統合化ファイルアクセス法（IFAM）のサーバ化

4B－9

鈴鹿 豊明　盛屋 邦彦
日立ソフトウェアエンジニアリング（株）

1　はじめに

ネットワークの普及につれワークステーションは相互に結合され、さまざまなアプリケーションがその上で稼働している。それらのアプリケーションではネットワークに結合されていることを活かして、相互にデータを共有するということが当然の前提となっているものが多い。

一方、筆者らは既存のアクセス法の統合とマルチメディアデータの効率的な操作・蓄積を目的に、統合化ファイルアクセス法（IFAM）を作成してきた[盛屋92] [古舘92]。IFAMは無限可変長かつ編集可能なレコード、柔軟な検索キー、集合操作などの特徴を持ち、さまざまなアプリケーションへの応用が可能である。

今回このIFAMをネットワーク上でクライアント・サーバ方式のファイルサーバとして利用できるよう拡張した。その際OSの提供するソケット、共有メモリ、セマフォを利用し、バッファ共有や、排他制御を実現した。

2　サーバ化概要

本システムは各IFAMシステムにサーバプロセスを実行し、各アプリケーションはネットワーク上の複数のワークステーションからそのクライアントとしてサーバにアクセスするクライアントサーバシステムとなっている。このときサーバは各アクセスの排他制御を行う。

IFAMでは2次記憶上の領域をブロックという固定長の単位に区切って管理している。入出力はこのブロック毎に行う。1つのブロックには1つのファイルに関する情報だけを持つものもあるが、複数のファイルの情報を持つものもある。すなわち1つのファイルの占有する領域の境界は、物理的な境界と一致していない。

このためIFAMでは“ファイル”のような論理的な単位だけでの排他制御ではうまくいかず、ブロックのような物理的な単位の排他制御が必要となる。

ところで、これらのことを実現するに当たりデータだけを共有化するのではなく、入出力バッファも共有化する必要がある。もしこれを共有化せず、各クライアントプロセス毎にバッファリングを行うことにすると、バッファ内での更新が相互に反映されずそれぞれのバッファ間でデータの一貫性が破壊されてしまう可能性があるからである。

3　サーバ化の実現

バッファリングは空いているバッファをリストにして管理するフリーリストと、ブロック番号からバッファを効率良く求めるためのハッシュテーブルと、ブロックをバッファ中に残す優先順位を管理するLRUリストからなる管理部分及びバッファ本体から構成される。次節以降、バッファとその管理部分であるバッファ管理データを区別して呼ぶ。

3.1　IFAMクライアント・サーバ・システム

図1はIFAMクライアント・サーバ・システムの概略を表している。アプリケーションはIFAMのクライアントライブラリをリンクし、デーモンで実行されているIFAMサーバにソケットを使用してIFAM操作のリクエストを送る。このデーモンは受付専門であり、リクエストを受けつけた後forkして子プロセスを作り、実際の処理はそちらで行なう。forkされた子プロセスは共有メモリを利用したバッファ管理、及びバッファを通して入出力を行いそれぞれのアプリケーションの要求に応じる。

3.2　バッファ管理データの共有化

バッファ管理のデータ構造は、相互に密接に関係しあっているということと、極短時間にその処理が終了することから全体を排他制御の単位とすることとした。この実現には単純にOSの提供するセマフォを利用する。

サーバ化に伴い「全てのバッファが使われてしまい、バッファ・フリー・リスト、LRUリストが共に空になってしまう可能性がある」という問題が

Enhancement of Integrated File Access Method to a Server
Toyoaki SUZUKA, Kunihiko MORIYA
Hitachi Software Engineering Co., Ltd.

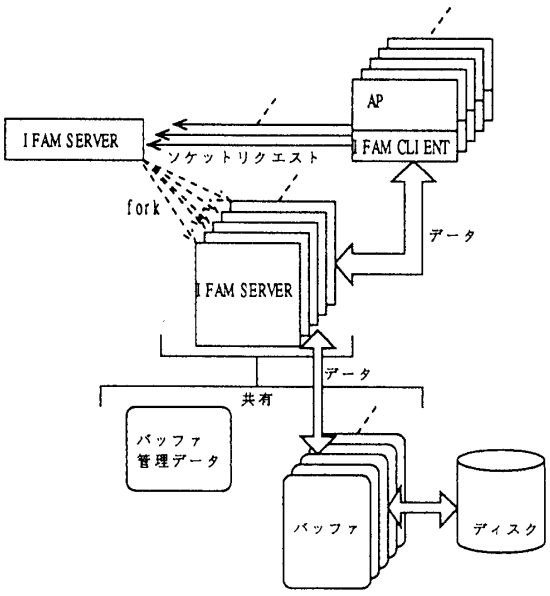

図 1: IFAM クライアント・サーバ・システム概略

発生した。単一プロセスによるIFAMでは、同時に2面以上のバッファを確保しないように設計してあったのでこの問題は起こらなかった。サーバ化後も個々のプロセスはそうなっているが、クライアントプロセス数がバッファ面数を超えてしまえば、当然バッファ不足が起こり得る。バッファ不足が起きた場合、バッファを要求しているクライアントプロセスを“待ち”にしなければならない。このために次の2つの処理方式を考案した。

1. LRU リストとバッファ・フリー・リストの和と同じ値を持つセマフォを用意し、このセマフォが0の時に待ちになる。
2. IFAM サーバの子プロセスの数をバッファの面数以下にする。

1. の方式では、バッファ管理データの操作中にセマフォの操作をしなくてはならない。つまり、バッファ管理テーブルをロックしている最中にセマフォを操作しているため、もし待ちになってしまった場合、他のプロセスがバッファを解放できない。したがって、2. の方法を採ることにした。

3.3 バッファの共有化

バッファ管理データへのアクセスは必ず更新を伴っていたため、一時に1プロセスだけをアクセス可能としていたが、バッファへのアクセスは参照だけの場合もある。このため一時に1プロセスという制御は制約のしすぎとなる。そこで、参照のみのプロセスはいくらでも（前節で述べた制約によりバッファの面数まで）アクセスを許し、更新の伴うプロセスの場合にのみ、参照のプロセスを含む他の一切のアクセスを禁止することにした。このためのアルゴリズムを図2に示す。

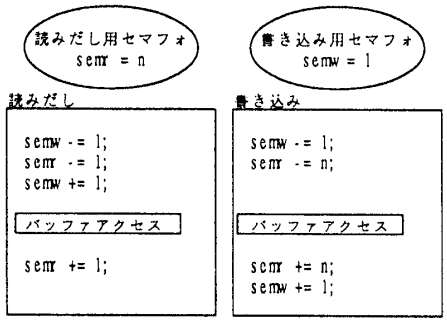

図 2: バッファ排他制御のアルゴリズム

3.4 ファイルの排他制御

これまで述べてきたような物理的な排他制御によりIFAMシステムとしたデータの一貫性を保つことは可能となるが、アプリケーションのデータとしての一貫性を保つことはできない。そこでファイル毎の排他制御の機能を導入した。これはファイルアクセスにreadとwriteのモードを設け、バッファの共有化と同様の制御にデッドロック防止のためのタイムアウトの機能を加えて実現した。

3.5 むすび

本稿はマルチメディアデータに対応するための新しい機能を備え、かつ従来のファイルアクセス法の構造／機能を包含したIFAMについて、そのサーバ化の手法を述べた。OSの提供する機能を用いることにより、物理的なデータ構造及びファイルに対しての排他制御を実現している。

参考文献

[盛屋 92] 盛屋、鈴鹿: 統合化環境を目指したファイルアクセス法の提案, 情報処理学会第44回全国大会, 1992.

[古舘 92] 古舘、鈴鹿、吉田、盛屋: 統合化ファイルアクセス法（IFAM）の実現手法, 情報処理学会第45回全国大会, 1992.