HiTactix/Symbioseの開発（3）
—Virtual File Systemの設計—

2Z－3

中野 隆裕，岩嵜 正明，中原 雅彦，竹内　理，児玉 昇司，川田 容子，田口 しほ子
（株）日立製作所システム開発研究所

1. はじめに

HiTactixでは，現在，QoS保証を施したファイルシステム（SMFS[1]）をはじめ，複数の異なるファイルシステムが利用可能である．ただし，現状，各ファイルシステムは，それぞれ独自の入出力I/Fを提供し，入出力I/Fが不統一である．HiTactix/Symbioseの開発により，ウインドウ・システムを利用したデバッグなどが可能となり，仮想端末やパイプを含む，複数の異なるファイルシステムを統合し，アプリケーションに統一的なファイルアクセス手順を提供するVFS（Virtual File System）が必要となった．

本研究は，QoSを保証するアプリケーションに対し，SMFSが提供するQoS保証入出力機能を利用可能にするとともに，QoSを保証しないアプリケーションと混在して仮想端末を利用可能にするVFSについて検討する．本稿では，その基本設計について記述する．

2. HiTactixのリアルタイム入出力処理

QoS保証を施した連続メディアの入出力処理の実現には，大容量のデータ入出力を低オーバヘッドにて実現する機能と，周期的な入出力処理の実行機能が必要である．

HiTactixは，ユーザデータ領域に対し直接DMA転送を行い，大容量のデータ入出力を低オーバヘッドにて実現するダイレクト・バッファ・マッピング（以後DBM）機構[2]を備えている．ただし，アプリケーションは，入出力デバイスとのアクセス競合を回避するため，入出力の実行が完了するまで入出力中のバッファ領域へのアクセスが禁止される．

HiTactixでは，起動周期Tと，1周期あたりのCPU時間Lが保証された周期スレッド[3]を用いて周期的な入出力処理を実現できる．周期スレッドは，1周期あたりCPU時間L分の時間枠が割り当てられる．周期スレッドは，その時間枠中，他の処理(割り込み処理を除く)に優先してCPUが割り当てられ，処理が実行できる．周期スレッドは，1周期分の処理を終えると，CPU解放関数を呼び出す．周期スレッドは，時間枠を超えてもCPUを解放しなかった場合，デッドラインミスとなり，次の周期まで処理が中断される．

周期スレッドは，一般的な入出力I/Fである同期入出力I/Fを利用すると，入出力の完了まで実行がブロックされるため，時間枠を越える可能性が高い．さらに，同期入出力I/Fを利用すると，周期的に複数の入出力を扱うアプリケーションの実現が困難となる．

これを解決するため，HiTactixの各ファイルシステムでは，入出力処理を起動処理と実行完了確認処理に分離する非同期入出力I/Fを提供している．非同期入出力I/Fを利用するアプリケーションは，入出力の起動処理を行った後CPUを解放し，次の周期にて入出力の実行完了を確認する．さらに，複数の入出力起動処理が実行可能であり，周期的に複数の入出力を扱うアプリケーションが，容易に実現可能となる．

3. HiTactix用VFS I/Fの設計

HiTactix用のVFSでは，一般的なVFSの機能，即ち，入出力I/Fの統一や，名前空間の一元化，仮想端末機能の追加に加え，連続メディアのQoSを保証する入出力処理を可能にする必要がある．

このため，2節にて示したHiTactixのファイルシステム同様，VFSにも非同期入出力I/Fを用意する．即ち，以下の関数をHiTactix用VFSの非同期入出力I/Fとして追加する．

```
int pread_async(fd, buf, size, pos, *id);
int pwrite_async(fd, buf, size, pos, *id);
int check_io_done(fd, id, flags);
```

関数pread/pwrite_asyncは，XPG4.2[1]に定義される関数pread,pwriteに入出力要求を識別する非同期入出力IDへのポインタidを引数に追加した関数である．また，関数check_io_doneは，非同期入出力IDに対応する入出力の実行完了確認を行う関数であり，flagsに，同期か非同期かを指定できる．

4. HiTactix用VFS 内部構造の設計

同期入出力I/Fに加え，非同期入出力I/Fを利用可能と

A Development of HiTactix/Symbiose (3)
- A Design of Virtual File System -
Takahiro NAKANO, Masaaki IWASAKI, Masahiko NAKAHARA, Tadashi TAKEUCHI, Syouji KODAMA, Yoko KAWATA, Shihoko TAGUCHI.
Systems Development Laboratory, Hitachi, LTD.

[1] X/Open Portability Guide Issue 4.2

すると，両者が混在する場合のQoS保証を検討する必要が生じる．たとえば，仮想端末では，非周期スレッドが同期入出力I/Fを用いた出力処理中に，CPUの割り当てが周期スレッドに移り，この周期スレッドが非同期入出力I/Fを用いて出力処理を要求する場合が発生する．

この場合，図1に示す従来のVFSでは，入出力処理をファイルシステムに要求する前にロックを獲得し，先行するスレッドの出力が完了するまで，後続するスレッドの実行をブロックしていた．

ブロックする時間は，出力要求の量によって異なるため，時間枠内に周期スレッドの処理が完了するか否かを保証できない．この方式は，ロック競合のため，出力要求時に周期スレッドの時間枠をオーバーする危険があり，周期スレッドから利用できない．周期スレッドから仮想端末を利用可能にするためには，ロック機構を排除する必要がある．

HiTactixのVFSでは，この問題を解決するため，ファイルシステムへの処理要求をメッセージ化し，そのメッセージを，ファイルシステムの処理要求キューに接続する処理要求逐次化方式を用いる（図2）．ファイルシステム内に逐次スレッドを設け，この処理スレッドが処理要求キューからメッセージ化された処理要求を順に取り出し，要求された処理を代行し，処理の実行完了を要求スレッドに通知する．この方式では，各スレッドが処理要求キューにアクセスする際に，細粒度プリエンプト制御[4]によって，スレッド間の排他制御を行う．これにより，上記のロック競合による優先度逆転を回避できる．

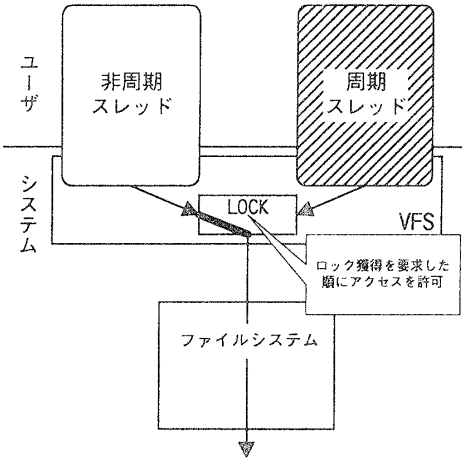

図1　従来のVFS構成図

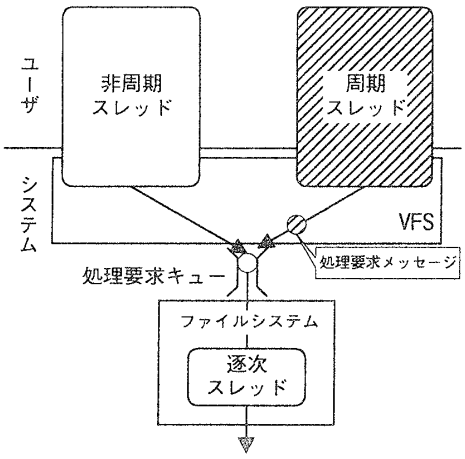

図2　HiTactixのVFS構成図

この方式では，逐次スレッドが出力処理中であっても，周期スレッドは処理要求キューに処理要求を接続し，即座に処理に復帰することができる．これにより，仮想端末の出力処理が実行完了までに数周期分の時間を費やす場合でも，周期スレッドは，実行が完了するまで毎周期，実行完了確認を行うことで，デッドラインミスを起こすことなく，他の処理（たとえば音声データの送信処理など）を実行し続けることが可能となる．

上記VFSの処理要求逐次化方式では，同期入出力I/Fを利用する場合，VFSは入出力処理の実行完了が通知されるまで，アプリケーションの実行をブロックする．非同期入出力I/Fを利用する場合，アプリケーションは，処理実行完了が通知されているか否かを処理実行完了確認によって検査する．

5. まとめ

HiTactix用のVFSの設計検討を行った．周期スレッドを用いた連続メディアの高品質な入出力処理を実現するには，入出力の低オーバヘッド化，非同期入出力I/Fの追加，ロック機構の排除が必要であることを示した．この要件を満たすため，HiTactix用のVFSでは，DBMの対応，非同期入出力I/Fの追加，処理要求の逐次化を行うことを示した．今後，本方式の実装を進め，実機による検証を行う予定である．

参考文献

[1] 児玉他，「HiTactix/Symbioseの開発(4)―ストライプト・メディア・ファイルシステム―」，情報処理学会第59回全国大会論文集，Sep. 1999.

[2] 中野他，「連続メディア処理向きマイクロカーネルの開発(4)－入出力方式の設計と評価－」，情報処理学会第53回全国大会論文集(1)，pp 147-148，Sep. 1996.

[3] 竹内他，「連続メディア処理向きOSの周期駆動保証機構の設計と実装」，情報処理学会論文誌，Vol. 40，No. 3，pp 1204-1215，Mar. 1999.

[4] 中原他，「連続メディア処理向けマイクロカーネルにおける内部排他制御方式」，情報処理学会論文誌，Vol. 40，No. 6，pp 2635-2644，Jun. 1999