5F－5

連続メディア処理向きマイクロカーネルの開発(2)
－サイクリック・スケジューラの設計と実装－

竹内　理，岩嵜正明，中原雅彦，中野隆裕，芹沢　一
(株)日立製作所　システム開発研究所

１．はじめに

近年、VOD(Video On Demand)やマルチメディアCSCW(Computer Supported Cooperative Work)をはじめとする、連続メディア処理（動画、音声などの連続メディアデータの圧縮・伸長や入出力処理）を行なうアプリケーションが、実用化に向けて急速な展開を見せている。

連続メディア処理は、各メディアが持っている時間的制約を守りながら処理を進める必要がある[1]。例えば、動画データを画面に表示する場合、1/30秒以内に１画面分の動画データをフレーム・バッファに書き出さねばならない。しかし、既存の汎用OSの多くは、ラウンドロビン・スケジューリングやFIFO順の資源割り当てを行なう。そのため、連続メディア処理を実行中に他スレッドにCPUを奪われるなど、処理の途中で予測不能な資源待ちが発生する可能性があり、上記時間的制約を保証できない[2]。

我々は、上記アプリケーションの実行に適した連続メディア処理向きマイクロカーネルHiTactixを開発している。HiTactixは、資源競合による予測不能な待ちを完全に排除し、上記問題を解消している。本稿では，HiTactixが提供する連続メディア処理向きスケジューリング機能（サイクリック・スケジューリング）の概要を示す。

２．連続メディア処理の特徴と要求機能

連続メディア処理の特徴と要求機能を、図１を例に考える。図１では、スレッド１は、ビデオカメラからデジタル化された動画データをバッファ１に読み込む。スレッド２は、バッファ１に読み込まれた動画データのMPEG圧縮を行ない、MPEGデータをバッファ２に書き出す。スレッド３は、バッファ２に書き込まれたMPEGデータをATMネットワークに送信する。

図１のような連続メディアアプリケーション（以後、連続アプリと略す）は同一の処理を周期的に繰り返す。図１

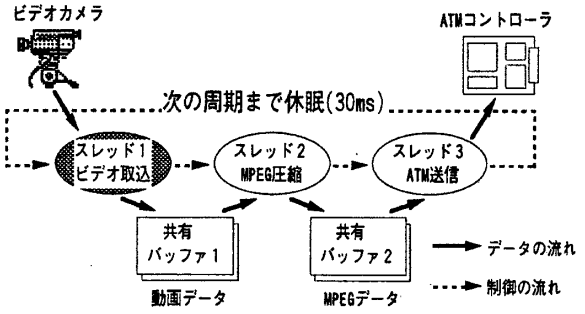


図１　連続メディア処理の典型例

では、ビデオ取込→MPEG 圧縮→ATM送信という処理パターンを周期的に繰り返す。そして、処理パターンの実行開始時刻の間隔は厳密に一定でなければならない。図１では、スレッド１の実行開始時刻の間隔がゆらぐと、バッファ１のアンダーフローやオーバーフローが生じ、その結果、スレッド３が送信するMPEGデータの品質が低下する。

また、一部の連続アプリは、図１と同様の処理を複数並列に実行する必要がある。例えばCSCWは、動画データの送信と並行して、受信も図１と同様に処理する。

従って、連続メディア処理向けのスケジューラには、複数のスレッドを、各スレッド毎に指定された周期を厳密に守りながら並列に周期駆動する機能が必要となる。

３．既存OSインタフェースの問題点

既存OSインタフェースを用いて連続アプリを構築する場合生じる問題点の一例を、UNIX*[3]を例に述べる。

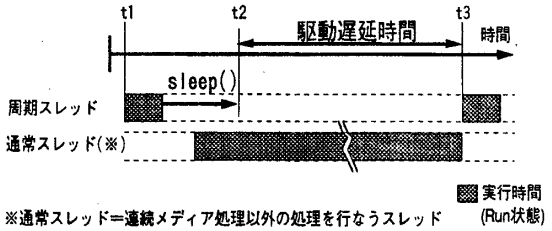
UNIXでは、スレッド[注1]の実行中断、再開のインタフェースとしてsleepが提供されている[3]。UNIX上での、sleepを用いたスレッド周期駆動プログラムを図２に示す。

```
while(1){
        < 周期処理プログラム >
        sleep(interval_time);
}
```

図２　UNIXにおける周期駆動プログラム

図２のプログラムを実行する連続メディア処理スレッド（以後、周期スレッドと略す）を複数並列に動作させた場合、図３に示す通り、複数の周期スレッド、又は周期スレッドと通常のスレッドが同時にReady状態になる可能性がある。図３の例では、周期スレッドは、時刻t2において指定休眠時間が経過しReady状態に遷移している。しかし、時刻t2ではRun状態のスレッドが他に存在するため、時刻t3までRun状態への遷移が遅延されている。そのため、周期スレッドを厳密に一定の間隔で周期駆動できていない。

このように、UNIX（のスケジューラ）は、２節で述べた要求機能を提供していない。他の既存ＯＳも同様であり[4,5]、上記のような問題点の解決には、連続メディア処理向きスケジューリング機能の新規設計が必要となる。



※通常スレッド＝連続メディア処理以外の処理を行なうスレッド

図３　駆動遅延の発生

A Micro-kernel for Countimuous Media Processing
－A Design and Implementation of Cyclic Scheduler－
Tadashi Takeuchi, Masaaki Iwasaki, Masahiko Nakahara, Takahiro Nakano, Kazuyoshi Serizawa
Systems Development Laboratory, Hitachi Ltd.

* UNIX は X/Open Company Ltd. の登録商標です。

[注1] 厳密に言えば、UNIXではスレッドは提供されていない。本節では、UNIXのプロセスをスレッドと呼ぶ。

4．サイクリック・スケジューリングの概要

本節では、連続メディア処理向きスケジューリング機能であるサイクリック・スケジューリングの概要を示す。

サイクリック・スケジューリングでは、周期スレッドが、図4のように、予め必要な周期Tと1周期あたりの実行時間Lをスケジューラに要求する。TやLは、扱う連続メディアや処理内容により、アプリケーションごとに予測可能な値である。

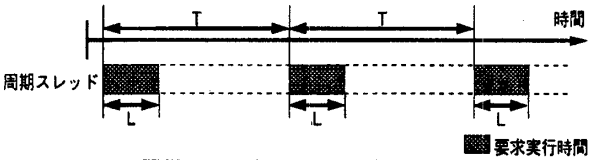


図4　周期と1周期あたりの実行時間の要求

要求を受けたスケジューラは、各スレッドが要求したTとLを満たすタイムスロット・テーブルを作成する。タイムスロット・テーブルとは、図5に示すように、タイマ割り込み発生時刻を境界に、時間をタイムスロットと呼ばれる単位に分割し、各タイムスロットごとにスケジューリングすべき周期スレッドを記したテーブルである。

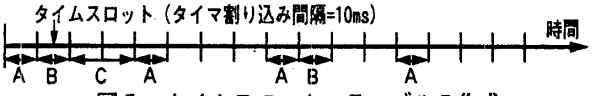


図5　タイムスロット・テーブルの作成

このタイムスロット・テーブルは機械的に作成可能である。例えば、周期スレッドA～Cが表1に示すTとLを要求した場合、これらの要求を満たすタイムスロット・テーブルは図5のようになる。

表1　周期と1周期あたりの実行時間の要求例

	T	L
A	40ms	10ms
B	80ms	10ms
C	160ms	20ms

タイムスロット・テーブルの作成を完了したスケジューラは、タイマ割り込みを契機に起動され、タイムスロット・テーブルに記された周期スレッドを順番にスケジューリングする。

このスケジューリング方式では、周期スレッドは、一定間隔で確保したタイムスロットにおいて必ずRun状態に遷移することが保証される。従って、各周期スレッドの駆動間隔は厳密に一定に保たれる。

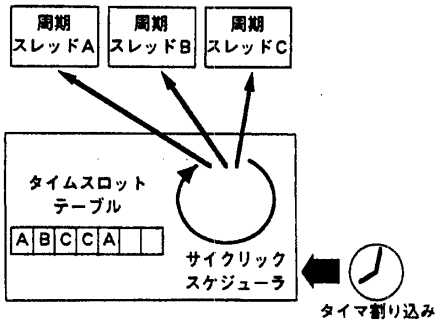


図6　周期スレッドの起動

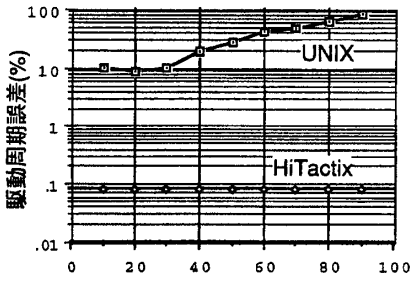


図7　周期駆動性能比較

5．評価結果

本節では、サイクリック・スケジューリングの周期駆動性能の定量的評価の結果を示す。

評価方法

1）HiTactix及びUNIX上で、周期スレッド（T＝160ms）を4多重に並行実行する。

2）並行に動作させる周期スレッドのLの値を可変にし、各周期スレッドの駆動周期誤差の平均を測定する。

但し、駆動周期誤差Δtは、

$$\Delta t = |(\text{実際の駆動間隔}) - T| / T \times 100(\%)$$

CPU利用率ρは、

$$\rho = (L \times 4 / T) \times 100(\%)$$

で与えられる値である。

測定結果を図7に示す。UNIXでは、CPU利用率に応じて10～100%程度の駆動周期誤差が発生しているのに対し、HiTactixでは、CPU利用率に関わらず、0.08%程度の駆動周期誤差しか発生していない。なお、HiTactixの周期駆動誤差は、ネットワーク・パケット到達などの非同期イベントが発生しても殆んど変化しない。

6．おわりに

本稿では、連続メディア処理向けのスケジューラには、複数のスレッドを、各スレッド毎に指定された周期を厳密に守りながら並列に周期駆動する機能が必要であることを示した。そして、上記機能を提供するサイクリック・スケジューリングの概要について述べた。実装の結果、サイクリック・スケジューラは、0.08%程度の駆動周期誤差で周期スレッドの周期駆動が可能なことを確認した。

参考文献

[1] 河内谷清久仁他, "連続メディアのQOS制御のためのOSサポート", 情報処理学会コンピュータシステムシンポジウム, 1994.

[2] 和田英彦他, "Keio-MMPにおけるマイクロカーネルアーキテクチャ", 情報処理学会全国大会, 1994.

[3] Maurice J. Back, "UNIX カーネルの設計", 共立出版, Oct.1990.

[4] Tatuso Nakajima etc, "Experiments with Real-Time Servers in Real-Time Mach", Usenix Mach III Symposium, 1993.

[5] 坂村健, "μITRON3.0ハンドブック", パーソナルメディア, 1993.