1R−8

オブジェクト・サーバにおけるオブジェクト共有管理

何　千山　森本康彦　福田剛志　小坂一也
日本アイ・ビー・エム株式会社　東京基礎研究所

1　はじめに

我々は、拡張可能な関係データベースに基づき、マルチメディア環境でマルチメディア・オブジェクトを共有するためのオブジェクト・サーバCOSMOSを構築している。この論文では、オブジェクト・サーバにおけるオブジェクト共有管理を中心にし、クライアントーサーバ間の効率的なオブジェクト転送方法、オブジェクト・キャッシング、トランザクション管理を述べる。

クライアントーサーバ・アーキテクチャを取っているOODBMSはオブジェクトサーバとページサーバに大別できる。オブジェクトサーバでは、ネットワーク上での転送単位はオブジェクトで、サーバが問い合わせ機能を持って必要なサブセットを送る。ページサーバでは、ネットワーク上での転送単位はページで、サーバが問い合わせ機能がなくクライアントが問い合わせ機能を持っている。ページサーバでは、よいクラスタリング及びインデックスを付けている場合、ページフォールトが少ないので、ネットワーク上のページ転送量は少なくなり、また、クライアント側が速いCPUと大きいキャッシュを持っている場合、高速アクセスができる。一方、オブジェクトサーバでは、上述の条件があまりよくない場合ではページサーバと同じぐらいのパフォーマンスが得られる[1]。異機種のサーバとクライアント環境では、ページの構造が異なるので、ページサーバのアプローチは取れない。COSMOSのプロトタイプはOS/2とAIX*に基づき、クライアントはOS／2の上で、サーバはAIXの上で構築されるので、オブジェクトサーバのアプローチを取る。図1はCOSMOSのクライアントーサーバ・アーキテクチャの構成図である。

2　効率的なオブジェクト転送方法

COSMOSでは、TCP/IPを基づいたRPCを使ってオブジェクトをクライアントとサーバの間に転送する。RPCでオブジェクトを転送する場合、一つのパケットサイズ以内（0から1．5 KBぐらいまで）のオブジェクト転送時間はほとんど同じであるので、オブジェクトが小さくなるほど、転送のオーバヘッドが大きくなる。そこでなるべくいくつかのオブジェクトをまとめ、パケットサイズ以上の単位で転送することが望ましい。商用のページサーバを基本とするOODBMSの転送単位はページ（4 KBぐらい）であるので、これはパフォーマンスを上げる一つ重要なポイントである。COSMOSでは、RPCの転送単位はコンテナ・オブジェクトであり、コンテナ間にリンクを張っている場合、リンクを辿ってリンクされるコンテナを送る。クライアントからサーバにコンテナを送る場合、まずコンテナをバイド列にパックする。アプリケーション・プログラムをコンパイルする時に、永続クラスの中のリンクが検出され、リンクを辿るパック用メソッドが作成される。実行時にパック用メソッドを呼び出し、ヒープ上にあるコンテナをキャッシュ・メモリへバイド列にパックしてサーバに送る。

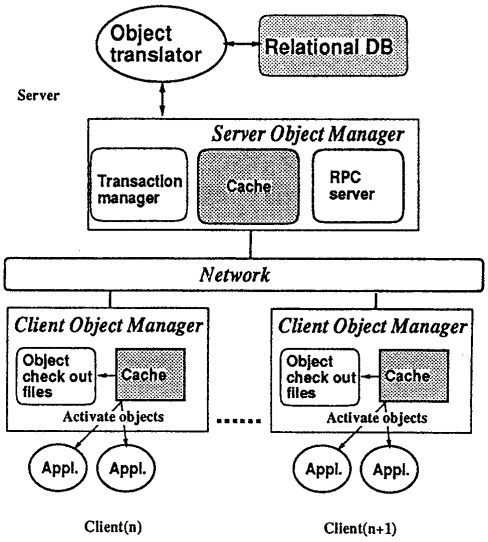

図1: クライアントーサーバの全体構成

3　オブジェクト・キャッシング

データベースのアクセスとネットワーク上のオブジェクト転送を減らしてパフォーマンスを上げるために、キャッシングは不可欠である。サーバは大容量キャッシュを持ち、各クライアントからのアクセスがある場合、まずキャッシュにあるかどうかをチェックする。クラス情報は頻繁にアクセスされるので、サーバの初期化時にRDBMSにあるすべてのスキーマ情報がキャッシュに置かれる。また、クライアントからサーバのオブジェクトを取り出す時に、まずRDBMSのタプルをオブジェクトにトランス

Object Sharing Management in an Object Server
Qianshan He, Yasuhiko Morimoto, Takeshi Fukuda and Kazuya Kosaka
Tokyo Research Laboratory, IBM Japan Ltd.
*IBM社の商標です。

レートしてキャッシュに置いてからクライアントに送る。クライアントにもキャッシュがあり、同一クライアント内の複数のアプリケーションから共用される。キャッシュはオブジェクトIDのhashテーブル、キャッシュ・ディレクトリ、キャッシュ本体からなり、コンテナ・オブジェクトの単位でキャッシングを行ない、サーチの対象はオブジェクトIDである。キャッシュが満ぱいになる時、キャッシュを更新する必要がある。キャッシュ更新アルゴリズムとしてleast-recently-used(LRU)アルゴリズムは効率でよく使われているが、オブジェクトサーバでキャッシュしたオブジェクトが可変長で、キャッシュの空きスペースのリアロケーションは非効率であるので、first-in-first-outの更新アルゴリズムを採用する。

サーバとクライアントの両方はキャッシュを持っているので、キャッシュ一貫性管理が必要となる。キャッシングしたすべてのオブジェクトがバージョン番号を持っている。あるクライアントのあるオブジェクトが更新される場合、このクライアントのキャッシュとサーバのキャッシュは同時に更新されるが、このオブジェクトを持っているすべてのクライアントのキャッシュを更新する必要がない。サーバがクライアントからread要求を受け取った後、サーバのオブジェクトのバージョンとクライアントのオブジェクトのバージョンを比べ、一致しない場合だけにクライアントに送る。必要に応じてサーバからコピーするので、無駄なオブジェクト転送を避けることができる。

4 トランザクション管理

各クライアントからの同時アクセス、障害回復を制御するために、トランザクション管理が必要となる。各クライアントにはローカル・データベースの存在を仮定していないので、トランザクション管理はすべてサーバで集中管理する。アプリケーションには二相ロック（two-phase locking）と楽観的ロック（optimistic locking）の二種類の並行制御プロトコルを提供する。二相ロックは主に資源を確保してアクセスしたいトランザクションに使われ、楽観的ロックは主にチェック・インーチェック・アウト使い方の長いトランザクションに使われる。クライアントは大量のオブジェクトをチェック・アウト・ファイルに持ってくることができる。二相ロックではコンフリクトが起こる時、時間軸の順にトランザクションがブロックされる。楽観的ロックではコンフリクトが起こる時、トランザクションはブロックされないが、トランザクションのコミットフェーズにアボートされる。また、二つのアプリケーションが異なったプロトコル（一つは二相ロック、もう一つは楽観的ロック）を使用する場合、楽観的ロックのトランザクションが先にコミットしようとしても、もし二相ロックのトランザクションとコンフリクトすれば、楽観的ロックのトランザクションがアボートする（二つの楽観的ロックのトランザクションの場合、先にコミットしょうとするトランザクションは必ずコミットする）ので、二相ロックのトランザクションは資源確保を保証する。

リンクでつながっているツリー構造あるいはグラフ構造の複合（complex）オブジェクトに対してロックをかける場合、一度親にロックをかければすべての子孫（グラフ構造では辿られる範囲）もロックされるのは効率的であるが、極端な場合は全データベースをロックする可能性があるのであまり現実ではない。しかし、すべてのオブジェクトをロックの対象にすると、オーバヘッドが大きくなる。我々はその中間を取り、いくつかの独立していないオブジェクトからなるコンテナ・オブジェクトをロックの単位とし、コンテナのオブジェクトIDに対してロックをかける。そうすると、アプリケーションでサーバにアクセスできる単位はコンテナで、コンテナの中から他のコンテナをリンクを持っている場合、必要に応じて（on demand）でロックをかけてアクセスする。コンテナ・オブジェクトはRDBMSのいくつかのテーブル、タプルに跨る可能性があるので、RDBMSのトランザクション・マネジャを利用する場合、それらのタプルをロックする必要がある。ロックの最小単位がタプルではなくテーブルあるいはページである場合、この方法は適切ではなく、RDBMSと独立にサーバ・オブジェクト・マネジャはトランザクション管理機能を持たなければならない。今回我々は使用しているRDBMSのStarburst[2]はテーブルに対してロックしているので、Starburst本体のロックを外してサーバ・オブジェクト・マネジャでトランザクションとロック管理を行なっている。サーバ・オブジェクト・マネジャにおけるトランザクション・マネジャとロック・マネジャはStarburstのトランザクション・マネジャ、ロック・マネジャを改良して利用し、オブジェクトIDにロックがかけられるようにする。

5 むすび

この論文では、オブジェクト・サーバCOSMOSにおけるオブジェクト共有管理を実現するための仕組を述べた。現在では詳細設計段階が終り、実装中である。その後にページ・サーバに基づいたOODBMSと比べてパフォーマンス評価を行なう予定である。

参考文献

[1] D. J. DeWitt and D. Maier, “A Study of Three Alternative Workstation-Server Architectures for Object Oriented Database Systems”, Proceedings of the 16th VLDB Conference, pp.107-121, 1990.

[2] G. Lohman et al, “Extension to Starburst: Objects, Types, Functions, and Rules”, CACM, 34(10), pp.94-109, Oct. 1991.