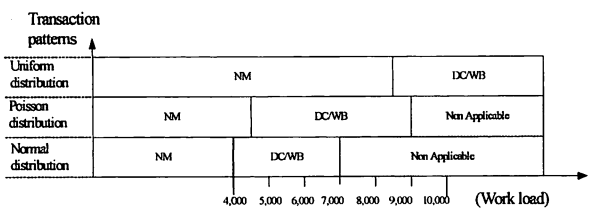
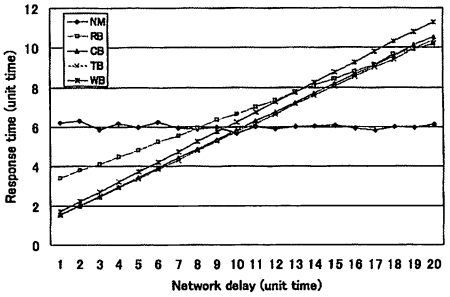
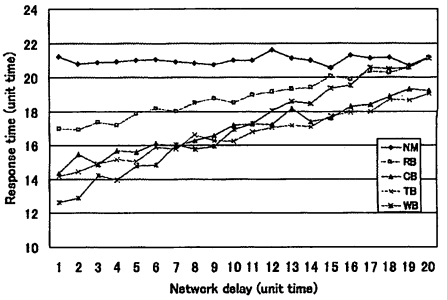


図10 制御方式と適用領域の関係



(a) レスポンスタイム（Tr.数4000）



(b) レスポンスタイム（Tr.数5000）

図11 レスポンスタイム比較

到着分布パターンとしては，正規分布（Z平均0，分散1），ポアソン分布（Z平均-1.5，分散1.5），一様分布（Z平均0，分散100）の3種類を設定した．正規分布についての結果は，5.1節の結果を用いる．

平均レスポンスタイムの結果を図9に示す．図9（a）はポアソン分布， 図9（b）は一様分布に従っている．これらの結果から，到着分布の相違によるレスポンスタイムとして，以下の結果が確認できる．

（良）“一様分布”>”ポアソン分布”>”正規分布”（悪）

一様分布に従う到着が最も良いレスポンスを示し，正規分布に従う到着が最も悪いレスポンスとなることを示している．一様分布に従う到着を対象とするシステムでは，トランザクション数の増加に対しても，各サイトが余裕を持ってトランザクションを処理できていることが，観察される．

これらのレスポンスタイムに関するシミュレーションから，図10に示す制御方式とアプリケーション適用領域（ドメイン）の関係図を得ることができる．正規分布では，投入トランザクション数が4000から7000の間で，DC方式（分散型）またはWB方式（集中型）を使用すると効率が良い．トランザクション数が7000の時には，NM方式と比べてDC方式は9.2倍，WB方式は8.8倍，応答時間が良くなっている。トランザクション数が7000以上では，急激に応答時間が悪化するため，マイ

グレーション不適用の領域となっている．同様に，ポアソン分布では，トランザクション負荷が4500からトランザクション負荷が9000の範囲がマイグレーションの適用領域となっている．一方，一様分布を見ると，8500以上継続してマイグレーションの有効な領域となっている．

図10から，トランザクションの到着分布パターン及び投入トランザクション数の大きさにより，適用領域が変化していることが観察できる．傾向としては投入トランザクション数が大きくなるに従い，正規分布よりポアソン分布と一様分布の方がレスポンス効率が良く，また投入トランザクション数も増加し，マイグレーションが有効に働いていることが観察される．

5.4 サイト数・伝送遅延の相違による変化

システムを構成するサイトの数を変化させて実験を行った．サイト数を50と500に設定した．到着分布を正規分布とした．トランザクション数をマイグレーション有効領域である4000から8000まで変化させた．レスポンスタイムとスループットの結果については，サイト数を変化させても，大きな変化はないことが観察された[8]．

また，各サイト間を結ぶネットワークの伝送時間を変化させて，レスポンスタイムを測定した．トランザクションの負荷状況，および各トランザクションの処理時間と伝送時間の比率を1:1から1:20まで変化させて，レスポンスタイムを観測した．トランザクション数が4000の場合と5000の場合の，伝送遅延の変化によるレスポンスタイムの変化を，図11(a)，図11(b)に示す．NM方式と他方式との交点が，トランザクション数が増えるに従い右(ネットワーク遅延の大きい方)にシフトしている．これはトランザクション数が多ければ多いほど，マイグレーションした方が良い領域が拡大していることを意味している．トランザクション数が4000であれば，伝送遅延時間が計算時間の10倍程度まで，5000であれば20倍程度まで，マイグレーションが有効であることが確認できる．一方，ネットワークの伝送時間が大きくなると，各種マイグレーションアルゴリズムによる差異は少なくなり，直線的にレスポンスタイムは悪化している．レスポンスタイムは，以下の近似式で表わされる．

＜ﾚｽﾎﾟﾝｽﾀｲﾑ＞≒＜相対ﾈｯﾄﾜｰｸ遅延＞／２　＋　Ｃ

相対ネットワーク遅延は，ネットワーク伝送時間をトランザクションの処理時間で割った相対比であり，Cはトランザクション負荷により決定される値である．上記式により，相対ネットワーク遅延とCが与えられれば，予想されるレスポンスタイムが算出できる．また，当該予想値とマイグレーションを行わなかった時のレスポンスタイムを比較することにより，マイグレーションの適用領域を決定することが可能となる．

5.5 シミュレーション結果

以下に，本研究のシミュレーション結果をまとめる．

- 全ての方式において，トランザクションが増加すると，レスポンスが急速に悪化する損益分岐点が存在する．トランザクション数が損益分岐点を超えると，マイグレーションなど何らかの対応を行い負荷を減らす必要がある．
- 集中型制御方式を使うマイグレーションについては，次の結果が得られた．到着するトランザクションの数が少ない場合は，マイグレーションを行う必要はないが，到着するトランザクション数が多くなるとマイグレーションを行うと，レスポンスタイムとスループットはともに大幅に向上する．各種の負荷分散制御方式を，レスポンスタイムで比較すると以下のようになった．

  ”（良）WB＞CB＞TB＞RB＞NM（悪）”
  WB: Working Buffer based
  CB: CPU Power based.
  RB: Random based.
  NM: No Migration.

  レスポンスタイムが急速に悪化し始める直後について較べると，WB方式は，NM方式(マイグレーションを行わない方式)に較べてレスポンスタイムが8.85倍良くなることが観察され，その有効性が確認できた．スループットについては，WB方式よりもCB方式の方が良いが，全体的に大きな差異は見られなかった．CPU使用率については，NM方式，CB方式およびWB方式が一様に高く，他の方法(RB)より20%程度高くなっている．この結果は，レスポンスタイムとスループットから見て，NM方式とは対照的に，CB方式とWB方式は非常に効果的にCPUを使用しているといえる．
- 集中型制御方式において，最も良好なレスポンスタイムを示したWB方式では，他サイトからの利用可能なマイグレーション数を決定する値（WB値）を設定する必要がある．トランザクション数に応じた適切な値（WB値は大き過ぎても小さ過ぎても良くなく，各サイトのレスポンスタイムが一様となるような値）を設定することが重要であることを確認している．[12]

- 分散型制御方式は，各種集中型制御方式よりも良好なレスポンスタイムとスループットを示した．レスポンスタイムについては，分散型制御方式は，集中型において最も良好なWB方式よりも34%の向上することが確認された．スループットについては，集中型で最も良かったCB方式に対して，分散型であるDCは11%の向上が観察された．この規模では，最良の制御方式であるといえる．
- 同じ量のトランザクションを3つの到着分布(一様分布，ポアソン分布，正規分布)を想定してシミュレーションを行った．レスポンスタイムを観測すると，以下の結果が得られた．

”（良）一様分布＞ポアソン分布＞正規分布（悪）”

同じ量の計算を一様分布にすると，正規分布に較べて41.8倍レスポンスタイムが良くなることが確認された．到着分布については，システムを構築する時点で，システムが取り扱うアプリケーションがどの様な到着分布であるのかを確認する必要がある．一般的には”一様分布”になるようにトランザクションを分散させると非常に有効であることが確認された．

- 数十サイトから数百サイトまでサイト数を変化させてシミュレーションを行った．少なくとも数百サイト程度の規模においては，本モデルはスケーラブルであり，本シミュレーション結果が有効であることが確認された．
- 計算時間と伝送時間の比率を変化させてシミュレーションを行った．トランザクション数が多くなると，それに伴いマイグレーションした方が良い領域が，拡大されることが観察された．伝送時間が計算時間の20倍以内程度であれば，マイグレーションによる負荷の分散は極めて有効であることが確認された．

6. ワークロード管理とその制御

6.1 ワークロード設計スキーマ

図12に，負荷分散に対応するための設計スキーマを示す．まず，トランザクション数，トランザクションの到着分布パターン，ネットワーク遅延およびマイグレーション状態テーブルなどのアプリケーション環境データを観察する．そして，以下に記述する基本ポリシーのもと，負荷分散設計を行い，DIST言語を使用して設計スキーマを実装している．DIST言語は，負荷分散制御を実現するために設計した分散処理言語である．[14] 図13に，当該言語のBNF構文を示しておく．

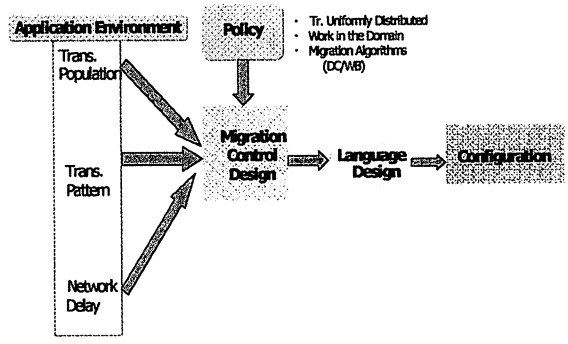


図12 設計スキーマ

```
DIST::= #DIST {ALL | <domain names> | <direct address> | NULL}
<domain names>::= <domain name> | <domain name>,<domain names>
<domain name>::= <direct address> | <direct address>,<domain name>
<direct address>::= <IP address> | <symbolic address>
<symbolic address>::= <IP address> | <IP address>, <symbolic address>
```

図13 DIST言語のBNF記述

基本ポリシーは，シミュレーション結果から導出された，以下の事実を基準としている．

- 各サイトはMST（マイグレーション状態テーブル）を持つ
- トランザクションパターンは，一様分布に従うのが良い
- 処置時間とネットワーク遅延の関係からマイグレーション可能な適用領域を得ることができる
- MSTは，転送ポリシーを決定する．位置決定ポリシーとしては，集中制御の場合にはWB方式を，分散制御の場合にはDC方式を，採用した方が良い

6.2 ワークロード管理の設計と制御

前節の設計スキーマに従った，ワークロード管理の設計とその制御方法について，具体的に示す．[13]

（1）まず，アプリケーション環境として，以下の要素データを把握する．
- ・各トランザクションの負荷の平均
- ・トランザクションパターン

（2）マイグレーション状態テーブルを設定する．

（3）処理時間とネットワーク遅延時間の関係から，マイグレーション適用領域を算出する．次に，各サイトのトランザクションパターンおよびトランザクション数（ワークロード）から，マイグレーション適用領域を算出する．両方の制約を満足するアプリケーション環境だけが，マイグレーションの適用領域となる．もし，当該アプリケーション環境が適用領域外であれば，マイグレーションは行われない(上限環境の設定)．

（4）適用範囲内であれば，1ドメイン内のトランザクション負荷が，それぞれのサイトにおいて一様になるように負荷を分散させる．また，そうなるように各サイトとドメインの関係を設計する．
（5）上記設計関係をDIST言語上に反映させる．
（6）ドメインの最大数が数百サイト程度で強力なサーバを持たないのであれば，マイグレーション制御方式としてDC方式を選択する．集中型を採用するのであれば，WB方式を選択する．
（7）分散デスクトップグリッド環境を設定し，プラットフォームを起動する．

7. おわりに

本論文では，分散デスクトップコンピューティングを対象とした，コードマイグレーションを用いた負荷分散制御方式の評価と設計について記述した．コードマイグレーションを行うためのミドルウェアのプロトタイプをデスクトップコンピュータを使用して実装した．本プロトタイプの開発により，一般のデスクトップコンピュータを使用してのマイグレーションに必要な時間や転送決定ポリシーの判断基準となるマイグレーション状態テーブルなどの特性を明らかにした．更に，本マイグレーション状態テーブルを使用して，位置決定ポリシーに基づく各種制御方式をシミュレーションで評価した．そして，当該シミュレーション結果をもとに，ワークロード制御のための基本ポリシー，設計スキーマ，およびトランザクションのワークロード制御のための設計方法を提案した．

本分散コンピューティンググリッドを1つのクラスタと考え，更に大規模な大規模コンピューティンググリッドを考えることも可能である．年々急速にパワーアップするコンピューティング環境ではあるが，今後はグリッドコンピューティング環境として，デスクトップコンピュータにだけでなく，携帯電話、ノートパソコンなども加わる可能性がある．本シュミュレーション結果は，これら将来環境に適応可能であり，効果的なパフォーマンスを実現できると考える．ただし，分散デスクトップコンピューティングの実用化のためには，性能に加えて，耐久性やセキュリティなども重要な要素である．今後の課題としては，耐久性など信頼性への対応，セキュリティ保障機の設定および超大規模なネットワークへの対応などが挙げられる．

参考文献

1) Venugopal, S., Buyya, R. and Ramamohanarao, K.: A Taxonomy of Data Grids for Distributed Data Sharing, Management, and Processing, *ACM Computing Surveys* (2006)
2) Choi, S., Buyya, E., Kim, H., et al.: A Taxonomy of Desktop Grids and its Mapping to State-of-the-Art Systems, *Technical Report, GRIDS-TR-2008-3* (2008).
3) Theotokis, S.A. and Spinellis, D.: A survey of peer-to-peer content distribution technologies, *ACM Computing Surveys*, Vol.36, No.4, pp.335-371 (2004).
4) Milojicid, S, D., Vana, K., Rajan, L. et al.: Peer-to-Peer Computing, Technical Report, HPL-2002-57 HP, Laboratories Palo Alto (2002).
5) Coulouris, G., dollirnore, J. and Kindberg, T.: Distributed Systems, Coulouris, G.: *Distributed Systems: Concepts and Design*, Addison-Wesley (2005).
6) Shah, R., Veeravalli, B. and Misra, M.: 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).
7) Santos, P, P, L.: Load Distribution: A Survey, Technical Report UM/DI/TR/96/03 (1996).
8) Yoshida, M and Sakamoto, K.: Performance Comparison of Load Balancing Algorithms through Code Migration in Distributed Desktop Computing Grids, *Proc. of the 3rd IEEE Asia Pacific Services Computing Conference*, pp.781-788 (2008).
9) Yoshida, M. and Sakamoto,K: Performance Comparison of Decentralized Workload Control through Code Migration in Distributed Desktop Computing Grids, *The 5th IEEE International Symposium on Embedded Computing* , pp.356-363 (2008).
10) Sakamoto, K. and Yoshida, M.: Design and Evaluation of Large Scale Loosely Coupled Cluster-Based Distributed Systems, *IFIP International Conference on Network and Parallel Computing Workshop*, pp.572-577 (2007).
11) 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プラットフォームの開発と評価，平成18年度　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).
12) 阪本憲司，吉田誠：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，FIT2008 第7回情報科学フォーラム(FIT2008)，第4分冊 pp.15-18 (2008).
13) Yoshida,M. and Kojima.K.: Design Methodologies of Workload Management through Code Migration in Distributed Desktop Computing Grids, *The 10th International Conference on Algorithms and Architectures for Parallel Processing, LNCS 6082*, pp.100-111(2010).
14) 小島一峰，寺岡正樹，吉田誠：コードマイグレーションを利用したワークロード管理の設計，2010年電気情報通信学会総合大会，pp.231(2010).
15) Buyya,R. Ranjan,R. and Calheirons,R,N: InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services, T*he 10th International Conference on Algorithms and Architectures for Parallel Processing, LNCS 6081*, pp.13-31(2010).

Design Methodologies of Workload Management Through Code Migration in Distributed

Makoto Yoshida, Kazumine Kojima*

Department of Information & Computer Engineering, Faculty of Engineering,
Okayama University of Science,
**Graduate School of Engineering,*
Okayama University of Science,
1-1 Ridai-cho, Kita-ku, Okayama 700-0005, Japan

(Received September 17, 2010; accepted November 9, 2010)

This paper describes the design methodologies of workload management focusing on distributed desktop computing grids. Based on the prototype experiment, several simulations were performed; several centralized and decentralized algorithms for location policy were examined, and the design methodologies for distributed desktop computing grids are derived from the simulation results. The methodologies for domains, language and control algorithms for computing grids are described. The language for distributed desktop computing is designed to accomplish the design methodologies.

Keywords: Grid computing; Migration; Simulation; Performance Evaluation.