システムソフトウェアと
オペレーティング・システム 64−10
マルチメディア通信と分散処理 65−10
（1994. 5. 19）

実時間オペレーティングシステムにおける連続メディアファイルサーバーの実験†

多田征司‡

横河ヒューレットパッカード株式会社

あらまし　著者は，音声やビデオのような「連続メディア」を取り扱うファイルシステムの研究開発を行ってきている．連続メディアファイルシステムを構築するにあたり実時間マイクロカーネルを基盤技術として用い，ユーザレベルサーバ群によって連続メディアアプリケーションをサポートしていく実装モデルを検討している．この実装モデルの有効性を検討するにあたって実時間オペレーティングシステム Real-Time Mach 3.0 マイクロカーネル上でビデオファイルを再生するファイルシステムを試作し評価実験を行った．本稿では，試作システムの実装と評価実験の結果について報告する．

キーワード　実時間オペレーティングシステム，マイクロカーネル，連続メディア，ファイルシステム，X ウインドウシステム

Experiments with Continuous Media File Server in Realtime Operating System

Seiji Tada

tada@sfc.keio.ac.jp, Yokogawa Hewlett Packard Ltd.

Abstract　The author has been developing a file system which supports continuous media data such as audio and video. Our approach is to construct the continuous media file system as a group of user-level servers on top of realtime microkernel. The author implemented a QuickTime file server on Real-Time Mach 3.0 microkernel and performed some performance evaluation. In this paper, implementation and evaluation of the experimental system are described.

Keyword　Realtime Operating System, Microkernel, Continuous Media, File System, X Window System

†この研究は，情報処理振興事業協会 (IPA) が実施している開放型基盤ソフトウェア研究開発評価事業「マルチメディア統合環境基盤ソフトウェア」プロジェクトのもとに行なわれた． ‡ 開放型基盤ソフトウェア湘南藤沢キャンパス研究室の研究員として IPA に登録されている．

1 はじめに

アプリケーションの要求やハードウエアの規模に応じて構成を容易に変更できるような柔軟なシステムの研究が行われてきている．マイクロカーネル技術を採用し必要なコンポーネントを組み合わせることによってシステムを動的に再構成する技術や，ユーザーレベルスケジューラなどの採用により必要なリソースのスケジューリングポリシーをユーザーレベルで選択することができる技術などが注目されている．このようなシステムでは，近年注目されている情報携帯端末(PDA)のような比較的小規模なハードウエアから高速のワークステーションあるいはスーパーコンピュータに至るさまざまなシステムキャパシティに対応することができるほか，アプリケーションの要求に応じて動的にスケジューリングポリシーを選択することができ柔軟性に富むシステム構成が可能となる．

著者は，このようなシステムにおいて柔軟なシステム構成の利点を保ちながら，ビデオやオーディオデータといった連続メディア(Continuous Media)を扱うためのシステムアーキテクチャの研究を行っている．ここで，われわれは，実時間マイクロカーネルである Real-Time Mach3.0 [2] を基盤技術として採用し，マイクロカーネル上で互いに協調して動作する複数のユーザレベルサーバ群によって連続メディアアプリケーションをサポートするモデルを考えている．[3]

われわれは，まずローカルなディスク上に置かれたビデオファイルをウインドウシステム上で再生する簡単な連続メディアアプリケーションを試作し評価することによって，ユーザレベルサーバ群による実装モデルの有効性について検討することから着手した．本稿では，試作したシステムの実装と評価実験の結果について述べる．

2 プロトタイプの設計と実装

2.1 プロトタイプの目的

従来のファイルシステムでは，ハードディスクのアクセスの際のシーク時間等のばらつきに伴うアクセス遅延のばらつき(delay variation)のため，連続メディアにそれを適用した場合，記録／再生時に動画や音声の途切れが起こるという不都合が生じてしまう問題点があった．著者らは，アプリケーションのアクセスレートに基づいてデータのプリフェッチを行うことによりこの問題を解決する方針で研究開発を進めている．

連続メディアの記録／再生においては，データの持つ論理的な時間軸と同期しながら処理を行う「同期的処理」が不可欠である．一方，データプリフェッチは，システムの並列性を向上させるために「非同期的処理」で実現されることが望ましい．

そこで著者は，以下の二点を目的としてプロトタイプを試作することにした．

- Mach 3.0 が提供する外部ページャインターフェースを用いてプリフェッチ戦略に基づくメモリマネージャを試作し，その有効性を検討する
- 「同期的処理」と「非同期的処理」が混合した連続メディアセッションをユーザレベルタスク群で構成し，その問題点について検討する

2.2 システム概要

試作システムは，クライアントサーバ方式で Apple 社の QuickTime[4] ファイルをウインドウシステム上で再生するもので，メモリマネージャ，アプリケーション，拡張 X ウインドウサーバ，の3つのユーザレベルタスクから構成される．図1に試作システムの概要を示す．

メモリマネージャは，ハードディスクに格納された QuickTime ファイルをメモリマップトファイルとしてアプリケーションに提供する．アプリケーションは，QuickTime ファイルを仮想空間にマップしてアクセスし，拡張 X ウインドウサーバに画像フレームの描画を依頼する．このとき，実時間マイクロカーネルが提供する周期スレッドを用いて一定周期でファイルアクセスとフレーム描画を繰り返していくことにより動画ファイルの再生を行う．メモリマネージャでは，データプリフェッチのためのスレッドと外部メモリマネージャ(EMM)通信のためのスレッドの二つのスレッドがクライアントごとに割り当てられる．連続メディアのデータストリームは，メモリマネージャ内の二つのスレッドとアプリケーションの周期スレッド，および X サーバのディスプレイスレッドの計4つのスレッドから構成されることになる．EMM スレッドとディスプレイスレッドは，アプリケーションの周期スレッドの要求に応じて動作する．すなわち，EMM スレッド，アプリケーションスレッド，ディスプレイスレッドの三者はメディアデータの時間制約にしたがって同期的に動作する．一方，プリフェッチスレッドはこれらのスレッド群に対して非同期的に動作する．

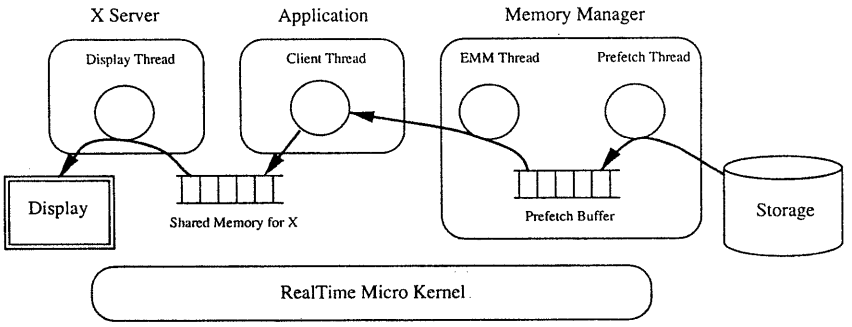

図 1: システム概要

以下では，システムを構成する各コンポーネントについて述べる．

2.3 メモリマネージャ

メモリマネージャは，QuickTime Movie ファイルパーサを備えており， Movie ファイルに記述された動画フレームのインデックス情報を用いてアプリケーションに先行してデータのプリフェッチを行う「知的な」ページャとして振る舞う．

メモリマネージャは，Mach の外部ページャインターフェースを用いて実装されており，アプリケーションは，Movie ファイルをメモリオブジェクトとして自身の仮想アドレス空間にマップしてアクセスを行う．外部ページャインターフェースによるページング処理の概略を図 2に示す．

アプリケーションの仮想空間でページフォールトが発生すると，カーネルは当該メモリオブジェクトを管理しているメモリマネージャにページ要求を行なう．メモリマネージャは，外部ページャインターフェースを用いて，要求されたページをカーネルに供給する．このとき，メモリマネージャにおいてデータのプリフェッチを行うことにより，ディスクアクセスに伴う遅延を低減し，データ参照時遅延によるジッタを抑制することができると考えた．プリフェッチは，アプリケーションのアクセスパターンとアクセスレートに応じて行われる必要がある．試作システムでは，簡単のため，アプリケーションのアクセスパターンは Movie ファイルに格納されたすべてのビデオフレームをシーケンシャルにアクセスするものと仮定してプリフェッチを行うようにした．

メモリマネージャとアプリケーションとの間のインターフェースは，MIG(Mach Interface Generator) を用いて実装されており，Movie ファイルのオープン／クローズなどのインターフェースを提供している．アプリケーションは，ファイルオープン時に動画再生に必要な情報（フレームサイズなど）を取り出すことができるようになっている．

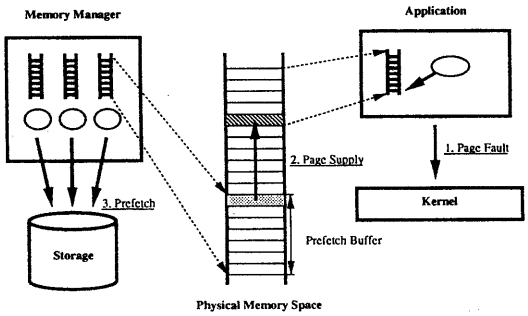

図 2: メモリマネージャ

2.4 アプリケーション

アプリケーションは，メモリマネージャが提供する QuickTime Movie ファイルをマップしてアクセスし，周期スレッドを用いて画像フレームを一定フレーム周期で描画する．画像フレームの描画は，拡張 X ウインドウサーバに描画依頼を行うことでなされる．描画依頼は，

XPutImage 関数と XSync 関数の組合せを用いて行い，XSync によって X サーバにおける処理の終了を待ち合わせることにより，アプリケーションと X サーバ間で描画処理に関する同期をとっている．

2.5 拡張 X ウインドウサーバ

ウインドウシステムには，X ウインドウシステムを改良したものを用いた．X ウインドウシステム (XFree86 ver1.3) を改良してクライアントサーバ間通信を Mach IPC と共有メモリによって実装し，プロセス間通信におけるデータコピーを低減することにより，描画処理にかかるコストを低減させた．X クライアントは，X サーバとのコネクション確立時に 64 Kbyte の共有メモリ領域を確保し，これを X プロトコルの通信路として用いる．共有メモリ領域に対するアクセス初期の遅延を抑制するために，共有メモリ領域を実記憶空間にワイアダウンするようにした．X クライアントは，X プロトコルを共有メモリ領域へコピーした後，Mach IPC を用いて X サーバに対してメッセージを送信することによって処理を依頼する．X サーバからのリプライメッセージやイベントの受け取りは，Mach IPC によって直接なされる．

3 評価実験

ここでは，前節で述べた試作システムの評価実験の結果について述べる．

本実験では，プリフェッチ手法によるアクセスレート保証機構の評価，および，連続メディアセッションを複数のユーザレベルタスクで構成した場合のスレッド間の資源調停の問題の評価をその主な目的としている．

実験では，拡張 X サーバ，メモリマネージャ，クライアントプログラムの三者を走行させて実験を行った．ウインドウマネージャは走行させていないが，これは，装飾ウインドウの生成やウインドウのリマッピングなどの処理が混入することを防ぐためである．

Intel i486DX2/66MHz[1] を搭載した IBM PC AT[2] 互換機 Gateway 2000 4DX2-66V で 250 ナノ秒精度のタイマボードを用いて描画処理に関連する種々のコスト測定とタイミング測定を行った．実験には，RT-Mach MK83, UNIX サーバ UX39 を使用した．

[1]Intel, i486DX はインテル社の商標です．
[2]IBM, Personal Computer AT は IBM Corp. の商標です．

実験に用いた QuickTime ファイルは，画像サイズが 160 × 120 ピクセル，8bit256 色カラー，総フレーム数 71 フレームで，SCSI ハードディスクに格納されている．このファイルを毎秒 10 フレームで再生させる実験を行った．

3.1 スレッド間の資源調停

3 種類のスケジューリングポリシーについて実験を行い，その性能を比較した．表 1に，各スレッドに付与したプライオリティを示す．

	Display	Client	EMM	Prefetch
ポリシー A	Time Sharing			
ポリシー B	High	Low	High	Low
ポリシー C	High	High	High	Low

表 1: スケジューリングポリシー

このとき，プリフェッチバッファサイズは，約 10 フレーム分を用意した．バッファ充填量のトレースとプリフェッチスレッドのスケジューリングのタイミングの監視結果からいずれのポリシーにおいても再生中にプリフェッチが有効に働いていることを確認した．

一画像フレームについて，ファイルアクセスから描画終了までにかかった経時コストを測定した．結果を図 3 に示す．

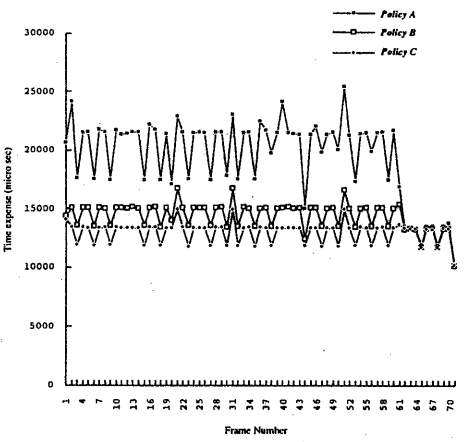

図 3: フレーム描画の経時コスト

ポリシー A の場合にはコストが大きく，またその大き

さも安定していないのに比べて，ポリシー C では，平均コストが三分の二程度にまで低減され，そのコストも比較的安定していることがわかる．[3]

ポリシー A (時分割ポリシー) においてコストが高く不安定である理由は，非同期に動作するプリフェッチスレッドが他のスレッドを邪魔していることが主な原因であると考えられる．ポリシー B では，クライアントスレッドのみがプリフェッチスレッドに邪魔されることになり．これがポリシー C にくらべると若干の性能低下として反映されている．

最後の 10 フレームを描画する際には，いずれのポリシーにおいてもほぼ同じコストで終了していることがみてとれる．これは，プリフェッチスレッドが最終フレームの読み込みを終えてその処理を終了したためであり，ポリシー間のコストの差はプリフェッチスレッドの干渉によるものであることを裏付けている．

ここで，描画コストの内容をさらに詳しく調べてみた．本試作システムでは，一フレームの描画は以下のステップを経て行われる．

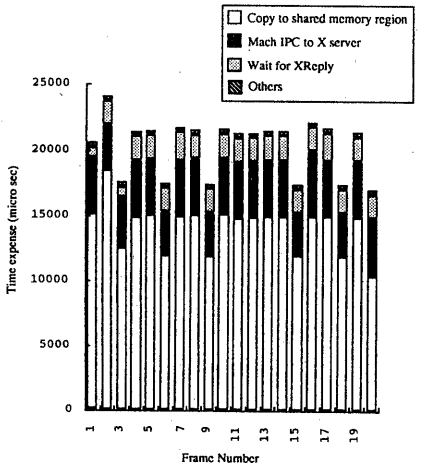

図 4: 経時コストの内訳（ポリシー A）

- X サーバとの共有メモリ領域へのフレームデータのコピー (ここにはページング処理の遅延も含まれる)
- Mach IPC による X サーバへの処理依頼
- 描画終了による X サーバからのリプライメッセージの待ち合わせ (XSync によるクライアントサーバ間の synchronization)

各ステップに要した時間をフレームごとに測定した．図 4，図 5に，ポリシー A とポリシー C の場合のそれぞれにおける経時コストの内訳を示す．

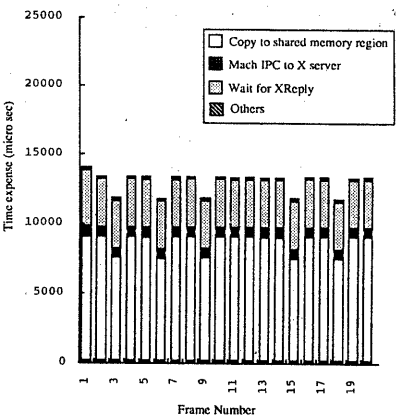

図 5: 経時コストの内訳（ポリシー C）

ポリシー A では，全体のコストが高くなっているほか，特にクライアントから拡張 X サーバへの IPC にかかる時間が非常に大きくなっていることがわかる．これは，IPC によるスレッドスイッチのタイミングで他のスレッドがスケジューリングされてしまうことによるものである．

アプリケーションのクライアントスレッド，メモリマネージャの外部ページャスレッド，X サーバスレッドの三者は同期的に動作しているが，それらに対して非同期的に動作しているプリフェッチスレッドは，セッションを構成する他のスレッドと干渉するおそれがあり，スレッドプライオリティの選択によっては，予測できない遅延をひきおこすことがわかった．これらの実験結果は，セッションを構成する各サーバ内のスレッド間のプライオリティ管理が重要であることを示唆している．

[3]ポリシー C においても定期的なコスト変動がみられる．これは，フレームデータ境界とページ境界は一致していないため，フレームによってフォールトページ数が異なってくるためである．メモリマネージャでのページング処理回数とフレームデータのアドレスを追跡することによりこれを確認した．

3.2 プリフェッチの効果

前述のポリシー C において，プリフェッチバッファのサイズを変えて実験を行った．プリフェッチ量を 0 とした場合には，アクセス遅延によるデッドラインミスが生じ．動画再生時のジッタとして観測された．本実験の範囲では，2 フレーム程度のバッファサイズでジッタを抑制することが可能であることがわかった．複数セッションを同時に開いた場合の性能評価も行ってみたが，この場合シーク遅延が増大するため，例えば 2 セッションの場合には，バッファサイズは各セッションにつき 5 フレーム以上を要することがわかった．

4 問題点および今後の課題

このほか，30 fps にフレームレートを上げた場合の実験も行った．ポリシー C では，30 fps でも問題なく動作したが，時分割ポリシーでは，システムの他のアクティビティに邪魔されるなどの影響により，デッドラインをミスする場合がでてくる．このとき，いったんデッドラインをミスすると，クライアントのアクセス周期が短くなる現象がみられた．Real-Time Mach 3.0 における周期スレッドでは，デッドラインミスが生じると遅延を取り戻そうとして周期スレッドのスケジューリングが優先され，スケジューリング周期が短くなる．このため，アプリケーションの実質的なフレームレートが増加することになる．この状況は，堆積した遅延が回復するまで継続する．フレームレートがディスクアクセスのバンド幅を超過すると，プリフェッチバッファはやがて枯渇してしまい，セッション全体が破綻をきたしてしまうこともあった．これについては，リアルタイムスケジューラを改良し，デッドラインミスの負債をためない周期スレッドセマンティクスをサポートすることで対処することを検討している．[5]

その他の問題点としては，メモリマネージャとカーネルの間のデータ転送コストが高いことがあげられる．データ転送をページ単位で行った場合，160 × 120 ピクセル，8 ビットカラーの動画データで，1 フレームあたり 4 回から 6 回のページ転送が行われる[4]．メモリマネージャとカーネル間のページ転送のコストは数百マイクロ秒と比較的大きいこと，ページ転送の回数分だけメモリマネージャとアプリケーションとの間のタスクスイッチングが発生すること，などから，フレームサイズが大きくなるにしたがってそのコストは深刻な問題となってくる．Mach 3.0 の外部メモリマネージャインターフェースでは，複数ページを一括して転送することもできるようになっており，これを利用して転送コストの低減を目指すことを考えている．

[4]回数はフレームデータとページ境界の関係に依存する

5 おわりに

アプリケーションのアクセスパターンに基づいてプリフェッチを行う連続メディアファイルサーバのプロトタイプを実時間オペレーティングシステム Real-Time Mach3.0 上で実装し，複数のユーザレベルサーバ群からなるシステムの性能評価を行った．

試作システムのように，連続メディアの再生処理に関わる「同期的処理」とプリフェッチのための「非同期的処理」が混在するシステムでは，両者のプライオリティ管理がメディアの時間制約を満たすうえで重要であることがわかった．

謝辞

本研究を行うにあたりご討論いただいた開放型基盤ソフトウェア研究開発評価事業「マルチメディア統合環境基盤ソフトウェア」プロジェクトの皆様に感謝致します．また，御指導いただいている慶應義塾大学環境情報学部の斎藤信男教授，徳田英幸助教授，萩野達也助教授に感謝致します．

参考文献

[1] 徳田, 斎藤: “マルチメディア統合環境プロジェクトにおけるリアルタイム処理技術,” 情処研報, Vol. 92, No. 20, 93-ARC-99, pp. 9–15 (1993).

[2] H. Tokuda, T. Nakajima and P. Rao: “Real-Time Mach: Towards a Predictable Real-Time System,” Proc of USENIX Mach Workshop, pp. 73–82 (1990).

[3] 西尾: “Real-Time Mach 3.0 上の連続メディア処理のための協調サーバ群の設計と実験,” 情処研報, Vol. 94, No. 32, 94-OS-63, pp. 57–64 (1994).

[4] “Inside Macintosh: QuickTime”, Apple Computer, Inc. Addison-Wesley, 1993.

[5] 河内谷, 緒方, 徳田: “連続メディア処理のためのリアルタイムスレッドモデルの拡張,” 第 48 回情処全大論文集 (1H-1), (1994).