システムソフトウェアと
オペレーティング・システム 81－3
（１９９９．５．６）

データ型を基にしたスケジューリング方式の設計

岡村 真一郎　　並木 美太郎

東京農工大学大学院工学研究科

リアルタイム処理は単一のタスクが周期的に起動することによって処理が行われ、各タスクが同期をとりながら実行されることによってメディア処理を実現している。しかし、複数の連続メディア処理が実行されると資源の競合や、タスクの周期実行が困難となる。本論文では、システムが複数の周期的なタスクの実行保証を行うためにデータ型を考慮したスケジューリング方式の設計について述べる。連続メディアデータはデータ型ごとの特性をもっており、処理時間や実行周期が異なる。この特性をシステムに登録し、そのパラメータを基にシステムがデバイスなどの資源管理を含めてスケジューリングを行うことでCPU資源の無駄な使用や周期性の保証を行うことができる。また、周期タスクを入力、復号、出力といった処理単位に分離することで厳しいリアルタイムを必要とする出力処理の周期性の確保と、スケジューリングの可能性を上げることができる。

The Design of Scheduling Method Based on Data Type

Shinichiro Okamura　　Mitaro Namiki

Graduate School of Technology,
Tokyo University of Agricultuer and Technology

In this paper, a design of scheduling method based on data type so that a system guarantees periodic execution of multiple tasks is described. Recently, continuous media has been large capacity because ability of a computer has been improving. Data of continuous media has characteristics of each data type. processing time of each data type and execution period of each data type are different. These characteristics are registered in a system so that use less use and the guarantee of the periodicity can be done by doing resource management and scheduling method based on these data type.

1 はじめに

近年、計算機の処理能力の向上にともない音声や映像などの大容量連続メディアによる情報の表現を行うようになってきている。これはテキストデータのような静的な表現方法よりも、短時間に表現できる情報量やインタフェース面で優れているからである。また、連続メディアデータにはパターン特性から様々なデータ型が存在し、単体で扱われることは少なく複数の連続メディアデータを使用して情報表現を行なう。実際に計算機でこれら連続メディアデータを扱う場合、計算機資源が有限なためにデータ処理に必要な資源を、必要とする時に使用できないことがある。

映像や音声など連続性を持つデータを扱う際には、メディアごとの時間制約に対する条件を満たすために、CPU や各デバイスなどの計算機資源を、時間制約を守れるように使用できる機構をシステムが提供しなければならない。このような場合、方法としてサービスの質(Quality of Service)を使用できる資源量に合わせて変更し処理可能な状態にする。動画であれば、フレームレートや映像の質をさげたりすることによって時間制約を守れるように調節を行う。

使用できる資源量からスケジューリングを行い、そのままのサービスの質で実行できない場合には QoS をさげて実行可能な状態にする。しかし、実際には様々なデータ型が存在することから時間制約、使用する資源の違いがあるので、システムはこのようなデータ型ごとに持っている特性を考慮した資源配分を行う必要がある。現在使用できるだけの資源をある周期タスクに渡したとしても、タスクにとって意味のない資源量では結果としてその時間の資源は無駄となってしまう。

本稿では、周期的処理を必要とするタスクが複数動くようなシステムにおいて、データ型ごとに異なる時間制約や処理内容をシステムが把握し、それを基にデータ型に合った資源管理と分配を行うスケジューリング方法について述べる。

2 連続性のあるメディアの特徴

連続メディアデータにどのような特徴があるかを述べる。例えば MPEG 形式と呼ばれる連続メディアデータなどの動画データは圧縮アルゴリズムによるフレームの圧縮と、フレームとフレームの依存関係を大きくすることによってデータの圧縮を行っている。

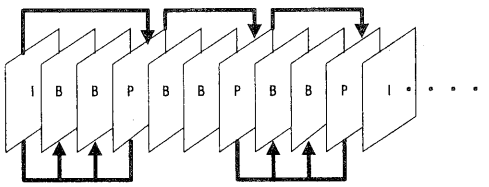

図1 MPEG 形式のフレームの依存関係

独立したフレーム（I ピクチャ）は、1 秒間に表示される 30 フレーム中に、4 枚から 5 枚ほど挿入されてる。図 1 からもわかるように、独立したフレーム以外のフレームを再生するには依存するフレームの復号は必須である。B ピクチャを再生するためには前後のフレームである I、P ピクチャが復号されていなければならず、P ピクチャを復号するためには I ピクチャと P ピクチャの復号が必要である。

MPEG の規格では 1 秒間に 30 フレームの再生を標準としているが、マシンのスペックやシステムの状態に合わせて再生フレームの枚数を自由に変更することができる。表 1 に示すように、I ピクチャだけの再生を行うと 1 秒間に約 5 フレームの再生が行われ、I、P フレームだけの再生を行うと 1 秒間に約 12 フレームの再生となる。

I、P、Bフレームの再生 (1)	30 f/sec
I、P、Bフレームの再生 (2)	21 f/sec
I、Pフレームの再生	12 f/sec
Iフレームだけのの再生	4 f/sec

表1 依存関係を考慮したフレームレート

音声データはある時間における点が集合し、全体として連続した波形を形成する。これよりその波形データを1秒間に何回再生するかによって再生パフォーマンスを変更することができる。よく使用される周波数は電話品質である11kHz、ラジオ品質である22kHz、CD品質である44kHzなどであり、これら以外の品質での再生をおこなっても人の耳には品質の変化を聞き分けることはできない。

このように連続メディアデータはシステムの状況やユーザの要求によってサービスの質を変化することができるが、どのように変化させるかはデータ型ごとの特性により異なる。

3 資源管理方法の設計

連続メディア処理は周期タスクとして実装されるが、複数のタスクが動くシステム上において連続メディア処理タスクを起動すると資源の確保が困難となり、周期実行ができなくなる。資源量が十分に確保できなくとも周期実行ができるように、周期を資源量に合わせて変化させるのがQoSである。従来のQoS方法としては周期タスクに対しシステムが使用できる資源量を渡し、その資源量にたいして実行可能な周期でタスクを起動させていた。しかし、システム側が資源利用状況から利用できる資源量を通知しても、2章で述べたようにメディアの圧縮特性によってはそれだけのフレーム再生には、それ以上のフレームの復元が実際には必要となる場合がある。よって、フレームレートをさげるにしてもこれらの依存関係を考慮して行わなければならず、そのように行わなければシステムに対する負荷は表示されているフレームレートほど実際には下げられていないことになる。また、データ型によってその圧縮形式はさらに異なり、適切なQoS管理を行わなければCPU資源が無駄に使用されるために、本来スケジューリングができる状況でもスケジューリングができなくなってしまう。

システムが周期タスクに対し適切な資源配分を行うために、システムが処理されるデータ型の特徴を把握していれば無駄にCPU資源を消費することなくスケジューリングが行える。そこでデータ型に基づいた資源管理方法と資源の割当てを行うスケジューラの設計を行う。資源利用の無駄をなくすことによって、複数のタスクが起動していても適切な資源配分を行うことができ、連続メディア処理も適切に行うことができる。設計方針を以下に示す。

- リアルタイム性を必要とする処理に対し周期的に実行できるようにスケジューリングを行う
- システムの資源状況に合ったQoS制御を行うにあたり、データ型を基にしたQoS管理を行う
- もっとも厳しい時間制約を必要とするデバイス出力処理をスケジューリングの対象とする

次に詳細について述べる。

3.1 データ型に基づくQoS管理

QoSを考慮した場合の処理内容はデータ型ごとに静的に決まってくる。MPEGであればフレームの依存関係から表1のフレームレートがシステムにたいする負荷をさげる値となる。このようにデータ型ごとにQoS値が静的に決まってくるのであれば、その値をシステムに登録しておくことでシステムが資源配分を行う際に、資源を有効に使われるかどうかを知ることができ、スケジューリング時にCPU資源の有効配分を行うことができる。

3.2 出力デバイスの管理

連続メディア処理は周期タスクで行われるが、処理の流れはデータの読込み、データ型ごとの復号処理、デバイスへのデータ出力処理が周期的に行われる。連続メディア処理において厳しい時間制約を必要とするのは出力処理である。データの

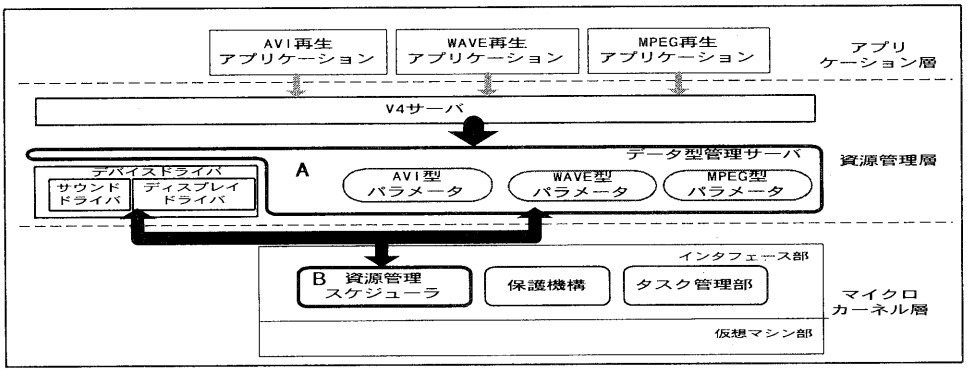

図2　OS/omicron 第4版における全体構成図

入力処理と復号処理は、出力処理が実行される時間までに終了していればよい。出力処理の周期性が確保できれば、それ以外の処理は出力処理ほどの厳しい時間制約は必要としない。そこで、3つの処理過程から出力処理を分離し、出力処理をタスクとして実装する。復号処理タスクと出力処理タスクの依存関係はスケジューラに明示し、スケジューラはこの依存関係をくずさないようにスケジューリングを行う。厳しい時間制約を必要とする出力処理と復号処理を分離し、スケジューリングを行うことにより出力の周期タスクを守らせることができ、タスクを作業単位ごとに分割し、タスクの実行時間を短くすることによってスケジューリングの柔軟性を確保できる。

3.3　連続メディア処理を行うためのスケジューリング方法

連続メディア処理は周期タスクによって実現されている。周期タスクセットに対し利用されるスケジューリングアルゴリズムとして RM(Rate Monotonic)法があるが、これを基にしてスケジューリングを行うとタスク間の依存関係は考慮されず、固定優先度によるスケジューリング方式なので、CPU 使用率の上限が低いことも知られている。3.2 で述べたように、タスクを処理過程ごとに分離することで時間制約の厳しい部分の周期性を確保するためには復号処理までが必ず終了していなければならず、依存関係を考慮しなければならない。また、CPU 使用率を上げるためには状況に応じて優先度を変更させる必要があることが知られている。EDF（Earlist Deadline First）法を使用すると CPU 使用率は最大で 100%になるが、デッドラインの早い順に実行するアルゴリズムであるので、このままではタスクの依存関係を考慮してスケジューリングを行うのは無理である。

データ型の特徴をシステムが把握することで CPU 資源の無駄をなくし、タスクを分離することによってスケジューリングの柔軟性を確保した。

そこで、CPU 資源を最大限に使うことができる EDF アルゴリズムが望ましい。依存関係のあるタスクにたいしては、デッドラインと起動時刻をずらすことによってタスクにデッドラインを基にした優先順序をつけることが可能となる。これに EDF 法に適用すれば、依存関係を考慮したスケジューリングを行うことができる。

4　データ型に基づく資源管理の実現

データ型を基にしたスケジューリング方式を設計するにあたり独自ＯＳである OS/omicron 第4版（以下 V4 と略す）を基盤として設計を行った。V4 は手書きや画像データといったマルチメディアデータの編集と再生をターゲットにしたパー

ソナル OS である。V4 における全体構成を図 2 に示す。

4.1 データ型を基にしたスケジューラによる資源管理モデルの実現

スケジューリング方法としてデータ型に基にしたスケジュールにより資源の割当てを行うために、スケジューラはデータ型ごとに特徴を示したパラメータを持つデータ型管理サーバ(図 2-A)と、データ管理部のデータを基にしたスケジューリング、デバイス管理を行う資源管理スケジューラ(図 2-B)で資源管理スケジューラを構成している。図 2 で示すように資源管理スケジューラはマイクロカーネル層に、データ型管理サーバは上層である資源管理層に実装し、協調作業で資源管理を行う。

データ型の管理をサーバとして分離することで、データ型の登録、追加、削除、変更といったプリミティブをカーネルの再構築を行わなくても可能である。また、システムが動くハードウェア環境によっては実行速度が変化するので、環境に合ったパラメータ値に変更することできることによって、適切な資源管理を行うことができる。

データ型管理サーバにはデータ型ごとに次のようなパラメータを持っている。

・QoS として意味のある値ごとの実行周期と実行時間
・デッドライン
・使用する周辺デバイス

タスクの生成処理の流れを図 3 に示す。データ型管理サーバは実行されるタスクの起動要求を受けると、タスクがどのようなデータ型を扱うかを確認する。次に扱うデータ型から使用されるパラメータを選択し、資源管理スケジューラにスケジューリング要求を通知する。資源管理スケジューラは、現在の資源利用状況をデータ型管理サーバに通知し、データ型管理サーバは静的に決定されているデータ型ごとのパラメータ表から、現在の資源使用状況に応じたパラメータ表を資源管理スケジューラに提供する。資源管理スケジューラは受け取ったパラメータを基にスケジューリングの可能性を判定する。

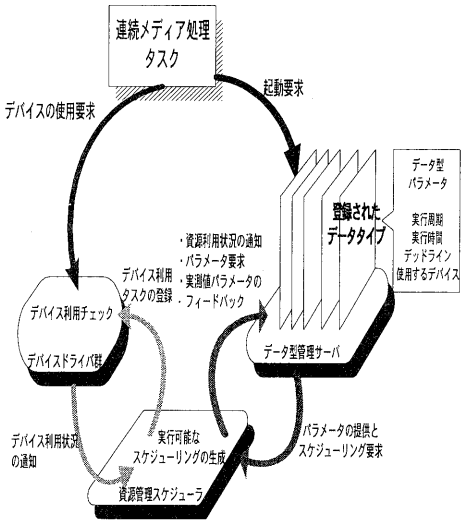

図3　タスク生成処理の流れ

3.3 で述べたようにスケジューラはタスクの開始時刻とデッドラインを変更し、優先順位をもたせることで、タスクの実行順序を持たせる EDF アルゴリズムで行う。優先度はデッドラインの早いタスクほど高く、タスクの処理時間とデッドラインを基にスケジューリングを行う。スケジューリング時に、現在の資源利用状況では実行ができない周期タスクが発見された場合には、スケジューる表からデッドラインの遅いタスクを見つけ、そのタスクにたいし QoS 値の低いパラメータをデータ型管理サーバに提出させて、際スケジューリングを行う。デッドラインが同じタスクが起動する場合には、先に起動要求がだされたタスクを優先してスケジューリングを行う。

4.2 処理分割の実現

もっとも厳しい時間制約を必要とする出力処理を別タスクとして実装することで、スケジュー

リング時に復号などの周期性がとれなくとも出力の周期性を保つことができる。V4 では関数をタスクとして実装させるプリミティブが用意されており[1][2]、これによりデバイス出力処理などをタスクとして実装が可能である。

出力処理の周期性を保証できない場合にはその周期保証ができるまで出力は行わず、アイドル時間がある場合には、その時間に復号処理だけを先に行い出力処理の周期保証を確保できた時点で出力を行うことができる。また、資源管理スケジューラによる調停がうまくできない場合は、データ型管理サーバに現状での実行不可能を通知し、現在のシステム資源状況にあったパラメータをデータ型管理層に提供させる。

資源管理スケジューラはそれぞれのタスクが提供した資源利用時間と実際に利用された時間とを監視し、データ型管理サーバにフィードバックさせることで、各システムに合った実測値でのスケジューリングが可能となる。連続メディアの特性（同じ処理を繰り返し行う）からこの値は一度測定してしまえばそのシステムのハードウェア環境が変更されなければ、測定の必要はない。

4.2 デバイス管理の実現

デバイスの利用を保証するためにドライバにスケジュール表を持たせ現在ドライバを使用するタスクにだけ使用できる許可を発行する。資源管理スケジューラがスケジューリングを行い、スケジュール表を作成すると同時に、タスクが使用するドライバごとにタスクの ID を発行し、起動するタスクがどのような周期で使用するかをドライバに通知する。ドライバ側では使用要求をだしてきているタスクにたいし、利用要求をだしたタスクの ID とドライバがもっている ID 表を使用してデバイス使用の管理を行うことによって、利用保証を行う。

6 おわりに

本稿では連続メディア処理に必要な周期タスクの保証のためにデータ型ごとの特徴を考慮し、CPU 資源の無駄な使用をなくすためにデータ型を基にしたスケジューリング方式の設計と、デバイスなどの利用保証をするための資源管理方法の設計につてい述べた。今後の課題はスケジューラの実装と評価、連続メディア処理に適したメモリ管理方法の構築である。

参考文献

[1]森永智之，早川栄一、並木美太郎、高橋延匡：単一２次元アドレス空間を提供する拡張可能なマイクロカーネルの開発，情報処理学会論文誌、Vol.38，No.5，pp.1016-1025、1997

[2]佐藤元信、森永智之、早川栄一、並木美太郎、高橋延匡：OS/omicron 第４版におけるシーケンシャルデバイス管理方式の拡張、情報処理学会コンピュータシンポジウム論文集、pp.179-186、1996

[3]河内谷清久仁：マルチメディア処理の動的 QoS 制御のためにフレームワーク、信学会和文論文集 B-I、Vol. J-80-B-I、No.6、pp.465-471、1997

[4]河内谷、徳田：MKng プロジェクトにおける動的 QoS 制御サポート、第 53 回情報処理学会論文集、5B-8、pp..1-47-1-48、1996