データベース・システム 84－8
（１９９１．７．１８）

ネットワークにおけるクライアント・サーバ型モデルの実現

広野 真吾　　橋本 朗　　草場 吉明

富士通ネットワークエンジニアリング㈱

コンピュータのダウンサイジングが進み、高性能のワークステーション（ＷＳ）もしくはパソコン（ＰＣ）を個人で使用できるようになった。それに伴い情報の処理形態は、メインフレームを中心に、情報の保有と処理を行う集中処理型から、各ＷＳ，ＰＣ上で処理を行う分散処理へ移行してきている。

ここでは分散処理に関連して、ネットワーク上の各ノード（ＷＳ，ＰＣ）に情報が存在し、他のノードからそれらの情報にアクセスするための手法について述べる。

Making a Database System aiming at Client and Server Model

Shingo Hirono　　Akira Hashimoto　　Yoshiaki Kusaba

Fujitsu Network Engineering Limited

Recently it became possible that we, individualy, use a workstation (WS) or a personal computer (PC). According to that, centralized data processing that data is stored and processed by mainframe has turned into distributed data processing by WS or PC.

In this paper, we discribe the way to have access to data of other node in network on which node (WS, PC) has data.

1. はじめに

近年のコンピュータ関連技術の急速な進歩により、個人で一台のコンピュータを占有できるようになってきている。ところが個人単位で情報を管理したくなる傾向にあり、結果として以下のような問題が新たに発生してきている。

・同一の情報が複数箇所で重複して所持されるために、情報の同一性が損なわれないように管理することが難しい。

・情報の所在が管理されないために、必要な情報がどこにあるかが分からない。

つまり、共通で使用する情報は集中して管理し、個人で使用する情報はローカルサイトで管理することが必要になってきた。

それらの問題を解決する手段として、ローカル・エリア・ネットワーク（ＬＡＮ）を用いて各コンピュータを有機的に接続する方法がある。ＬＡＮ上のネットワークＯＳによって、共通な情報はサーバ内に一元管理することで、上記の問題は解決される。

現在ＬＡＮ上で動いているアプリケーションは、ＬＡＮ上のサーバをファイル／プリンタサーバとしてしか使用していないものがほとんどである。これでは、ＬＡＮのもう一つのメリットである水平連携が生かされない。また、多数のコンピュータから同時にファイルの入出力要求を行うと、伝送路が飽和し性能が極端に悪化するという問題も発生する。

ここでは、コンピュータのＣＰＵ能力を有効に生かし、伝送路のトラヒックを減少させるのに有効な方法である、クライアント・サーバモデルを実システムに応用する手法を述べる。

2. クライアント・サーバモデル

2.1 クライアント・サーバモデルの概要

初期のＬＡＮの主な目的は、コンピュータ資源の共有にあった。すなわちプリンタ，ディスクなどを共有することで、装置の利用効率を上げることを目指していた。

上記の目的がほぼ達せられた現在、その次に実現しようとされているのが分散処理環境である。分散処理環境では、コンピュータが接続されたネットワークを、複数のプロセッサを持つ大きな一つのコンピュータとみなし、ネットワーク全体で処理の分担をおこない、全体のスループットを向上させることを目的とするようになってきている。

分散処理のひとつとして、ユーザ・インタフェースを取り持つフロントエンド部と、データの管理をおこなうバックエンド部に分離し、前者をローカル側に、後者をサーバ側においたものがクライアント・サーバモデルである（図１）。

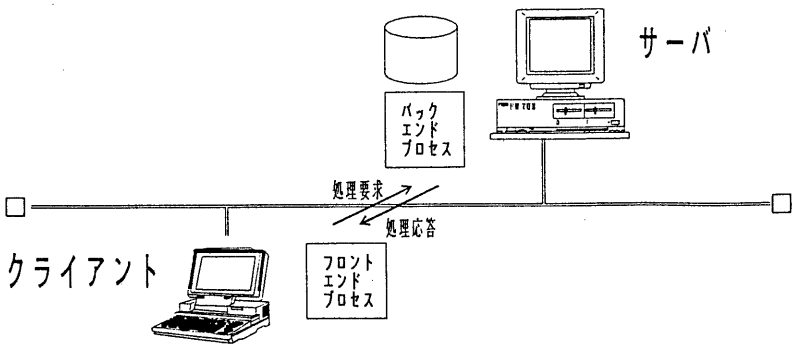

図1　クライアント・サーバモデル

2.2 クライアント・サーバモデルの特徴

クライアント・サーバモデルの特徴を以下にあげる。

・クライアント側のマシンの特長を生かしたユーザ・インタフェースが構築できる。
ユーザとのインタフェースはすべてクライアント側で処理するので、従来のダム端末等と比較して、マシンの特長を生かしたＭＭＩが作成できる。

・ファイル入出力が発生しないので、伝送路の負担が少ない。
ファイルアクセスを必要とする処理（検索等）は、バックエンドプロセスがすべて処理し処理結果のみをフロントエンドへ返すので、トラヒックが減少する。

・クライアント側は比較的パワーが少ないマシンでも担当できる。
負荷が大きいバックエンド部をサーバが担当するので、クライアント側の負荷は軽くなる。

このように、クライアント・サーバモデルはホスト集中型とスタンドアロン型の両者の長所を兼ね備えている。

2.3 プロセス間通信機能

クライアント・サーバモデルを実現するためには、ＯＳがプロセス間通信機能をサポートしていることが必要である。プロセス間通信機能とは、ＯＳ上に各プロセスから書込み／読み込みができる共有エリアを設けて、プロセス間で情報の送受信ができるようにしたものである。

3. オブジェクト・モデル

3.1 オブジェクト・モデルの特徴

筆者らはクライアント・サーバモデルを実現するために、オブジェクト・モデルを用いてシステムを構築した。それは以下の理由から、クライアント・サーバモデルにもっとも適しているからである。

・処理の同期が取りやすいので、ネットワークとの親和性が高い。

物理的に離れたコンピュータ間のオブジェクト同士でも同期が容易にとれる。具体的には、ＯＳに用意されているプロセス間通信用の機構（キュー）にイベントを書き込むことによって同期をとる。

・データがカプセル化されているので、セキュリティレベルが高い。

ネットワークで稼働すると、他のコンピュータからのアクセスも当然発生するので、スタンドアロンで動作するときよりも、データが破壊される危険性が増加する。その点オブジェクトは、あらかじめ設定されたメソッドでしかデータを操作できないので、誤ったアクセスを未然に防げる。

・独立性が高い（疎結合である）ので、分散処理を実現しやすい。

各オブジェクトは、他のオブジェクトと通信しながら処理を進めていくが、通信はメッセージ・パッシングによってのみ行うので、オブジェクト間の独立性は高い。つまり共有メモリ等を使用しないので、論理的に接続されているオブジェクトならば、自分のコンピュータの中にあるか外にあるかは問わないで通信ができる。

3.2 オブジェクトの概念

ここでいうオブジェクトとは、データ構造とそのデータ構造に対する操作をカプセル化したものである。オブジェクト内部のデータに対しては、実行可能な操作によってのみアクセスできる（図２）。

オブジェクトには、ネットワーク上で一意のノード識別子と、システム内で一意の識別子が付けられ、ネットワーク上の他のオブジェクトと区別される。

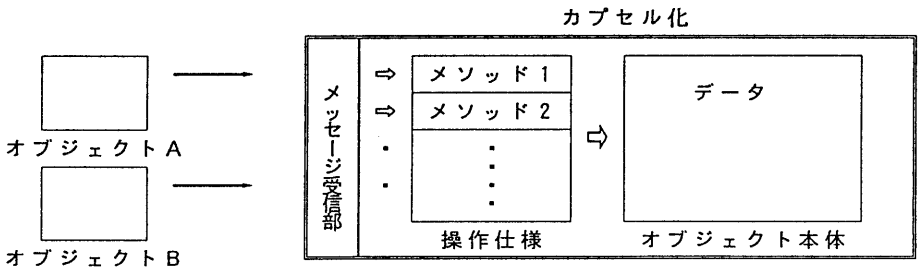

図２　オブジェクトの構造

3.3 メッセージ・パッシング

オブジェクト間の通信はすべてメッセージ・パッシングによっておこなわれる。通信の内容としては以下のものがあげられる。

①処理依頼（データ要求，書き替え要求，計算要求など）
②処理応答（上記応答）
③オブジェクトの起動／終了
④エラー通知

これらのメッセージは、キューを用いて送受信される。すなわち、オブジェクトはそれぞれ一つのキューを持ち、このキューにメッセージが書き込まれることで、イベントが発生したことを知ることができる（図3）。

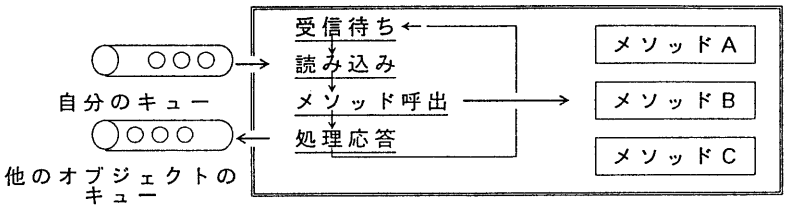

図3　キューを使った通信

4. 実システムへの応用とその効果

4.1 システム構成

筆者らが構築したシステムの構成を図4に示す。このシステムは、公衆網を用いた遠隔地からのアクセス、およびＬＡＮでのアクセスをサポートするシステムである。

このシステムでは伝送路が2種類存在するため、通信オブジェクトから受けたメッセージをネットワーク（ＮＷ）制御オブジェクトが、相手先端末に合わせて伝送路を選択し送信する。

オブジェクト管理部は、各オブジェクトの起動／終了を管理し、他端末からの起動要求も受け付ける。

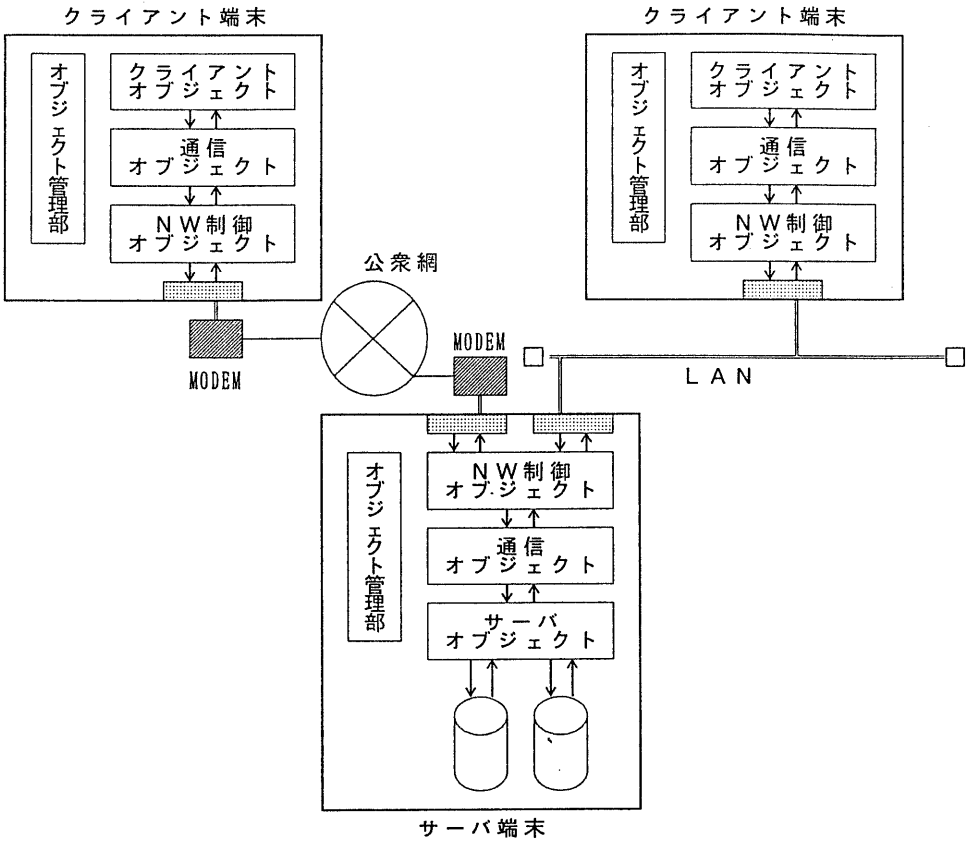

図4　システム構成

4.2 クライアント・サーバモデルの効果

クライアント・サーバモデルを採用して、実際に以下の様な効果が得られた。特にネットワークの種類を意識せずに通信が可能なのは、効果として大きい。

・ネットワークを意識しない処理

ネットワークの選択処理を通信オブジェクトが一括しておこなうので、通信の相手を意識しないシステム構築が可能となった。

・サーバマシンが不要

専用のサーバマシンを設けずに、すべてのコンピュータ内のデータにアクセスが可能となった。すなわち、サーバ専用プログラムは必要なく、自分のコンピュータで使用しているオブジェクトが他のコンピュータからも同様に使用可能であることを意味する。

5. おわりに

今回の開発ではクライアント・サーバモデルを実現することに主眼を置いて開発をおこなった。これから解決しなければならない課題がいくつも残っているが、特に以下の項目は重要である。

・データの透過的な分散

現在まだデータの読出側がデータの所在を知っていなければならない。これをデータの格納場所を管理するノードを設けることにより、データの所在を意識しないシステムの構築を目指す。

・処理の分散化

データの分散だけでなく、処理の分散をおこなう。つまり、個々の処理に適したＣＰＵをネットワーク上で捜し出して処理を依頼し、ネットワーク全体の処理効率を向上させる。

特に前者の課題は今後ますます重要になる機能であり、効果的なデータの管理方法を確立する必要がある。

参考文献

〔1〕Paul J. Fortier 著　『分散型オペレーティング・システムの設計』　日系ＢＰ社