システムソフトウェアと
オペレーティング・システム 63−10
（１９９４．３．２２）

マルチメディア処理におけるOSレベルでのリソース・リザベーション

松尾聡[1,2]　岡村耕二[2]　荒木啓二郎[2]　福田晃[2,3]
{satosi-m,oka,araki,fukuda}@is.aist-nara.ac.jp

[1]ミノルタカメラ（株）　西神情報センター

[2]奈良先端科学技術大学院大学　情報科学研究科

[3]九州大学工学部　情報工学科

我々は、時間的制約のあるマルチメディア処理を行なうために、基本ソフトウェアであるオペレーティングシステムのスケジューリング方式や資源管理方式に時間的制約の保証を行なう機能を組み込む研究を行なっている。本稿では、マルチメディア処理の時間的制約を保証するためのリソースの予約確保の方式を提案する。我々は、リソースを予約確保するためにリソースを時分割により定量化して管理を行う方式を示し、本方式を用いた、CPUの予約確保について考察を行う。

Resource Reservation in Operating Systems for Multimedia Processing

S.Matsuo[1,2]　K.Okamura[2]　K.Araki[2]　A.Fukuda[2,3]
{satosi-m,oka,araki,fukuda}@is.aist-nara.ac.jp

[1]Minolta Camera Co.,Ltd.
Advanced Science Center of Seishin
4-4-1 Takatsukadai, Nishi-ku, Kobe 651-22, JAPAN

[2]Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama, Ikoma 630-01, JAPAN

[3]Department of Computer Science and Communication Engineering, Kyushu University
6-10-1 Hokozaki, Higashi-ku, Fukuoka 812, JAPAN

We have been researching the mechanisms which guarantee the timing constraint for the scheduling and resource management of operating system for proper multimedia processing. In this paper, we propose the mechanism of the resource reservation for multimedia processing which need to guarantee timing constraint. We show how to reserve the resources such as CPU and Network by correct time slicing.

1 はじめに

近年、ワークステーションやパーソナルコンピュータの高性能化、ネットワークの大容量化が飛躍的に進んできた。これらにより、ネットワークを介した分散環境でのマルチメディア処理が実用的になってきた [1]。

数値計算などの従来の処理とは異なり、マルチメディア処理では個々のメディアは連続的に処理され、これらのメディア処理の間には関連性がある。しかし、現在のオペレーティングシステム（以下 OS と表記する）ではこれらのマルチメディア処理の制約に対する機能が備わっていない。そこで、我々は従来の計算機の分散環境でのマルチメディア処理のモデル化を行なった。また、その実現のために OS にそれらの機能を組み込んだ PDE(Parallel Distributed Environment)-II の開発を行なっている [2]。

図 1は多数のエンティティがエンティティ間に張られるコネクションを通じてマルチメディアデータの交換を行なっている様子を表している。図中、円筒の芯の部分は CPU やネットワーク、メモリなどのシステムリソースを、円筒を貫くエンティティ間のリンクはコネクションを示している。一方、円筒の表面部分とコネクションの交わりは、同期をとるためのポイントを示している。システムリソースはエンティティ間で通信を行なうために必要なシステムのリソースを抽象化し、ある単位で細分化したもので、それを共通に使用するエンティティの集合に対して 1 つ存在するものとする。システムリソースは有限で、その高さはシステムの全処理能力を表している。個々のコネクションはその設定に必要な品質から計算できる量を予約確保し使用する [3]。本研究はそのシステムリソースの管理を行なうものである。

本稿では、マルチメディア処理の時間的制約を保証するための PDE-II での同期機構 [4] のための周期スレッド [5] でのリソースの予約確保の方式とそのためのリソースサーバについて提案し、本方式による CPU の予約確保について述べる。

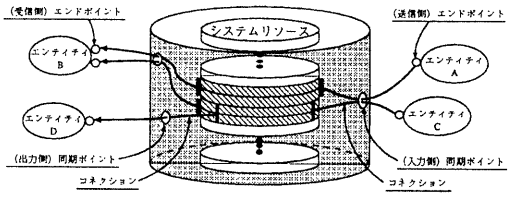

図 1: マルチメディア処理モデル

2 マルチメディア処理

2.1 マルチメディア処理の特徴

音声や映像といったメディアは、データが時間的制約を持つという特徴を持つ。本稿では、メディアの 1 つの処理のためのデッドラインを周期性、その処理が繰り返し行なわれることを連続性として考える。連続性は周期性によって実現される。例えば映像の 1 つであるアニメーションでは一般に 1 秒間に 24 枚の絵を動かしている。これをコンピュータ上で行なうためには、1 枚の映像データは 1/24 秒以内に処理しなければならない。この場合 1/24 秒が周期性、繰り返し 1 秒間に 24 枚処理することが連続性である。

マルチメディアではこれらの複数のメディアを扱っていて、それらのメディア間には相互関連が存在する。マルチメディア処理では特にメディアの連続性が重要であり、そのために周期性を保証しなければならない。これによりメディア間の相互の関連をとることも可能になる。

本稿では、マルチメディア処理のために扱う CPU、ネットワークを抽象化したものをリソースと総称して扱う。これらのリソースはそれぞれ、処理を行なう CPU、データを転送するネットワークというように性質が異なっている。しかし、あらかじめリソースを抽象化することで、リソースサーバはこれらのリソースを同等に扱うことができる。

2.2 現在のOSにおけるマルチメディア処理時の問題点

マルチメディア処理のために予約確保する場合に要求される点と、UNIX に代表される現在のOSで上記の特徴を持つマルチメディア処理を行なう場合の問題点を以下にリソースごとに挙げる。

- CPU
  - 要求される点
    メディアの連続性を保証するために処理を周期的に行なわなければならず、常に要求された周期でプロセスをCPUに割り当てなければならない。
  - 問題点
    常に一定時間でプロセスの切替えを起こす機構が備わっていない。
- ネットワーク
  - 要求される点
    映像や音声データの転送のために必要なネットワークバンド幅を常に確保しておかなければならない。
  - 問題点
    トラフィックの増加により、必要とするネットワークバンド幅を維持することが困難である。

例えば音や映像の連続メディアの場合、これらの問題点のために処理が不規則になり、そのため音や映像の不連続な出力や、音と映像の同期がずれたりする現象が生じる。

3 リソースの予約確保

これらの問題点を解決するためのリソースの予約確保の方式を本章で提案する。

3.1 リソースの予約確保

我々は、リソースの予約確保を以下の様に行なった。

1. リソース毎に設定するリソースサーバがそれぞれのリソースを管理する。
2. リソースサーバがアプリケーションからの予約要求に対して判断する。

例えば、ネットワークを介した映像データの処理を考えると、CPUの予約確保は映像データを処理するための周期的なプロセス処理の保証を行なうことであり、ネットワークの予約確保は映像データを転送するために必要なネットワークバンド幅を確保することである。

これらのリソースの管理および予約確保はリソースサーバにより行なわれる。

3.2 リソースサーバの構成

リソースサーバは各リソース毎に設置される(図2)。各リソースサーバはそれぞれのリソースを定量化することでリソースを抽象化する。

CPU サーバは数値化された CPU の処理能力、ネットワークサーバは数値化された通信能力を用いて予約確保を行なう。

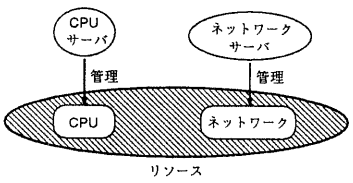

図 2: リソースサーバ

3.3 リソースサーバ間の調停

実際に資源の予約確保を行なう場合、CPU、ネットワークの各リソース間には依存関係がある。

リソースを予約確保するためにはリソースの管理をしなければならず、管理処理のためにCPUを予約確保する必要が生じる。この場合、リソースサーバ間で調停を行ないリソースの予約確保を行なう（図3）。

例えば、ネットワークを予約確保する時は、ネットワークサーバはCPUサーバに管理処理用のCPUの予約確保を依頼し、その結果を受けてネットワークの予約確保を行なう。

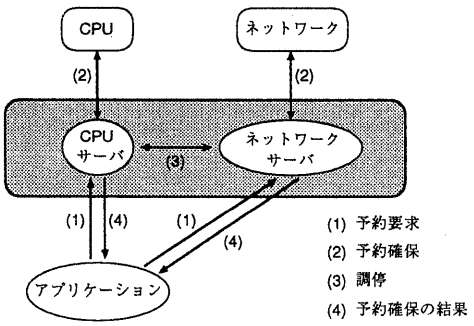

図3: 調停による予約確保

3.4 仮想リソース

リソースの使用効率を上げるために仮想リソースを提案する。これは、リソースは予約確保をされても実際には使用されない場合がある。この点を考慮して、リソースの許容範囲を越える予約確保を行ない、システム全体でのリソースの使用効率を100%に近づける。仮想リソースにより、物理的なリソースの総量よりも広大なリソース空間をユーザに提供する。

4 CPUサーバによるCPUの予約確保

本章では3章の定義に基づいたCPUサーバによるCPUの予約確保について述べる。

4.1 CPUサーバ実現における問題点

CPUサーバの実現における問題点は次の2つである。

1. プロセス処理のCPU占有時間の指定が不可能
2. プロセス処理の周期の保証ができない

ここで言う周期とは、プロセス処理がコンテキストスイッチによりRUN状態からREADY状態になり、再びRUN状態に戻るまでの期間を意味する。

1つめのCPUの占有時間の指定は、音や映像などの連続メディアの処理やメディア間での同期のためには必要不可欠の条件である。このためには、OSは一定間隔でコンテキストスイッチを起こし、周期的にプロセス処理をしなければならない。しかし現在のOSのプロセス処理において、コンテキストスイッチが発生するのは、lockなどによりWAIT状態に遷移する場合や、タイムスライシングによりREADY状態にもどる場合である。このために、コンテキストスイッチが必ずしも一定間隔で起こることは保証できない[6]。

2つめの周期の保証ができない原因として、

- プロセスがシステムコールを使用する
- プロセスが割り込みを受ける

場合があげられる。このとき、タイマーが割り込みをかけることができず、コンテキストスイッチを起こすことができない。

次にこれらの問題点を考慮したCPUサーバについて述べる。

4.2 CPUサーバ

CPUサーバは主に、

- CPUの定量化
- CPUの予約確保

- CPU の管理

の処理を行なう。

CPU の定量化

我々はリソースとして抽象化するために CPU の定量化を行なう。そのためには、CPU の処理能力を数値化して表すことが必要である [7]。実際には CPU の処理能力を時間当たりの処理量で表し、この時間を CPU の周期時間と定義する。例えば、MIPS とは 1 秒当たりの処理数で CPU の処理能力を数値化している。そしてコンテキストスイッチを起こす一定時間を単位時間とし、この単位時間で CPU の処理能力を時分割したものを予約確保のための基本単位（処理量／時間）とする（図 4)。また、CPU を仮想リソースとして扱うために物理的な CPU の処理能力より大きい値を予約確保のために用いる。

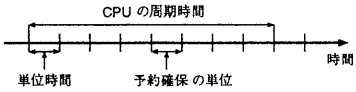

図 4: 時分割した CPU

CPU の予約確保

CPU サーバはそれまでに予約確保された総量を参照し、総量に予約要求の処理量を加えたものが CPU の仮想リソースの処理能力を越えるかどうかを判別し、越えなければ予約確保を行ない、成功通知を返す。そうでなければ、失敗通知を返すだけで予約確保は行なわない。予約確保の要求はメディアが必要とする周期時間あたりの処理量で行なう。

CPU の管理

CPU の管理はプロセス処理の周期性を保証することである。そのためにタイマーによるコンテキストスイッチと時間的制約のある処理（マルチメディア処理）の管理のために差分リスト [8] を用いる。タイマーは、定量化の時の単位時間毎にコンテキストスイッチを強制的に起こす。また、単位時間内に処理が終了しコンテキストスイッチが起こる場合にも差分リストの変更を行なうことで、CPU を効率的に使用する。

4.3 予約確保の不要な処理

本稿におけるリソースの予約確保の方式では、CPU は通常処理よりもマルチメディア処理を優先的に行なう。このために、通常処理とは別に時間的制約のある処理のキューを作成する（図 5)。

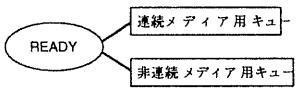

図 5: プロセスの管理

4.4 CPU の予約確保の例

ここでは、CPU の予約確保の例を示す（図 6)。

ここでは、CPU の処理能力を 10MIPS、単位時間を 1ms としている。

1. 1MIPS の予約要求を処理している。この場合は基本単位の 10 個に 1 個の割合で処理を行なうように予約確保している（図 6 (1))。

2. 3MIPS の予約要求を処理している。この場合は基本単位の 10 個に 3 個の割合で処理を行なうように予約確保している（図 6 (2))。

3. 7MIPS の予約要求を処理している。この場合、すでに CPU に対して 4MIPS の予約確保が行なわれているので、7MIPS の予約確保を行なうことができない（図 6 (3))。

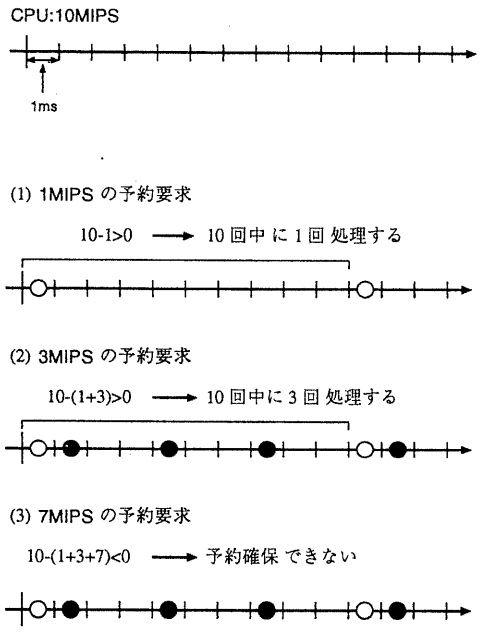

図 6: CPU の予約確保の例

5 CPU サーバの実現

5.1 CPU サーバの開発環境

本稿で述べた CPU サーバの開発環境は以下の通りである。

- IBM-PC 互換機
- ソースが公開されている UNIX (Linux、386/BSD)

現在の開発状況は CPU サーバのための基本研究で、任意の時間でコンテキストスイッチを起こすための機構の実現と周期性を保証するための差分リストを拡張したアルゴリズムの検証を行なっている。また CPU サーバは、既存の OS に対して必要な機能の拡張を行なうことで実現する。

5.2 周期性の保証

本研究では、周期性の保証と複数の処理の同期を実現するためにプロセス処理に差分リスト [8] を用いた。図 7は周期 30msec のプロセス A、周期 5msec のプロセス B、周期 12msec のプロセス C の例である。

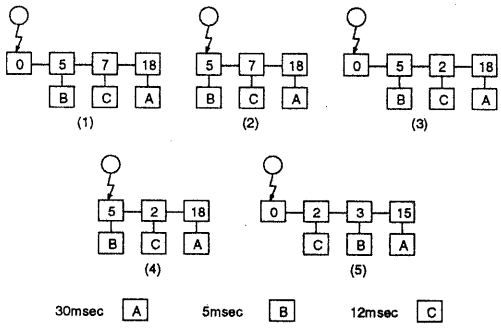

図 7: 差分リスト

1. 左から周期の小さいものを並べていく。上段の数字が時間を表していて、時間は直前のものからの差分となっている。時間の単位は msec である (図 7 (1))。
2. B の処理を行なう (図 7 (2))。
3. そこを時間の基準とし、次の B の周期ポイントを探す。C が 7msec なのでその前に B を挿入する。C の時間は B より 2msec 後なので時間を書き換える (図 7 (3))。
4. 次の処理を行なう (図 7 (4))。
5. 3 と同様にして、次の B の周期ポイントを探し、組み込み、リストの修正を行なう (図 7 (5))。

しかし、この方法では処理にかかる時間や複数の周期の重なりによる遅延が生ずる。そこで周期が重なる場合には、重なった処理の処理時間を

それ以後の時間から減らす。図8の例では処理時間を1として考えた場合であるが、周期の重なりの生じたA,Dの処理時間を上段の時間から減らして遅延を生じないようにしている。

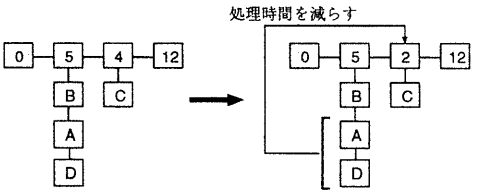

図 8: 差分リストの拡張

処理を切替えるためのコンテキストスイッチは、単位時間で起こすタイマー方式を基本としている。タイマーの精度だが、これは単位時間に大きく関係している。単位時間が小さいと、より細かな周期を実現できるがコンテキストスイッチの回数が増えることによりオーバーヘッドが生じCPUの使用効率が悪くなる。単位時間を大きくすると、オーバヘッドによるロスは減少するが細かな周期が実現できない。この単位時間の設定は重要である。また、タイマー以外にコンテキストスイッチが起こる場合、すなわち単位時間より短い時間で処理が終了した場合には、イベントドリブンによりコンテキストスイッチを起こし、時間的制約のない処理を行なう。これにより、CPUを効果的に使用することができる。我々はこのスケジューリング方式を「実時間ラウンドロビン」と呼ぶ。

5.3 カーネルモードへの対策

カーネルモードの状態の時間を短くするための対策として考えられるのは以下の2点である。

- マイクロカーネル化
- ユーザレベルデバイスドライバ

マイクロカーネル化することとは、分散処理を有効にするためにOSを必要最低限のマイクロカーネル部と、その上のユーザレベルで実現されるOSサーバから構成することを意味する。しかし、このようなOSのI/Oシステムでは OSサーバとデバイスドライバのタスク空間の相違からパフォーマンスが良くないという問題点がある。その対策として、OSサーバにデバイスのメモリ空間をマッピングしデバイスドライバをユーザレベルで動かす方法があり、これによりOSサーバとデバイスドライバ間でのデータ通信のオーバヘッドが減りパフォーマンスが向上することが明らかになっている [9]。我々は、マイクロカーネルとユーザレベルでデバイス操作を行なうことでカーネルモードを短くすることを目指している。

5.4 QOSに対する考慮

最後に、QOS（Quality of Service）[10] を考慮した予約確保の考察を行なう。QOSを考慮するとは、予約確保を唯一の値で指定するのではなく、許容範囲の幅を付加した形にすることである。この方法により指定値での予約確保が不可能な場合、許容範囲を加えた値で予約確保を行なうことが可能になる。

例えばCPUの予約の場合、周期時間に許容範囲の幅を付加して指定を行なうと、指定の周期時間での予約確保ができない場合に許容範囲を加え周期時間を大きくすることにより予約確保を行なうことができる。

QOSを予約確保に組み込むことで、要求に対して臨機応変な対応が可能になる。

6 今後の課題とまとめ

時間的制約のあるマルチメディア処理に対するリソースの予約確保について述べた。現状では、周期性の保証については差分リストが有効であるという結果が得られている。これによりプロセスの管理は行なえる。しかし、CPUの管理のための問題点であるカーネルモードについては何

も行なっていない。これは周期性にも関係があるので早急に検討の必要がある。また、図1の抽象化されたシステムリソースの単位を決めるために、CPUの定量化が必要である。これについては処理量の測定、単位時間の設定のための実測が必要である。カーネルモードの対策のためのマイクロカーネル、ユーザレベルデバイスドライバ、予約確保にQOSを組み込む方法についてはこれから検討していく予定である。

今後はOSへの組み込みおよび拡張についての研究し、それをふまえてCPUサーバの実装を行ない、他のリソースサーバについての研究を目指す。

謝辞

本稿の執筆に際して御助言を頂いた奈良先端科学技術大学院大学の荒木研究室、福田研究室の皆様に感謝致します。

参考文献

[1] D.P.Anderson, "Metascheduling for Continuous Media", ACM Trans. on Computer Systems, Vol.11, No.3, pp.226-252, Aug. 1993.

[2] 岡村, 稲垣, 吉川, 松尾, 田中, 荒木, "QOSに基づいたマルチメディアOS", 情報処理学会 システムソフトウェアとオペレーティング・システム研究会, 94-OS-62-9, March 1994.

[3] 岡村, 吉川, 稲垣, 荒木, "QOS指定可能なマルチメディアモデルの提案", 情報処理学会 マルチメディア通信と分散処理研究会, Nov. 1993.

[4] 稲垣, 岡村, 荒木, "PDE-IIにおけるメディア間同期機構の実現に対する考察", 情報処理学会 全国大会,1H-6, March 1994.

[5] 吉川, 岡村, 荒木, "PDE-IIにおける実時間同期のための周期スレッドの提案", 情報処理学会 マルチメディア通信と分散処理研究会, 94-DPS-64-1, March 1994.

[6] S.J.Leffler,M.K.Mckusick,M.J.Karels,and J.S.Quarterman, 訳者 中村, 相田, 計, 小池, "UNIX 4.3BSDの設計と実装", 丸善株式会社, 1991.

[7] C.W.Mercer, S.Savage, and H.Tokuda, "Processor Capacity Reserves for Multimedia Operating Systems", Technical Report, Computer Science Department, Carnegie Mellon University, CMU-CS-93-157, May 1993.

[8] D.Comer, "OPERATING SYSTEM DESIGN -THE XINU APPROACH-", Scholars Book Co.,Ltd., 1983.

[9] 加藤, 乾, 人見, "マイクロ・カーネル・アーキテクチャにおけるI/O driver実装方法に関する一考察", 情報処理学会 システムソフトウェア・オペレーティングシステム研究会, OS-58-5, 1993.

[10] 船渡, 徳田, "Real-Time Mach 3.0における連続メディアサーバの実験 -QOS制御を取り入れたQuickTime Playerの評価-", 情報処理学会 マルチメディア通信と分散処理研究会, 93-OS-60-11, July 1993.