1G−4 Machにおけるマルチメディア／リアルタイム拡張

矢崎 昌朋 中島 淳 松本 均
(株)富士通研究所

1. はじめに

近年、アプリケーションは、周辺デバイスの進歩に伴い、さまざまなメディアを用いたものが考えられている。しかし、オペレーティングシステム（OS）については、これらのデバイスを充分に使いこなせるだけの性能を持っていない。そこで、我々は、最新のハードウェア技術に対応したMachを使って、マルチメディアを扱うための拡張を行なった。本稿では、アニメーション、音声、音楽などの時間的に連続な情報を持つメディア（continuous media）についての機能拡張と、メディア間の同期制御法について述べる。

2. マルチメディア／リアルタイム拡張の概要

2.1 マルチメディアOSのリアルタイム性

連続メディアをサポートするためには、OSにリアルタイム性が必要である。それは、連続メディアの再生が、周期的な処理を繰り返すことによって実現されており、正確に周期的な処理が実現できないと、メディアの内容を正確に伝達できなかったり、品質の劣化を引き起こすためである。

周期的な処理に対する時間的な制約については、メディアを実現するデバイスと深く関係しているため、メディア毎に異なる。また、これらのメディアは、人間の感覚を相手にしているため、視覚に関係しているメディアと、聴覚に関係しているメディアの違いによっても異なる。

本拡張は、このようなメディア毎の時間的な制約の違いを取り扱えるように実現した。

2.2 同期制御法

複数のメディアの同期を制御する方法については、単一の制御構造で複数のメディアの同期を実現するよりも、メディア別に制御構造を用意し、それらの間で同期を実現した方が、アプリケーション開発の面から見ると、容易であり、拡張性に優れた構造にすることができる。このようなことから、メディア間の同期を制御する方法を考えると、以下のような方法がある。

- メディア毎にデバイスドライバを作成して、そのドライバ間で同期を制御する。
- ユーザモードでデバイスが制御できるようにし、メディア毎にサーバ化して共有可能にする。

前者の方法を用いた場合、メディア間の同期をとるためには、OSのデバイスドライバとしての性質上から、お互いのデバイスドライバが他のデバイスドライバの動きを意識するようにしなければならず、予め決められた動きしかできなくなるという欠点がある。一方、後者の方法を用いた場合、お互いのメディアの動きを意識するように作ることもできるが、メディア毎に制御構造を用意し、それらの制御構造間の同期は、リアルタイムなスレッドのスケジューリングを用いることによって、その結果に依存することができる。またその際、スケジューラに対して、パラメータ（時間的な制約など）を与えることによって柔軟な制御を行なうこともできる。

しかし、ユーザモードでデバイスを制御するようにした場合、従来、カーネル内で処理していたデバイスからの割り込みを、どのように処理するかの問題が生じる。我々は、この問題を解決するために、デバイスからの割り込みをイベントとして扱い、そのイベントをユーザのスレッドに通知する手段として、リアルタイムイベント通知を実現した。

2.3 リアルタイムなスレッドのデータアクセス

時間的に連続な情報を持つメディアは、膨大な量のデータを扱う可能性があり、これらのデータをメモリ上にロードした場合、スワップアウトの対象になる。そして、イベント通知を受けたリアルタイムなスレッドが、このようなデータをアクセスする際、データ領域が二次記憶へスワップアウトされていたら、そこからスワップインしてこなくてはいけないので、リアルタイムな処理が続けられなくなるという問題が生じる。

我々は、この問題を解決するために、メモリをアドレスと時間の二次元空間で管理するTemporal Paging Systemを実現することにした。

3. リアルタイムイベント通知

3.1 イベントの種類とタイプ

リアルタイムイベント通知の種類としては、非同期I/Oの終了、タイマの発火、ユーザ定義のイベント、デバイスからの割り込みをイベントとして扱う。

さらに、上記のイベントをメディアの性質や、そのメディアを実現するデバイスの時間的な制約を考慮した上で、マルチメディアデバイスの時間的な制約が、デッドラインとレスポンスタイムによって決まるという仮定を行なうことによって、以下のようなタイプに分類した。

Multimedia/Realtime Extensions for the Mach Operating System.

Masatomo YAZAKI Jun NAKAJIMA Hitoshi MATSUMOTO
FUJITSU LABORATORIES, Ltd.

- デッドライン駆動 (deadline-driven) – バッファをフラッシュすることによって連続メディアを実現する場合、フラッシュするデッドラインに間に合えば、メディアの品質は劣化しない。つまり、デッドラインより以前にバッファにデータを用意しても結果は同じになる。
- イベント駆動 (event-driven) – ある絶対的な時間にデータを入出力することによって連続メディアを実現する場合、レスポンスタイムが遅れるにつれ、メディアの品質は劣化する。

このようなタイプの分類は、同時刻に発生したイベントに対してスケジューリングする場合に、その時の優先順位を決める判定要素として使用する。

3.2 スケジューリング

商用のリアルタイム OS では、プライオリティをそれぞれのハンドラに割り当てるものが多いが、この場合、時間的な制約をプライオリティにマッピングすることは、難しい。

複数のデッドライン駆動型、イベント駆動型ハンドラが並行に動作する場合の適切なスケジューリングポリシーは以下のようである。

- イベント駆動型ハンドラをなるべく先送りしつつ、
- デッドライン駆動型ハンドラのデッドラインを守る。

このポリシーを持つスケジューリングを preemptive deadline driven スケジューリングと呼ぶ。このポリシーは、一般的なイベント駆動型ハンドラが、バッファーを持たず（あるいは、かなり小さい）、短い時間内に処理が終了するという仮定に基づいている。

今回の拡張では、スレッドをリアルタイムであるものと、そうでないものとに区別し、スレッドのスケジューリングポリシーをタイムシェアリングと preemptive deadline driven スケジューリングによるものかを区別することによって実現した。

3.3 システムコール

リアルタイムイベント通知を実現するために、以下の機能を持つシステムコールを追加した。

- *event_checkin*() は、 preemptive deadline driven スケジューリングの為のパラメータ（時間的な制約のパラメータを含む）を登録する。
- *event_wait*() は、スレッドを *event_checkin*() で登録したイベントに対して待つ状態にする。
- *event_checkout*() は、 preemptive deadline driven スケジューリング対象のスレッドから、通常のスケジューリングのスレッドに戻す。

4. Temporal Paging System

4.1 Temporal の定義

通常のメモリは、アドレスをインデックスにした一次元の配列である。これに時間項を加えて、アドレスと時間の二次元空間で管理するメモリを temporal と定義する。基本的に、temporal で獲得したメモリは、時間にならないとアクセスできない。

$$memory[a] \equiv temporal[a,t]$$
$$(0 \leq t < nQ, n = 0,1,2,\ldots)$$

但し、temporal で獲得したメモリに初期値は与えられるようにする。これは、初期値を予約することになる。

4.2 Temporal の応用

現在のハードウェアでは、temporal をそのまま実現するのは、効率的でない、そこで、paging と組み合わせて使うことにした。ページ単位でメモリを管理するのであれば、ページテーブルをうまく操作することによって実現することができる。

その際、temporal で獲得した領域に音声等のような時系列データのファイルをマッピングすることによって、リアルタイムなスレッドのデータアクセスに関する問題を解決する。

5. おわりに

インプリメントは、FM TOWNS 上に行なった。但し、Temporal Paging System については、まだインプリメントは行なっていない。マイクロカーネル化した Mach3.0 を使用することによって、遅延時間および、ワーストケースになる回数を減少させることができた。（詳しくは、[1] または、[2] を参照）。

リアルタイムイベント通知の機構は、モデルとして単純であるが、実際に利用する上では、かなり細かい点まで指定しなければならないという欠点もある。これらを解決するために、ハンドラをオブジェクト指向のライブラリとして構成したり、ハンドラの動きを視覚的にモニタできるようにすることを検討している。

参考文献

[1] Jun Nakajima, Masatomo Yazaki, Hitoshi Matumoto, "Multimedia/Realtime Extensions for the Mach Operating System," in Proceedings of the Summer USENIX Conference, July, 1991.

[2] 中島, 矢崎, 松本, "Mach 上でのリアルタイムイベント通知の実現," 情処研報 Vol.91, No.107, 91-OS-53, 1991.

[3] D. Golub, R. Dean, A. Forin, and R. Rashid, "Unix as an Application Program," in Proceedings of the Summer USENIX Conference, June, 1990.

[4] Eric C. Cooper and Richard P. Draves. "C Threads," Technical Report, Computer Science Department, Carnegie Mellon University, CMU-CS-88-154, March, 1987.