インスタンス移動に基づく最適化可能 DBMS *

4Aa－7

鬼塚 真　　小林 伸幸　　岡田 敏†

NTT 情報通信研究所‡

1 はじめに

近年，通信の高速化及び計算機の低コスト化と高速化に伴いクライアントサーバ (以下 C/S) 構成の DBMS が重要視されつつある．C/S 構成においては，クライアントとサーバでの処理分担の最適化と通信量・回数の軽減が性能上非常に重要である．つまりクライアントとサーバの計算機の処理能力と処理量，ホスト間の通信速度を考慮して最適な C/S 構成を決定できることが望ましい．しかし，通常の DBMS ではソケット通信インタフェースを用いて C/S を構成するため，C/S 間のインタフェースが固定的になってしまい，通信の高速化・計算機の高速化・クライアント数の増加などの環境変化に対応して C/S 構成を変更できないという問題があった．

本論文では，DBMS の C/S 構成を柔軟に変更するための DBMS インスタンス移動機構と，この機構に基づくアーキテクチャを提案する．本機構は分散オブジェクト技術に基づき，DBMS インスタンスを C/S 間で移動することで，DBMS の C/S 構成の切口を変更し，様々な環境変化に対応した DBMS を提供することが容易になる．

本稿の構成を示す．2 章では現状の C/S 構成の DBMS における問題点について整理し，インスタンス移動機構の必要性について論じる．3 章ではインスタンス移動について述べ，4 章ではインスタンス移動の実現方法について検討する．5 章では今後の課題について整理する．

2 現状の C/S 構成 DBMS の問題点

DBMS インスタンスの移動機構の必要性を説明するため，通常の C/S 型 DBMS の問題点を指摘する．一般的な C/S 型の DBMS の具体例として，筆者らが実装したデータ分析アプリケーション用 ORDBMS である LiteObject[1] のアーキテクチャを図 1 に示す．

LiteObject のデータベースサーバ部は以下の主要なサブシステムから構成されている．

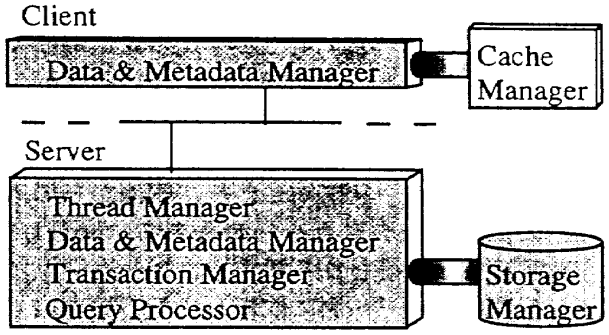

図 1: LiteObject アーキテクチャ

Thread Manager 個々のクライアントに対し 1 スレッドが対応するよう管理する．

Data & Metadata Manager アプリケーションが登録したスキーマ・メソッド及び実データを管理する．

Transaction Manager 排他制御管理やロールバックのためのログの生成・削除を制御する

Query Processor 問い合わせ文のパーズ，最適化，実行を行なう．

一方，LiteObject のデータベースクライアント部はアプリケーションに対し API を提供し，またクライアントサーバ間の通信回数を軽減するためスキーマ・メソッド及び実データのキャッシュを行なっている．

図 1 の構成において，クライアント数が多いアプリケーションを想定した場合，サーバの負荷を軽減するために問い合わせ文のパーズをクライアントで実行する対処策が考えられる．

また通信回数を削減するために，OID のリストを管理するクラスを用意し，このインスタンスをサーバ側からクライアント側へ移動することにより，OID をクライアント側で配布可能にする対処策も考えられる．

しかし，LiteObject に代表される通常の DBMS では C/S の切口が固定されているため，これらの要求を容易に満たすことができないという問題がある．

---

*Tunable DBMS Based on Instance Migration Mechanism

†Makoto ONIZUKA, Nobuyuki KOBAYASHI and Satoshi OKADA

‡NTT Information and Communication Systems Laboratories

3　インスタンス移動

2章の問題を解決する手段として，本論文ではDBMSインスタンス移動について述べる．ここでDBMSインスタンスとは，DBMSを構成するクラスのインスタンスであり，例えばLiteObjectでは，アプリケーションが定義するクラスを管理するClassクラスのインスタンス，トランザクションに関するTransactionクラスのインスタンスなどが存在する．

3.1　移動方法

DBMSインスタンスをDBMSのC/S間で移動する方法については，インスタンスの特徴に応じて以下の2種類がある．

複数クライアントが共有するインスタンス サーバ側からクライアント側へ移動する場合，インスタンスを適切にロックして，そのコピーをクライアントへ移動(キャッシュ)する．例えば，アリケーションが定義したクラスを管理するClassインスタンスやInstanceインスタンスなどが対象となる．

1クライアントが占有するインスタンス 移動したインスタンスを移動元から参照するため，プロキシを生成する．例えば，問い合わせを管理するSqlインスタンスやトランザクションを管理するTransactionインスタンスなどが対象となる．

3.2　移動の単位

DBMSインスタンスをDBMSのC/S間で移動する単位については，以下の2通りがある．

- 特定の1インスタンス．
- コンフィグレーションによりグループ化されたインスタンス群．例えば，Sqlインスタンスをルートとして作成されたパーズツリーを構成するインスタンス群は，グループ化して移動されるべきである．

4　実現方法

インスタンス移動を実現するDBMSのアーキテクチャを図2に示す．

Distributed Object Manager 分散オブジェクト環境であり，3章で述べた機能である，C/S間でのインスタンス移動，C側へのインスタンスキャッシュとS側でのインスタンスのロック，移動したインスタンスと通信するためのプロキシ，そしてインスタンスの移動の単位を決めるためのコンフィグレーションの機能をサポートする．これらの機能

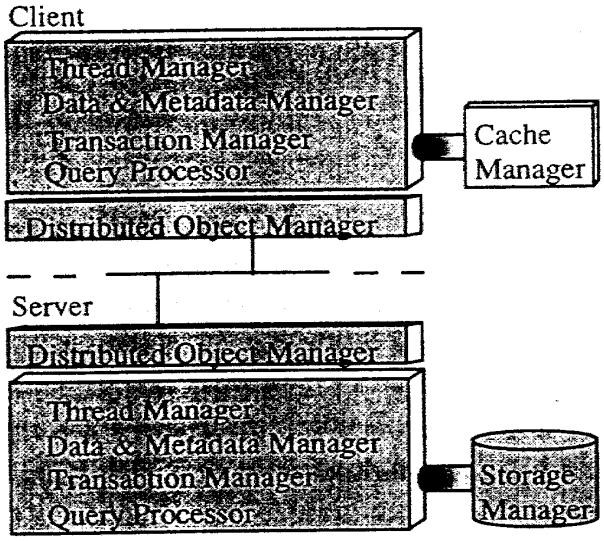

図2: インスタンス移動のためのアーキテクチャ

を用いることで，DBMS ServerとClientでは，全DBMSインスタンスがローカルにあるものとして扱うことができ，且つ柔軟にC/S構成を変更することが可能となる．

Client 図1とは異なりDBMSクラス情報を全てクライアント側でも保持する．これはサーバからクライアントへ移動した任意のDBMSインスタンスを扱うためである．Thread Managerについては，親スレッドのみが存在することとする．

5　今後の課題

今後は，既存の分散オブジェクト環境を適用・拡張することでDistributed Object Managerの部分を実現し，これを用いてC/S型のDBMSを実装する予定である．また永続的インスタンスのクライアントでの永続化により，長期トランザクションに対応する等の機能拡張について更に検討を進める予定である．

References

[1] M.Onizuka et al, "Object management of a visual data analysis tool," appears in TAPOS, vol.5, no.4.

[2] M.Shapiro et al, "Persistent and migration for c++ objects," proceeding of the ECOOP'89, pp.191-204, July, 1989.

[3] 原 隆浩他, "データベース移動を用いたATMネットワークにおけるトランザクション処理," 信学論D-I, vol.J80, no.6, pp. 505-513, Jun, 1997.