統計的手法による弱一貫性データの複製制御方法

4G－10

山下 高生，小野 諭
NTT ソフトウェア研究所

1 概要

インターネットによる計算機ネットワークの世界的な広がりにより、広域分散環境での、ディレクトリ・サービスやWWWなど様々なアプリケーション(AP)でデータのサービスが重要となってきている。このデータ・サービスには、耐故障性、応答時間、一貫性、スケーラビリティーが重要となるが、広域分散環境でスケーラビリティーを実現するためには、データ複製の利用が不可欠である。しかし、データ複製を行った場合、特に、応答時間と一貫性は、トレードオフの関係にある。必要とされる応答時間と一貫性は、APによって様々であり、また単一のAPでも、状況によって、より新しい情報が必要など、動的に要求される一貫性が変化する場合がある。よって、複製方法には(1)一貫性を弱めて短い応答時間を実現できること、(2)応答時間と一貫性レベルの特性を各APの平均的な要求に応じて調整可能であることおよび(3)必要時には、データを必要とする計算機側の動作の変更で動的に応答時間と一貫性を調整可能であることが必要となる。本発表では、まず、一貫性と応答時間を、それぞれ読出し時のデータの新しさとコーラムに含まれるノード数として表現する。そして、データ複製の一つであるコーラム手法において、非同期にデータ更新を伝搬させた時、読出されるデータの新しさが、時間的経過に伴って、確率的な意味で向上することに着目し、コーラムに含まれるノード数と更新伝搬の周期の変更によって上記3つの目的を実現する複製方法を提案し、その実現性を検討する。

2 データ複製と非同期更新に関する既存技術

データ複製手法は、Write-Allおよびコーラム手法に分類される[4]。前者は、特にread時の応答時間を短くできるが、耐故障性ということからは、ノード故障、リンク故障、ネットワークの分割の処理が複雑になる。後者は、耐故障性という面からは、実現が容易である反面、特にwriteまたは、read操作のどちらかは、操作の対象となるノード数が常に多いために、応答時間を短くすることが困難である。

一方で、Write-All手法におけるLazy Replication[5]やCommit Propagation[1, 2]、あるいは、市販DBMSのAsynchronous Update機能のようにコミットされたトランザクションを非同期に複製ノードに伝搬させる非同期更新方法が存在する。この非同期更新を用いると、ある程度シリアライザビリティを犠牲にすることになるが、スループットの向上、応答時間の削減、可用性の向上[2]を実現することができる。

3 統計的手法による複製方法

本章では、まず、複製制御方法の提案を行う。次に、本手法の実現性を非同期更新の例題を用いて検討する。

3.1 複製方法

最初に、あるデータに関して、最新の更新を反映しているノード数の複製ノード全体に対する割合を、全体最新度と呼ぶこととする。コーラム手法では、writeとreadコーラムに含まれるノードに対して各操作を行うため、非同期更新を導入しない時、全体最新度は変化しない。しかし、非同期更新を導入した場合、最初のwrite操作では、writeコーラムに含まれるノードにのみ更新が行われるが、時間経過とともに、更新が伝搬され全体最新度が高くなっていく。

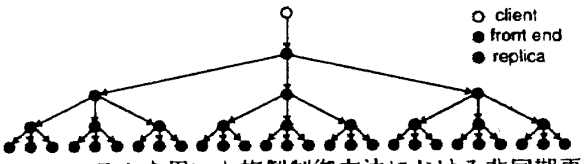

図1: コーラムを用いた複製制御方法における非同期更新

本発表で提案する複製制御方法は、この全体最新度の時間経過に伴う向上に着目したものである。本方法は、非同期更新方法から確率的に計算される全体最新度に応じ、APの要求する一貫性と応答時間に合わせて非同期更新の周期、writeおよびreadコーラムのサイズを変化させるものである。

ここで、本方法における一貫性と応答時間について説明する。本手法の対象としては、銀行のOLTPのようなシリアライザビリティが問題となるトランザクション間の干渉の強い厳密な一貫性を必要とするものではなく、ディレクトリサービスなど、トランザクション間の干渉の少ない、弱い一貫性のデータを対象としている。このため、一貫性としては、読出し時のデータが更新をどれだけ反映しているかが重要であるため、読出し時のデータの新しさを用いる。そして、この新しさを「時間T_a前までに行われた更新が、ある全体最新度f_aのノードに対して伝搬されている確率がp_aである」として表す。次に、単一サーバに対する応答時間は、単純にはサーバの処理時間と通信遅延時間の2倍の和であり、複数のサーバを選択して各操作を行った時には、各サーバへの応答時間の最大値が全体としての各操作の応答時間となる。今、この各サーバの応答時間が、ある確率分布をしていた場合、ランダムに選択された数が多いほど、長い応答時間をもつサーバが含まれる確率が高くなる。よって、応答時間として、コーラムのサイズを用いることとする。

本方法のwriteおよびread操作は、次のように行う。まず、read操作は、応答時間からコーラムのサイズを決定し、このコーラムに含まれるノードに対してread操作を行う。また、より新しいデータが必要な場合は、非同期更新方法から計算される全体最新度を元に、read操作を行うノード数を増していく。write操作は、非同期更新方法から計算される確率p_aでの全体最新度f_aの時間変化から、非同期更新周期T_pに対する比率rを求め、$rT_p = T_a$となるようにwriteコーラムのサイズとT_pを決定し、このサイズのコーラムに含まれるノードに対してwrite操作を行う。各操作の詳細、特にwriteコーラムのサイズとT_pの決定方法については、今後の検討課題である。

3.2 例題

以下では、本方法の実現性を検討するために、例題を用いて実際の数値を解析する。まず、非同期更新方法を説明する。図1に示すように、データ更新を行うクライアントは、フロントエンド・ノードに更新操作を送信する。フロ

A Data Replication Method by Probabilistically Changing Quorum Size
Takao YAMASHITA, Satoshi ONO
NTT Software Laboratories

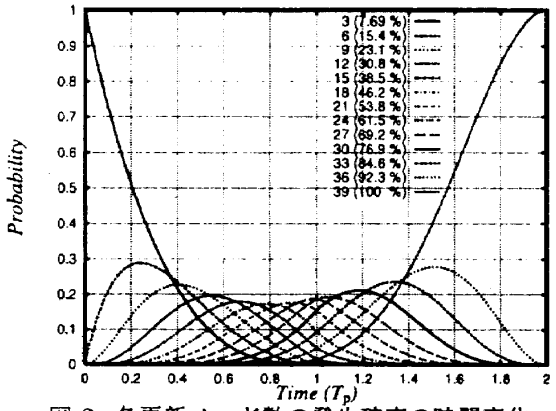

図 2: 各更新ノード数の発生確率の時間変化

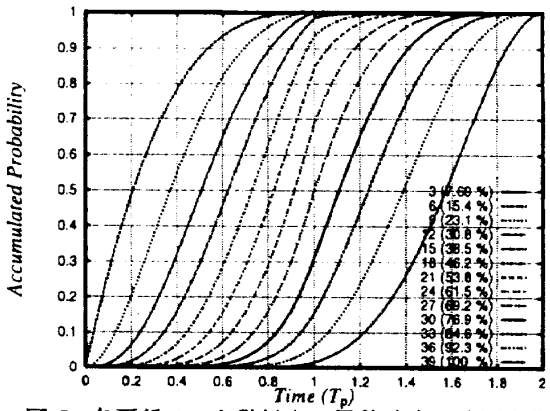

図 3: 各更新ノード数以上の累積確率の時間変化

ントエンド・ノードは、ランダムに選択した n_f 個のノードに同時に更新を行う。そして、更新をされたノードは、階層的にさらに n_f 個のノードに更新を行っていく。ここで、各複製ノードの更新周期は、T_p とする。

コーラムの決め方には、幾つかの方法があるが、本発表では、提案した方法の実現性を明らかにすることを目的としているため、最も簡単な多数決合意 [3] を用いた方法を用いる。また、複数の読出されたデータの中から、最新のデータを選択する方法には、バージョン番号を用いた方法とタイムスタンプを用いた方法がある [4] が、バージョン番号を用いた場合、write コーラムが全ノードの過半数で無い時に、最新データの区別ができなくなるため、ここでは、タイムスタンプによる方法を用いる。

3.3 全体最新度の時間変化

ここで例題として用いたモデルは、図 1に示すように、$n_f = 3$、複製ノードの総数は、39 個である。そして、最長で $2T_p$ の間に、更新が全ノードに伝搬される。

このような非同期更新を行ったとき、更新されているノード数 n_u は、$3 \leq n_u \leq 39$ の 3 の倍数となる。今、各ノードが同じ周期 T_p で更新を伝搬していて、かつランダムに更新ノードを選択して伝搬していくことで、各ノードで更新を受信してから、次のノード群へ伝搬していくまでの時間が、t_u となる確率密度関数は、$0 \leq t_u \leq T_p$ において、$1/T_p$ の一定値となる。この確率分布を元に、各更新ノード数が各時刻で生じる確率を計算すると図 2のようになる。横軸が時間であり、縦軸が各更新ノード数の生じる確率を表している。各曲線は、各更新ノード数の発生確率である。この図からわかるように、時間経過とともに、更新されたノード数の多いものの発生確率が高くなっていることがわかる。

3.4 読出しノード数とデータの新しさの関係

図 3に更新ノード数が n 以上である累積確率の時間変化を示す。この図をもとに、確率が p でノード数が n 以上となる時間変化を図 4に示す。同図から、AP の要求としてある時間 T_r までの新しさのデータをある確率 p_r で読出したい時、read 操作を行うノード数 n_r を次のように計算することができる。今、更新されたノード数が、p_r の確率で n_u、全体のノード数を $N = 39$ とする。多数決合意を用いているため $n_r + n_u > N$ である。よって、$n_r \geq N - n_u$ のノードにアクセスすることで最新のデータの読出しを実現できる。ここで、$n_r = N - n_u + 1$ とした時のノード数を、図 4の縦軸の括弧内に記す。この図から前述のアプリケーション要求の T_a が長いほど、また p_a が小さいほど、読出しノード数を削減できることが分かる。

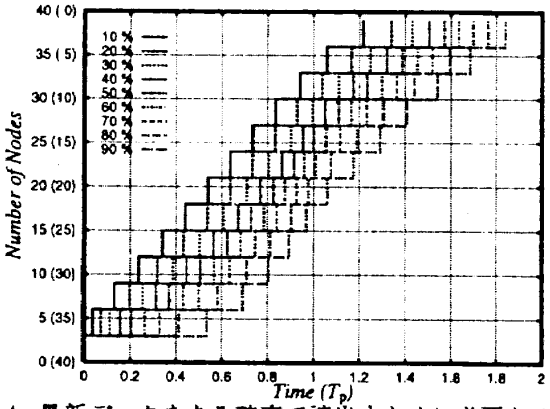

図 4: 最新データをある確率で読出すために必要なノード数の時間変化

4 今後の課題

今後、write および read 方法および非同期更新方法の詳細化を行っていく。

謝辞

研究を支援下さる市川晴久広域コンピューティング研究部長に感謝します。

参考文献

[1] A. El Abbadi and P. C. Aristides. Fast read only transactions in replicated databases. In *Proc. IEEE International Conference on Knowledge and Data Engineering*, pp. 246–253, 1992.

[2] J. J. Fischer and A. Michael. Sacrificing serializability to attain high availability of data in an unreliable network. In *Proc. 1st ACM Symposium on Principles of Database Systems*, pp. 70–75, May 1982.

[3] D. K. Gifford. Weighted voting for replicated data. In *Proc. 7th Symposium on Operating System Principles*, pp. 150–162, December 1979.

[4] A. Helal, A. Heddaya, and B. Bhargava. *Replication Techniques in Distributed Systems*. Kluwer Academic Publishers, 1996.

[5] Rivka Ladin, Barbara Liskov, and Sanjay Ghemawat. Providing high availability using lazy replication. *ACM Transactions on Computer Systems*, Vol. 10, No. 4, pp. 360–391, November 1992.