三層クライアント／サーバ・システムにおけるミドルウエアの開発
（4）端末制御方式の実装

4F－6

吉川雅昭 相馬健志 太田幸子 塩澤博之 末廣亮太
松下電器産業（株） マルチメディアシステム研究所

1. はじめに

近年、クライアント/ サーバモデルに基づく分散処理システムの開発が活発に行なわれている。

このような中で我々は、端末として電話、FAX、パソコン等の多種多様なメディアのサポートが可能で、拡張性にすぐれた分散処理方式を検討し、データベース、サーバ、端末の三層クライアント/ サーバ・システム (以下三層モデル) のミドルウェアを開発した。

本発表では、特にこの三層モデルのサーバー端末間に焦点を当て、複数の端末を 1 つのサーバで制御する方法について、端末としてパソコンを使用した場合に関する検討を行なったので、報告する。

また、三層モデルを採用することにより、障害の発生が複雑になるが、本ミドルウェアでの解決方法を、障害発生箇所に分けて検討を行なったので報告する。

2. 対象とするシステム構成

図 1 に本ミドルウェアが対象とするシステムの構成を示す。

端末 (パソコン) は ISDN を介して RPC(DCE RPC) でサーバと接続されており、クライアントプロセスが動作する。サーバには、データベースをアクセスするプロセスと端末を管理するプロセスが動作する。サーバは、 LAN を介して Oracle SQL*Net でデータベースと接続されている。

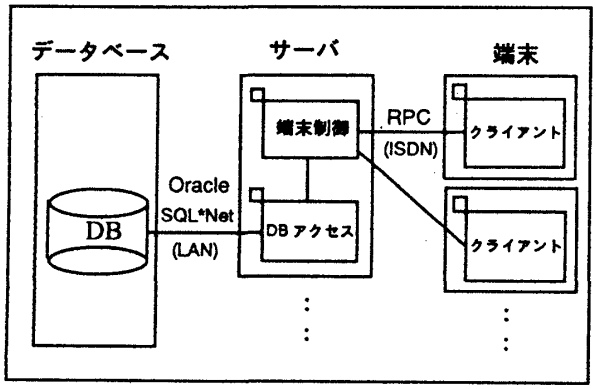

図 1 本ミドルウェアが対象とするシステム構成

3. 端末制御方式

複数の端末を管理するサーバ上で動作するプロセスの実装方式を検討するため、以下の 2 つについて検討し、端末制御方式を決定した。

・タスク管理方式 (マルチプロセス/ マルチスレッド)
・実行方式 (常駐方式/ オンデマンド方式)

3. 1 タスクの管理方式

複数の端末からの要求を並列に処理する方式としてマルチスレッド方式とマルチプロセス方式が考えられる。

マルチスレッド方式の場合、プログラムの性能が良く、 1 台のサーバで処理できる端末数は、マルチプロセス方式よりも多くできるがプログラムの作成が難しくなる。また、サーバ内で端末ごとの状態を管理するためには、端末の情報を管理する機構が必要となる。

一方、マルチプロセス方式の場合、プログラムの作成は容易になるが、性能面でマルチスレッド方式よりも劣る。また、プロセスを管理し、端末からの要求をプロセスに割り当てる機構が必要となる。

3. 2 実行方式

プロセスの実行方式として常駐方式とオンデマンド方式が考えられる。

常駐方式の場合、クライアントからの接続は、オンデマンド方式よりも速い。また、本方式の場合は、タスクの管理方式としてマルチプロセス方式と、マルチスレッド方式のどちらでも採用できるが、マルチプロセス方式の場合は、クライアントの処理要求を空いているプロセスに割り当てる機構が必要となり、マルチスレッドの場合は、マルチスレッドで使用可能なシステムコールを使うなど、処理の並列化を考えて作成する必要があ

Developing a Middleware in the Three Tier Client-Server System
(4) Implementation of Terminal Communication Control
Masaaki YOSHIKAWA, Kenji SOHMA, Sachiko OHTA, Hiroyuki SHIOZAWA, Ryota SUEHIRO
Matsushita Electric Industrial Co., Ltd.

る。

一方、オンデマンド方式の場合 (タスクの管理方式は、マルチプロセス方式となる)、クライアントとの接続は、常駐形式よりも時間がかかるが、接続時間がかかるのは、最初のコールの時のみ (クライアントからの要求ごとにプロセスを起動することは、接続に時間がかかり過ぎて現実的でない) である。また、起動されたサーバ上のプロセスは1つの端末からの要求を処理しているだけで、また毎回起動され直すので、障害対策が容易となる。

3.3 端末制御プロセスの構成

上記の検討結果より、プログラム作成および障害対策が容易であるマルチプロセス・オンデマンド方式を採用することにした。図2に端末を制御するプロセス (エージェント管理サーバとエージェント) の構成を示す。

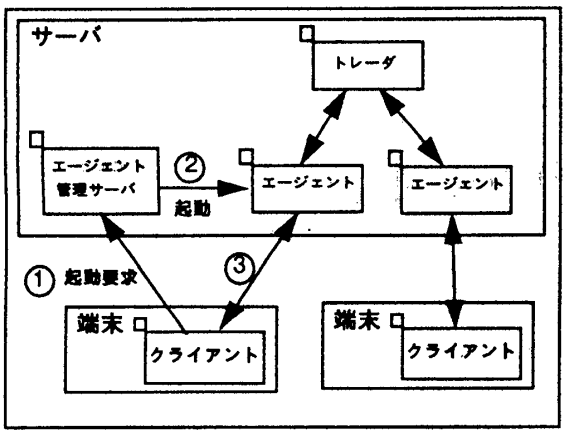

図2 端末制御プロセスの構成

エージェント管理サーバは、クライアントプロセス起動時にエージェントを起動する。

エージェント管理サーバは、エージェントを起動したクライアントの情報と共に、そのエージェントの情報 (バインディング情報等) を管理する。

クライアントプロセスは、終了するまで起動されたエージェントを使用して処理を行なう。

トレーダは、エージェントの要求を受けて、データベースをアクセスする。

なお、本方式の場合は、クライアントプロセス起動時に多少時間がかかり、マルチスレッド方式に比べて性能がやや劣るので性能評価を通して実用性の確認を行なうこととした。

4. 障害処理方式

図2で示した構成で、端末、サーバ、データベースおよびネットワークに障害が発生した場合の対処方法について述べる。

4.1 端末における障害

端末で障害が起こると対応するエージェントがそのまま残る。

このエージェントを処理するために、エージェント管理サーバは、エージェントを起動した端末のIPアドレスを管理しており、再度同一の端末からの接続が行なわれる際にエージェント管理サーバが旧エージェントを終了させてから、新規のエージェントを起動する。

また、エージェントは一定時間アクセスがないと自ら終了する。クライアントプロセスは、エージェントが存在しない場合は、エージェントを再起動して処理を継続する。

端末上のログ情報 (エラー、課金情報等) は, サーバにすべて転送し、サーバで一元管理する。

4.2 サーバにおける障害

サーバ (または端末－サーバ間のネットワーク) に障害が起こっても、クライアントプロセスは、自動的に代替のサーバに接続して処理を行なう機構をミドルウェアに組み込んだ。エージェント管理サーバに対する最初のRPCコール時に接続できなかった場合のみ代替のサーバに接続する。

4.3 データベースにおける障害

データベース (またはサーバ－データベース間のネットワーク) で障害が発生した場合でも、クライアントプロセスに制御を返すために、データベースに対して処理要求を発行するときには、エージェントはタイマを設定し、その処理がロックした場合でもタイムアウトするようにして、処理が完了しなかったことをクライアントプロセスに通知する機構をミドルウェアに組み込んだ。

4.4 ステートレスなエージェント

エージェントは、上記の障害処理で述べたようにクライアントプロセスが動作中に再起動されることがある。このため、エージェントはステータスを保持しないようにした。

5. まとめ

プログラム開発が容易で障害対策が容易である、マルチプロセス・オンデマンド方式で処理する三層モデル対応のミドルウェアを開発した。本ミドルウェアは、同一種類のエージェントが複数起動するシステム、および常駐型のエージェントを使用するシステムにも適応することができる。

今後は、マルチスレッド方式の対応と、クライアントプロセスからのトランザクション管理が可能なミドルウェアとして行く予定である。