マルチメディア通信と分散処理 91－12
（1999. 1. 22）

分散データベースにおける通信量を考慮した動的データ配置法

軍場 大輔　　多田 知正　　樋口 昌宏　　藤井 護

大阪大学 基礎工学部 情報科学科

分散データベースシステムにおける通信量を削減するためには個々のデータをどのサイトに配置するかということが重要となる．しかし各サイトで生じるデータ要求を事前に知ることは難しく，それぞれのサイトで頻繁にアクセスされるデータは時間とともに変化するため，静的に最適なデータ配置を決定することは困難である．そこで本稿では動的なデータ配置を行うことにより通信量を削減する分散データベースシステムを提案した．この分散データベースシステムの有効性を確認するために，シミュレーションにより従来の分散データベースシステムと通信量の比較を行い，有効性を確認した．またトランザクションのアボートにより生じる通信量の増加を防ぐために，多版並行性制御を導入することについて検討した．

Dynamic Data Allocation Method considering Communication Cost in Distributed Database Systems

Daisuke Kusaba　Harumasa Tada　Masahiro Higuchi　Mamoru Fujii

Depertment of Information and Computer Science, Faculty of Engineering Science
Osaka University

In distributed database systems, data allocation has a great influence on the communication cost. However, it is hard to know future data accesses in each site. Moreover, frequently accessed data in each site may change as time goes by. Therefore, it is difficult to determine the optimal data allocation statically. In this paper, we propose the dynamic data allocation in distributed databasa systems to suppress the communication cost. In order to evaluate usefulness of the proposed method, we compared the communication cost of it with that of an usual caching method through simulations. In addition, we examined how to introduce the multiversion concurrency control to suppress the communication cost by avoiding abortion of transactions.

1. まえがき

分散データベースシステムの主な特長として記憶容量や計算負荷の分散，障害時の可用性 (availability) の向上といったものが挙げられる．しかし最近では大容量のハードディスクや高速な CPU の普及により，記憶容量や計算負荷の分散を目的とした分散データベースシステムの意義は薄れつつある．一方で分散データベースには必要なデータを近くに配置することにより，ネットワーク負荷を削減できるという特長がある．近年広帯域ネットワークの研究が進められているが，その実現は大規模なインフラストラクチャの整備を必要とするため，現実のものとなるにはまだかなりの時間を要すると思われる．そこで本研究では分散データベースにおける通信量に着目し，これをできるだけ削減することを考えた．

分散データベースシステムにおける通信量を削減するためには個々のデータをどのサイトに配置するかということが重要となる．それぞれのデータはそれに最も頻繁にアクセスするサイトに置くのが望ましく，あらかじめ用意されたオペレーション系列から数学的手法を用いて配置を決定する手法の研究なども行われているが [1]，サイトのデータへのアクセスの傾向を事前に知ることは困難であり，またそれぞれのサイトで頻繁にアクセスされるデータは時間とともに変化することがある．これらの理由により最適なデータ配置を静的に決定することは非常に難しい．そこで本研究では動的なデータの配置を行う分散データベースを提案する．

従来，分散データベースシステムにおいて通信量を削減するための手法としては他のサイトから読み込んだデータをキャッシュに格納するという手法が一般に用いられている．ただしキャッシュを用いる方法はデータ読み出し時

の通信量の削減には有効であるが，書き込みの際には元のデータを書き換えなければならず，そのためにデータの通信が常に伴う．本研究で提案する分散データベースシステムではデータの配置を固定しないために，常に局所的なデータの更新が可能となり，書き込み時の通信量の減少が期待される．このことを確かめるため，シミュレーションによりキャッシュを用いる分散データベースシステムと本研究で提案する分散データベースシステムの通信量の比較を行った．

以降 2節で対象とする分散データベースシステムのモデルを説明し，3節で動的データ配置の基本的なアイデアとその実現のために必要な位置管理法について説明する．また従来のキャッシュを利用した分散データベースシステムと提案する分散データベースシステムの通信量比較を定性的に行う．4節では提案する分散データベースシステムについて，シミュレーションを行い，その結果に対する考察を行う．5節で並行性制御について述べ，6節でまとめを行う．

2. 準備

2.1 分散データベースシステム

本研究ではデータベースをデータの集合として定義する．また個々のデータの意味は考えない．データベースシステムはこのデータベースを操作する命令を持つハードウェアおよびソフトウェアモジュールの集合である．

分散データベースシステムとは複数のデータベースシステムが通信リンクで結合されたものであり，それぞれのデータベースシステムをサイトという．

2.2 トランザクションおよび並行性制御

トランザクションとはデータベースにアクセスするプログラムの実行である．トランザクションは読み出し操作と書き込み操作によってのみデータにアクセスする．データ x に対する読み出し操作を $read[x]$，書き込み操作を $write[x]$ で表す．$op[x]$ は $read[x]$ か $write[x]$ のどちらかを表す．

トランザクションが終了するときは正常終了 (コミット)，異常終了 (アボート) のどちらかの状態をとり，前者ではトランザクション中のデータ操作がすべて確定したものとしてデータベースに反映され，後者ではすべての操作内容を取り消さなければならない．

異なるトランザクション T_i, T_j 中の操作 op_i, op_j が同じデータに対する操作で，少なくとも一方が書き込みである場合，op_i と op_j は衝突するという．

複数のトランザクションを並行に実行する時は逐次化可能性が満たされていなければならない．トランザクションの並行実行 E に対して，ある逐次的実行 E_s が存在し，E 中の全ての衝突する操作の対 (op_i, op_j) について，op_i と op_j の実行順序が E と E_s において同じであるとき，E は逐次化可能であるという．トランザクションを並行に実行する場合，逐次化可能性を保証するために並行性制御を行う必要がある．並行性制御法としては 2 相ロッキング方式 (2PL)，時刻印順方式 (TO)[2] の 2 つが良く知られている．2PL ではトランザクションはデータにアクセスする前にそのデータをロックする．必要なデータが既に他のトランザクションにロックされているときは，ロックが解放されるまで待たなければならない．一度ロックを解放したトランザクションはそれ以上ロックすることができない．TO では各トランザクションに一意な時刻印を与え，オペレーションが衝突した際，それらは時刻印の順に実行される．したがって得られる並行実行は，その時刻印の順にトランザクションを逐次実行する場合と等価となる．

2.3 本研究における仮定

本研究では分散データベースシステムを構成するサイト数は一定で比較的小さく，メッセージの放送が可能であるとする．また通信リンク，サイトの故障については考えない．分散データベースシステム上に存在するデータ数は一定とする．以降データ x が存在するサイトを x の所持サイトと呼ぶ．

通信リンク上を流れるのはデータそのものと分散データベースの制御に用いられるメッセージの 2 種類である．制御メッセージのサイズはデータのサイズに比べてかなり小さいものとする．

3. 動的配置分散データベース

分散データベースシステムにおいて各サイトのアクセスの傾向を事前に知ることは難しい．またそれぞれのサイトで頻繁にアクセスされるデータは時間とともに変化することがある．したがって最適なデータ配置を決定することは困難である．よって動的にデータ配置を行い，通信量の削減をはかる．

3.1 データの読み出しと書き込み

サイト S_a で生じたデータ x の読み出しは，S_a が x の所持サイトなら局所的に x を読み出し，$S_b(S_a \neq S_b)$ が x の所持サイトなら x を読み込み，S_b 上に x の複製を生成する．

サイト S_a で生じたデータ x の書き込みは，S_a が x の所持サイトなら局所的に x を更新し，$S_b(S_a \neq S_b)$ が x の所持サイトならば x を更新して S_a に格納し，S_b 上の x を破棄する．

S_a が他のサイト上のデータ x に書き込みを行う時，x の所持サイトが S_a に変更される．このことをデータの移動と呼ぶ．

3.2 データの位置管理

データが移動した後，各サイトはデータの移動先を知らなければそのデータを参照できなくなる．そこで各サイトはデータの移動先を知るための手段が必要となるが，そのための手法として以下のようなものが考えられる．

集中管理方式 全サイト中から一つ位置管理サーバー S_s を選び，S_s が全てのデータの位置を管理する．データ x が移動するときデータ ID と移動先を記した制御メッセージがこの S_s に伝えられ，自分の所持していないデータが必要となったサイトは S_s に問い合わせて参照先を知る．

放送問い合わせ方式 自分の所持していないデータ x が必要となったサイト S 上のトランザクション T は放

送により全サイトに x を所持していないか問い合わせる．x を所有するサイト S_o はTに，x が S_o に存在することを知らせる．Tは S_o にデータを送信するように要求する．

放送通知方式 データの移動が生じるとデータID，移動先のサイトを含んだ位置管理用の制御メッセージが放送により全サイトに通知される．そのメッセージを受け取ったサイトは各自が所有するデータ配置テーブルを更新する．各サイトが全てのデータの配置状況を把握できるため，トランザクションは直接データの要求を出す事ができる．

追跡探索方式 各サイトはデータ配置テーブルを持つ．データ x が S_a から S_b に移動するとき，S_a の持つデータ配置テーブルの x の所持サイトを S_b に変更する．あるサイト S_i において x が必要となったときはデータ配置テーブルを参照し，そこに記された x の所持サイト S_j に要求を送る．S_j は x を所持していれば要求に応え，所持していないときは S_j のデータ配置テーブルに記されている x の所持サイトに要求を転送する．この動作を，x を実際に所持するサイトにたどり着くまで繰り返し行う．サイト S_k から x を獲得できたとき，S_i は所有するデータ配置テーブル上の x の所持サイトを S_k に変更する．

上記の方式においてデータサイズ $\gg$ 制御メッセージサイズであるため，制御メッセージの放送による通信量は相対的に小さいと考えられる．要求が生じてからデータを獲得するまでの時間を考えると直接要求を送信できる放送通知方式が最も短い．したがって本研究では位置管理法として放送通知方式を選択する．

3.3 アルゴリズム

動的データ配置を行う分散データベースシステムの動作アルゴリズムを示す．サイト S で発生したトランザクション T 中の $op[x]$ を行う時を考える．各サイトはデータ配置テーブルを持つ．

- x が S 上に存在する時
  1. $op[x]$ が $read[x]$ ならば局所的に x を読み込む．$op[x]$ が $write[x]$ ならば x を更新．
  2. $op[x]$ が $write[x]$ であるとき，全サイトに制御メッセージを放送する．制御メッセージには x の破棄要求および x の所持サイトが S であることが記されている．
  3. 制御メッセージを受け取ったサイトは x を所持しているなら破棄する．また，データ配置テーブルに x の所持サイトが S であることを記録する．
- x が S 上に存在しない時
  1. データ配置テーブルから x の所持サイトを調べ，所持サイトにデータを要求する．
  2. データの所持サイトから x を S に読み込む．
  3. $op[x]$ が $read[x]$ ならば複製を S 上に生成し，データ配置テーブルに x の所持サイトとして S を加える．$op[x]$ が $write[x]$ ならば x を更新して S に格納．
  4. $op[x]$ が $write[x]$ であるとき，全サイトに制御メッセージを放送する．制御メッセージには x の破棄要求および x の所持サイトが S であることが記されている．
  5. 制御メッセージを受け取ったサイトは x を所持しているなら破棄する．また，データ配置テーブルに x の所持サイトが S であることを記録する．

3.4 考察

提案した分散データベースではデータの移動を行うことにより局所的な書き込みを可能として，通信量の削減をはかっている．分散ファイルシステムにおいて，これまでにファイルの移動 (migration) の研究が行われているが [3]，この中では各サイトの負荷の分散を目的としているため本研究とは移動の手法が異なり，必ずしも通信量が最小になるわけではない．

ここで従来の分散データベースシステムと提案する分散データベースシステムの通信量について考える．ここでトランザクションの操作の衝突によるアボートについては考えない．つまりアボートに伴う無駄な通信が生じないと仮定する．ここで通信量とはサイト間を流れるデータのサイズと制御メッセージのサイズの総和とする．

3.4.1 コールバック手法を利用した分散データベース

従来の分散データベースシステムではキャッシュを用いて通信量の削減を行っている．しかしキャッシュを用いる方法では通常 *read* 時の通信しか削減できない．そこでコールバックと呼ばれる手法を用いて *write* 時の通信を減らす方法が提案されている [4]．そこで比較の対象にはキャッシュの管理にコールバック手法を用いた分散データベースシステムを選択する．以下この分散データベースを **CB**(Callback) と呼び，本稿で提案するデータを動的に配置する分散データベースを **DA**(Dynamic Allocation) と呼ぶ．

以下 CB の動作を記すサイト S で発生したトランザクション T 中の $op[x]$ を行う時を考える．S_o は x の所持サイトを表す．

- x が S 上に存在する時
  1. $op[x]$ が $read[x]$ ならば局所的に x を読み込む．$op[x]$ が $write[x]$ であるとき，x の write 権を持っているならば直ちに x を更新し，持っていなければ S_o に write 権を要求して，それを獲得した後に x を更新．
  2. $op[x]$ が $write[x]$ であるとき，x をキャッシュに保持しているサイトに x の破棄を要求する制御メッセージを放送する．write 権は要求があるまで S が所持する．
  3. 制御メッセージを受け取ったサイトはキャッシュ内の x を破棄する．

- x が S 上に存在しない時

  1. S_o に x を要求する．
  2. S は $op[x]$ を行い，送られてきた x をキャッシュに格納する．
  3. $op[x]$ が $write[x]$ であるとき，x をキャッシュに保持しているサイトに x の破棄を要求する制御メッセージを放送する．write 権は要求があるまで S が所持する．
  4. 制御メッセージを受け取ったサイトはキャッシュ内の x を破棄する．

S_o が S から x の要求を受けた時は以下のように動作する．

1. S_o は x の write 権を所持しているならば直ちに x を S に送り，所持していない時は write 権を所持しているサイト S_w に write 権と最新の x を要求する．これをコールバックと呼ぶ．
2. S_w から write 権と最新の x を受け取った S_o は $op[x]$ が $read[x]$ ならば x を，$write[x]$ ならば write 権と x を S に送る．

$write$ 権を持つサイト S_w は S_w 上の x のキャッシュに対して $write[x]$ を行え，変更を加える度に S_o に書き戻す必要はない．したがって S_w で $write[x]$ が連続する場合，他のキャッシュ管理方式よりも通信量が少なくなる．

CB においてはデータ x の所持サイトは変化しない．ただし他のサイトにおいて x が更新された場合，コールバックが発生するまでの間，最新の x は更新を行ったサイトのキャッシュにのみ存在する．すなわち，必ずしも所持サイトが最新の x を所持しているとは限らない．

3.4.2 通信量の比較

ここでデータ x の所持サイトを S_o とし，S_o 以外のサイト S_w が x の書き込みを行ったとする．その後 S_o，S_w 以外のサイト S 上で $read[x]$ が生じた時を考える．CB において $read[x]$ を行うときはまず所持サイトである S_o に x を要求する．S_o は最新の x を持たないため S_w に最新の x を書き戻すように要求し，x を獲得してから S の要求に応える (図 1)．それに対して DA では所持サイトが S_w に変化しているので，直接 S_w に x を要求して最新の x を獲得することができる (図 2)．

この状況における通信量を考える．$read[x]$ を行う時，CB では S_w から S_o，S_o から S と 2 回のデータの通信が必要となる．このような read をコールバック read と呼ぶことにする．DA では S_w から S と 1 回のデータの通信でよい．この時点でデータ通信 1 回分の差が CB と DA の間に生じる．

ただし CB においては S_o に x が置かれるため，これが破棄される前に S_o 上で x に対する操作が生じた場合，データの通信が 1 回省略される．

制御メッセージの通信量を考えないとき，DA と CB の通信量の差はすべてこのような状況によってのみ生じ，DA の通信量は常に CB のそれよりも少ないと言える．

DA と CB の通信量の差はデータの移動回数の差により生じるため，データサイズが大きければ大きい程差がつく．したがってデータサイズが大きい程 DA が有効となる．逆にデータサイズ ≫ メッセージサイズという関係が成り立たなくなると制御メッセージの放送により生じる通信量が無視できなくなり CB の通信量の方が少なくなる．

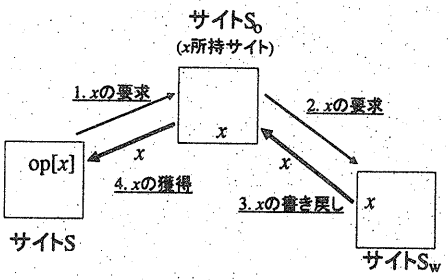

図 1 データアクセス (CB)

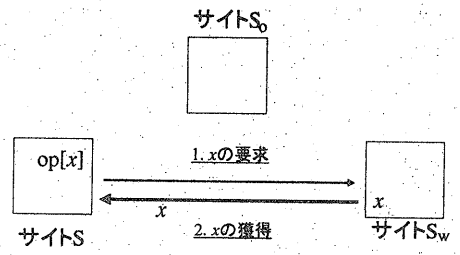

図 2 データアクセス (DA)

3.5 DA の問題点

DA はデータの位置管理などに制御メッセージの放送を利用しているためサイト数に限度がある．サイト数が多い場合は放送のためのコストが無視できなくなる．

また，記憶装置の容量は有限であるため，あるサイトにデータが集中するとそのサイトへのデータ移動が不可能になる場合がある．十分な容量の記憶装置を用意することによってこの問題を回避することができる．

4. シミュレーション

CB と DA の通信量の差を定量的に調べるためにシミュレーションを行う．

4.1 シミュレーションの詳細

シミュレーションにおける各種の値を次のように設定する．トランザクション数 10000，1 トランザクションあたりのオペレーション数を 10，データサイズ 100KB，制

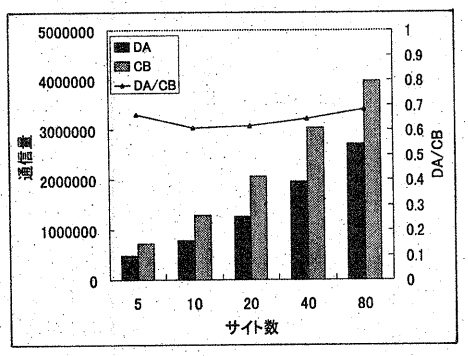

図 3　通信量 (パラメータ:サイト数, データ数:1000, read の割合:80%, 最大温度 10000, 最小温度 10, 温度減少割合:100)

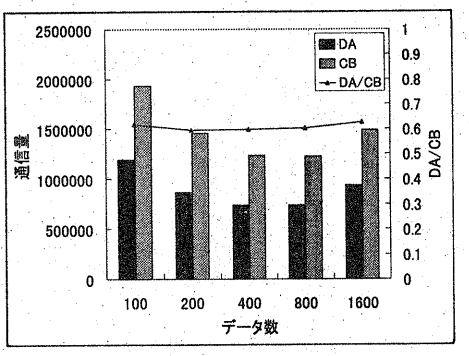

図 4　通信量 (パラメータ:データ数, サイト数 10, read の割合:80%, 最大温度 10000, 最小温度 10, 温度減少割合:100)

御メッセージサイズ 0.02KB である. またサイト数, データ数, *read* の割合, 時間的局所性をパラメータとする. データは各サイトに同じ数ずつ初期配置する. 時間的局所性とはあるトランザクションにアクセスされたデータはその後のトランザクションによりアクセスされる確率が高いという性質であり, 最大温度 T_{max}, 最小温度 T_{min}, 温度減少率 D の組合せによって表される. 各データは温度を持ち, 高ければ高いほどアクセスの対象として選択される確率が高い. 一度アクセスされると温度は T_{max} にまで上昇し, 一定時間が経過するごとに D ずつ減少していくが, T_{min} まで下がるとそれ以上は下がらない. シミュレーションは $T_{min} = 10$, $D = 100$ に固定し, T_{max} を変化させて行った.

4.2　シミュレーション結果と考察

図 3〜図 6にシミュレーションの結果を示す. 棒グラフが通信量を示している. 左が DA, 右が CB である. また各図における折れ線グラフは CB の通信量を 1 としたときの DA の通信量を表す. 全ての場合において DA の通信量は明らかに CB のそれよりも少なく, 時間的局所性の高い状況 (最大温度=10000) においては約 30%の通信量が削減されていることがわかる.

パラメータであるサイト数, read の割合, データ数を変化させたとき, 通信量は変化するが, DA/CB はおよそ 0.65 付近で一定である. しかし時間的局所性を変化させると 0.6 から 0.9 まで変動している. ここで通信量の差を生みだす 3.4.2節で述べたコールバック read の発生確率を調べると, 全ての状況において全オペレーションの 2%から 4%を占めることがわかった.

したがって CB における通信量は大半が所持サイトに最新のデータが存在する場合の read によるものであることがわかる. 時間的局所性の高い時, DA, CB ともに read 操作の多くが局所的なものになり, 通信量は全体として減少する. しかしコールバック read の数は大きく変化しないため, コールバック read によって生じた通信量の, 全体の通信量に占める割合は大きくなり DA/CB が減少する.

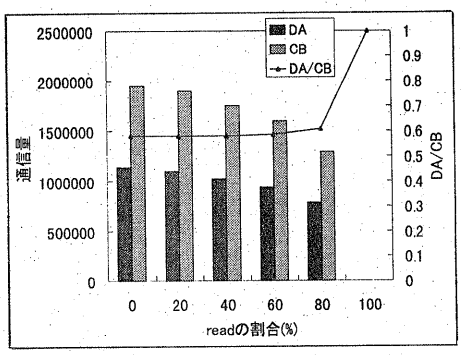

図 5　通信量 (パラメータ:read の割合, データ数:1000, サイト数:10, 最大温度 10000, 最小温度 10, 温度減少割合:100)

5.　並行性制御

今回のシミュレーションはトランザクションの操作の衝突によるアボートが発生しないという仮定のもとで行った. 実際には衝突に伴うアボートが発生するために何らかの並行性制御を行う必要がある. 本研究では並行性制御の手法として 2.2節で述べた時刻印順方式を取り上げ, 更に以下で述べる多版並行性制御を組み合わせることによってトランザクションのアボート数を減らすことを検討している.

5.1　多版並行性制御の導入

トランザクション処理においては, トランザクションに含まれるデータ操作が全て反映されるか, 全て取り消されるかのどちらかしかあり得ないという原子性が保証されなければならない. したがってあるトランザクション T がアボートされた場合, 原子性を保証するために T によって更新されたデータの値は全て T の実行前の値に戻され再実行される. つまりトランザクションのアボー

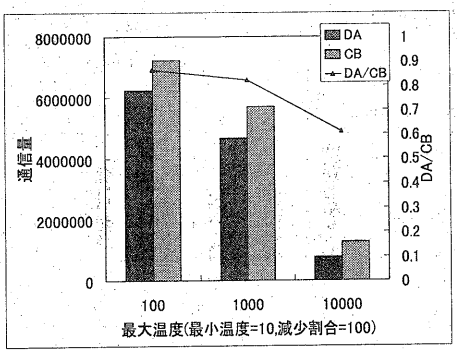

図 6　通信量 (パラメータ:最大温度，データ数:1000，サイト数:10，read の割合:80%，最小温度 10，温度減少割合:100)

トは通信量増加の可能性を持つ．そこで通信量削減のためにアボート数を減少する多版並行性制御 [5] の導入を考える．

多版並行性制御においてはデータ x の書き込みの際，古い x の値を保存する．すなわち x には複数の版 (version) が存在する．各オペレーションは対象となるデータの適切な版にアクセスすることでアボートを避けることができる．

多版並行性制御の利用により，提案する分散データベースに限らず通信量を削減することが可能であるが，提案型分散データベースでは以下のような問題がある．

5.2　データの位置管理

データの動的配置を行う分散データベースシステムでは，あるデータの各バージョンは複数のサイトに分散する．よって各サイト上のトランザクションがそれを参照できるようにするためには，各データの各バージョンの位置管理をする必要が生じる．この位置管理は 3.2節で述べた位置管理法に以下のような変更を加えることで実現できる．

- 位置管理用制御メッセージの内容 (移動したデータの ID および移動先のサイト ID) にそのデータのバージョンのタイムスタンプを追加する．
- 各サイトの所持するデータ配置テーブルに変更を加える．データ x のエントリにリスト L_{ver} を加える．L_{ver} の各ノードには x のそれぞれのバージョンのタイムスタンプとリスト L_s が格納されている．リスト L_s はそのバージョンを所持するサイトのリストである．

6.　まとめ

本稿では分散データベースにおいてデータを動的に配置することにより通信量を削減する手法を提案した．

シミュレーションの結果，時間的局所性の高い状況において提案する分散データベースはコールバックを用いてキャッシュを管理する手法と比べて有効であることがわかった．

今回のシミュレーションにおいては，トランザクションの実行中に衝突が無いものと仮定したが，衝突がある場合はトランザクションのアボートが生じる．このときアボートされたトランザクションにおいて行われていた通信が無駄になり，再実行時に同じ通信を行う．

トランザクションのアボートの回数を減らして，アボートにより生じる通信量の増加を防ぐために多版並行性制御の導入について考え，そのための手法を検討した．

今後はトランザクションの衝突がある状況で並行性制御を考慮したシミュレーションを行い，その性能を調べる予定である．

References

[1] Salvatore T.March and Sangkyu Rho, "Allocating Data and Operations to Nodes in Distributed Database Design", *IEEE Trans. on Knowledge and Data Engineering*, Vol.7, No.2, pp.305-317, April 1995

[2] Bernstein, P.A., Shipman, D.W. and Wong, W.S., *Concurrency Control and Recovery in Database Systems*, Addison-Wesley, 1987.

[3] Richard T. Hurley, Soon Aun Yeap "File Migration and File Replication: A Symbiotic Relationship", *IEEE Trans. on Parallel and Distributed Systems*,Vol.7, No.6,pp578-586, June, 1996

[4] Markos Zaharioudakis and Michael J.Carey, "Hierarchical,Adaptive Cache Consistency in a Page Server OODBMS", *IEEE Trans. on Computers*,Vol.47, No.4, pp427-443,April,1998

[5] Bernstein, P.A. and Goodman,N., "Timestamp-Based Algorithms for Concurrency Control", *Proceedings of 6th Intl. Conf. on Very Large Data Bases*, pp.285-300, 1980