周期駆動機能を持つリアルタイムタスクスケジューラの開発

1F－5

鈴木 貴明　吉澤 康文
東京農工大学 工学部

1 はじめに

動画像や音声などの連続メディアを処理するアプリケーション(AP)は、データ取得、展開、再生の基本処理を周期的に繰り返す性質を持つ。タイムシェアリングシステムをベースとし、CPU資源の公平分配を目的とする既存のオペレーティングシステム(OS)上で連続メディア処理を行うと、システム負荷の影響を受け、動画のこま落ちや遅延再生など、連続メディア処理特有の障害が出る。

また、OSがAPに対して提供する時間情報は貧弱な場合が多い。APは現在自分が処理している時刻などを把握することが不可能なため、きめ細かい時間制御などが困難である。

本研究では、連続メディア処理に最適な周期駆動機能を持つタスクスケジューラを開発する。

2 周期的タスクモデル

周期的なCPU資源分配の対象となるタスクを周期的タスクと定義する。周期的タスクは属性として周期長と周期内に必要なCPU時間(必要処理時間と呼ぶ)を持ち、生成時にこれらのパラメータをスケジューラに対して申告する。スケジューリング可能性判定の結果、生成が成功すると、タスクは周期的なCPU時間確保が保証される。

周期的タスクは毎周期の処理終了時に、周期内処理の終了をスケジューラに通知するシステムコールを発行する義務を負う。このシステムコールは同時に次の周期開始までスリープする機能を持つ。このスリープは周期開始時刻になるとOSにより自動的に解除される(図1)。

次周期処理開始時に、希望開始時刻と実際の開始時刻とのずれ(スケジューリング遅延)を取得できる。

また、デッドライン超過時や必要処理時間以上のCPU時間を消費した場合に行うリカバリ処理を、ハンドラとして登録可能である。

3 スケジューリング方式

周期的タスクの優先度は周期的タスク以外のタスク(非周期的タスク)より高く設定し、周期的タスクの周期駆動が他のタスクから影響を受けることを防ぐ。周期的タスク間の優先度決定は、周期の短いタスクを優先するRate Monotonic方式で行う。

周期長が同一の周期的タスクが複数個存在する場合はFirst Come First Service方式でスケジューリングする。

周期的タスクに対して与えたCPU時間は各タスク毎に集計し、周期内で与えた総CPU時間が必要処理時間を超過した場合、そのタスクを終了させ、システム全体のCPU時間の不足を防止する。

また、各周期的タスクの周期終了時刻を監視し、周期終了時刻に達した周期的タスクにはウエイクアップ処理を行い、次周期の処理を開始させる。

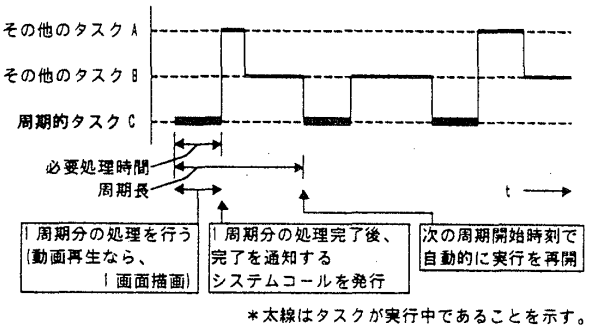


図1 周期的タスクモデル

4 スケジューリング可能性判定

周期的タスクは生成時に周期長と必要処理時間を申告するが、このとき、既存の周期的タスクのCPU時間確保を妨げることなく、希望どおりのCPU時間割当てが可能かを判定する。

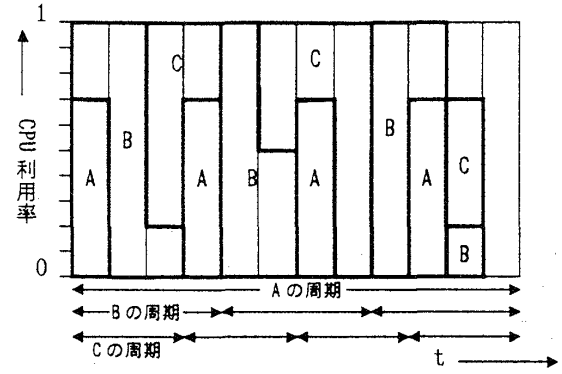
図2のように、各周期的タスクの周期の最小公倍数を範囲として、周期の短いタスクから必要処理時間を割当て、すべてのタスクが範囲に収まる場合、スケジューリング可能と判断する。

5 時間情報の提供

システムコールの処理時間はハードウェアの構成によって異なるため、処理時間の見積もりを必要とする連続メディアAPの構築を困難する原因の一つとなる。このような背景からシステムコールの処理時間を提供する機構を付加する。システムコールの処理が完了する毎に、ユーザが予め指定した変数に処理時間が自動的に格納される。単位はマイクロセカンドである。

Development of A Periodic Task Scheduler for Real-Time Operating Systems
Takaaki SUZUKI, Yasufumi YOSHIZAWA
Faculty of Engineering
Tokyo University of Agriculture and Technology

周期的タスクA(周期長12、必要処理時間3.5)と
周期的タスクB(周期長4、必要処理時間1.5)がすでに存在し、
周期的タスクC(周期長3、必要処理時間0.7)を新規に生成する場合



各周期の最小公倍数の範囲で、周期の小さいタスクから必要処理時間を当てはめていく。すべてのタスクを収めることが可能な場合、スケジューリング可能と判定する。

図2 スケジューリング可能性判定の例

6 タイマ割込み間隔の設定

タイマ割込みは、スケジューラ起動のトリガとなる。そのため、タイマ割込み間隔は周期的タスクのスケジューリング精度に影響する。例えば、タイマ割込み間隔を10msに設定した場合、最悪10msの間、スケジューラが起動せず、周期的タスクが駆動できない可能性がある(図3)。

本研究では、タイマ割込み間隔を1msとした。これにより、スケジューリング遅延を約1ms以下にすることが可能で、例えば、フレームレート30(枚/sec)の連続メディア処理を行う場合、周期長33msの周期的タスクを生成することで、遅延を3%以下に押さえることが可能である。

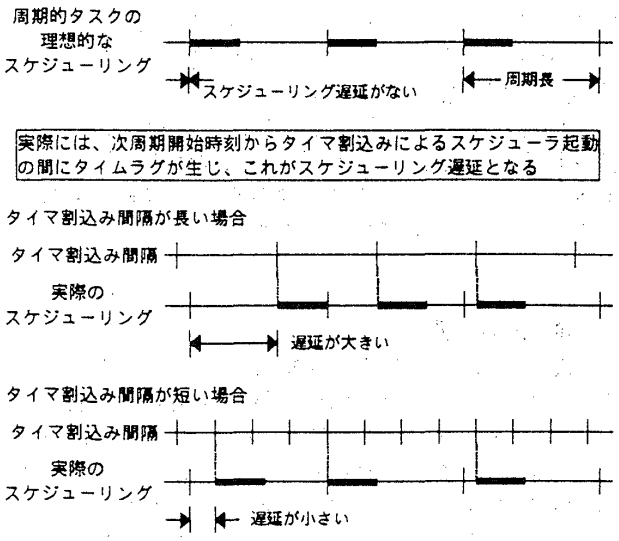


図3 タイマ割込み間隔とスケジューリング遅延の関係

7 実装

実装はLinux Kernel2.0.0上に行った。今回実現したシステムコールを次に示す。

period_start

非周期的タスクを周期的タスクに移行させる。同時にスケジューリング可能性判定を行う。引数として、周期長と必要処理時間を取る。周期的タスクからこのシステムコールを呼ぶと周囲長や必要処理時間の変更が可能である。

period_end

周期的タスクとしての動作を終了し、非周期的タスクへ戻す。

period_skip

次の周期の開始までスリープする。次周期開始時刻にスケジューラによりウエイクアップされる。同時にスケジューリング遅延時間を取得できる。

period_set_handler

デッドラインオーバ時及び必要処理時間以上のCPU時間を消費した場合に呼び出すハンドラを登録する。

syscalltime_start

システムコール処理時間の取得を開始する。引数として、処理時間の格納先アドレスを取る。以後、すべてのシステムコールが発行されるごとに、その処理時間が通知される。

syscalltime_end

システムコール処理時間の取得を終了する。

8 まとめ

周期的な資源要求を行う性質を持つ連続メディア処理を指向し、周期的タスクモデルと周期駆動機能を持つタスクスケジューラを開発した。また、周期的タスクが処理時間を把握するため、システムコール処理時間の取得機構を実装した。今後、周期駆動性能の評価を行う予定である。

謝辞

本研究は文部省科学技術研究補助金：基礎研究C-2-10680338の成果である。