システムソフトウェアと
オペレーティング・システム 85−12
（２０００．８．４）

連続メディア処理における適応的スケジューリングポリシ
Adaptive Deadline Modification

滝沢 泰久,† 芝 公仁,†† 大久保 英嗣†††

† （株）ATR 環境適応通信研究所
†† 立命館大学大学院理工学研究科
††† 立命館大学理工学部

連続メディアを扱うアプリケーションを周期タスクとしてスケジュールする場合，その時間制約とタスク間の依存関係を満たすためにリアルタイムスケジューリングポリシとリアルタイム同期プロトコルとを組み合わせる方式が有効とされている．しかし，この組み合わせ方式は，予測が困難な変動する環境には適用できない．また，リアルタイム同期プロトコルは密結合による共有メモリモデルを前提としているため，疎結合モデルによるメッセージ通信にはそのまま利用できない．本論文では，単一プロセッサ上の任意の周期タスク間でメッセージ通信する動的な環境において，Parallel Distributed Processing モデルと熱力学的モデルを用いることにより，その時間制約と通信依存関係に適応するスケジューリングポリシを提案し，その性能評価について述べる．

キーワード　スケジューリングポリシ　リアルタイムシステム　マルチメディア

Adaptive Deadline Modification:
An Adaptive Scheduling Policy for Continuous Media Processing

Yasuhisa Takizawa,† Masahito Shiba,†† Eiji Okubo†††

† ATR Adaptive Communication Research Laboratories
†† Granduate School of Science and Engineering, Ritsmeikan Univ.
††† Department of Computer Science, Facutly of Science and Engineering, Ritsmeikan Univ.

Combination schemes of real-time scheduling policies and synchronization protocols are useful for scheduling periodic tasks which manipulate continuous media. However, these schemes can not be applied to dynamic environments.Furthermore, these schemes can not be applied to the message passing, because shared memories can not be used. In this paper, a new scheduling policy based on Parallel Distributed Processing model and thermal dynamics model, which is adaptable for tasks with timing constraints and communication constraints is proposed, and its performance evaluation is presented.

Keywords Scheduling Policy Real-Time System MutilMedia

１．はじめに

マイクロプロセッサの急速な進歩により，パーソナルコンピュータ（以降PC）やワークステーション（以降WS）上で動画や音声に代表される連続メディアを処理するアプリケーション（以降連続メディアアプリケーション）が数多く出現している．連続メディアアプリケーション[4]は，その処理するメディアの特性上，ある時間周期で起動され，次の周期までに処理を完了しなければならない．すなわち，周期タスクとしての時間制約を持つ．マルチメディアシステムでは，このような時間制約を持つ周期タスクが複数実行され，かつ互いに通信を行う場合が多い．そのため，PCやWS上で周期タスクをスケジューリングする場合，リアルタイムスケジューラとリアルタイム同期プロ

トコルとを組み合わせる方式が多く採用されている．しかし，この組み合わせ方式は，タスクの時間制約や依存関係を事前に正確に知る必要があり，また，タスクが入出力処理を行わないことを前提としている．すなわち，任意のタスクが実行され，タスク間の通信や任意のタスクで入出力処理が行われる動的な実行環境には適用できない．

一方，PCやWS上のオペレーティングシステム（以降OS）は，必要最小限のサービスを提供するマイクロカーネルとマイクロカーネル上で動作する各種システムサービスを提供するサーバプロセス群により構成されている．このような構成では，アプリケーションのシステムサービス要求はシステムサーバプロセスとの通信に置き換えられる．また，アプリケーションのタスク間においても同様の方式が多く利用されている．すなわち，タスク間のデータ交換方式は疎結合モデルによるメッセージ通信方式が大部分を占める．従って，このようなデータ交換環境では，リアルタイム同期プロトコルで前提としている密結合（共有メモリ）モデルによる排他制御方式はそのまま利用することができない．

本稿では，連続メディア処理において以上の問題点を解決するために，

- スケジュール環境は，事前に予測が困難な動的な環境である．
- スケジュール問題は，タスクの時間制約と通信依存関係から構成される．
- いくつかのタスクは入出力処理を行う．
- タスク間のデータ交換方式は，疎結合モデルによるメッセージ通信方式である．

の４つを前提として，Parallel Distributed Processingモデル（以降PDPモデル）[6]と熱力学的モデル[3]を用いた適応機能を従来の組み合わせ方式に付加することにより，高いスケジュール可能性を導く適応的スケジューリングポリシAdaptive Deadline Modification（以降ADM）を提案する．また，そのADMの性能評価についても述べる．

以下，２章で周期タスクモデルを定義し，３章でその周期タスクに対する新たなポリシADMを提案する．さらに，４章では，ADMの性能評価について述べ，その有効性を議論する．

２．　周期タスクモデル

連続メディア処理における周期タスクモデルを次のように定義する．

（１）横取り可能

横取り可能なタスクとは，その実行中に，より高い優先度を持つタスクが到着した場合，実行権をより高い優先度を持つタスクに引き渡すことが可能なタスクである．

（２）時間制約

連続メディアを処理する周期タスクは，航空システムや原子力システムに見られる処理完了時刻に厳密なハードリアルタイムタスクと異なり，その処理完了時刻には許容される遅延幅があるソフトリアルタイムタスクとして考える．従って，従来の周期タスクモデルにデッドラインの許容遅延幅を追加し，各時間属性を次のように定める（図１参照）．

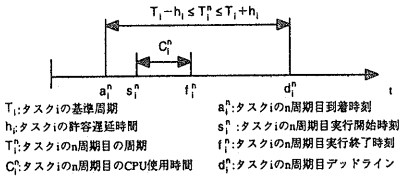

図１　周期タスクの時間制約

（３）メッセージ通信によるデータ交換

マイクロカーネルに基づく構成では，アプリケーションの各種資源要求呼び出しが資源を管理するサーバプロセスへのメッセージ通信に置き換えられる．また，タスク間のデータ交換も同様の手法を多く利用している．このことから，タスク間のデータ交換はメッセージ通信により実施されると考える．

（４）ブロッキング時間

周期タスクが到着し完了するまでの期間で，自分より優先度の低いタスクの実行時間とアイドル時間の和をブロッキング時間する．これは，タスク間通信や入出力処理における資源の競合により発生する．

3．提案スケジューリングポリシADM

資源競合によるブロッキング時間を抑制するリアルタイム同期プロトコルには，Priority Inheritance Protocol（以降PIP)[1]，Priority Celling Protocol（以降PCP)[1]やStack Resource Policy（以降SRP)[9]などがある．PIPは，資源競合が発生した場合，資源使用中のタスクの優先度を資源待ちタスクの中で最も高い優先度と一時的に同一にし，ブロッキング時間を抑える．しかし，連鎖ブロックを発生させる問題を持つ．一方，PCPやSRPは連鎖ブロックを解決した手法であるが，事前情報が必要であり，また，入出力処理がないことを前提としている．従って，動的な環境でのリアルタイム同期プロトコルとしてPIPを使用する．しかし，上記のようにPIPでは連鎖ブロックが発生する問題点を持っており，さらに入出力処理がある環境ではその問題が顕在化する可能性が高い．

提案ポリシADMは，入出力処理が行われ，かつ動的な環境において，タスク間通信や資源競合によるブロッキング時間を抑制するため，適応機能とリアルタイムスケジューラ・リアルタイム同期プロトコル機能（以降RTスケジュール機能）から構成される（図2）．適応機能は，資源競合を可能な限り抑制するタスク優先度をタスク到着時（実行直前）に算出する．一方，RTスケジュール機能は算出された優先度を基に実行順を決め，実行時に資源競合が発生した場合，そによるブロッキング時間を抑制する．以下，本章ではADMの各機能について説明する．

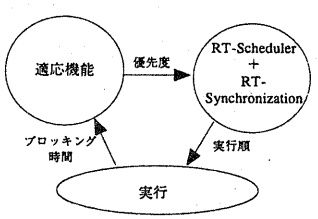

図2　提案ポリシADMの構成

3．1　RTスケジュール機能

リアルタイムスケジューラは，高いスケジュール可能性を持ち，また最適なスケジューラであるEarliest Deadline First（以降EDF)[2]を用いる．

また，リアルタイム同期プロトコルは事前情報が不要なPIP[1]を用いる．しかし，前述のようにリアルタイム同期プロトコルは，メッセージ通信によるデータ交換環境にそのまま適用できない．そこで，RT-Mach[5]で実現されているReal-Time IPC（以降RT-IPC)[7]と同等の機能を実装して，これを用いる．RT-IPCは，メッセージ通信機能にPIP機能を追加し，メッセージ通信時の資源競合におけるブロッキング時間を減少させる．

EDFにおけるスケジュール可能性判定式は，次式となる．

$$\sum_{k=1}^{i}\frac{C_k}{T_k}+\frac{B_i}{T_i}\le 1 \qquad (1)$$

ただし，Biはタスクiの最悪ブロッキング時間，タスクkはタスクiよりも優先度が高いタスクである．この式から分かるように，ブロッキング時間を減少させるとスケジュール可能性が高まる．従って，EDFとRT-IPCは，適応機能で算出された優先度において，タスク実行時に資源競合が発生した場合，それによるブロッキング時間を抑制し，スケジュール可能性を高める．

3．2　適応機能

メッセージ通信によるデータ交換環境において，RTスケジュール機能により発生する資源競合の状況は，メッセージ通信時のブロッキング時間として現れる．適応機能では，タスクの実行により計測されたこのブロッキング時間とタスク間の通信関係から，RTスケジュール機能により発生し得る資源競合のブロッキング時間を抑制するように，タスク優先度の修正を時間制約の範囲内で行う．これにより，高いスケジュール可能性を導く．

（1）タスク間通信におけるブロッキング時間

送信タスクsと受信タスクrの2つのタスクが通信する場合のブロッキング時間について考える．送信側タスクsの送信タイミング，および受信側タスクrの受信タイミングは任意とする．

送信側タスクsにブロッキング時間が発生する

場合，そのブロッキング時間は，送信側タスクおよび受信側タスクのデッドライン時刻の関係により，次式となる．

$$B_s^n = \begin{cases} (s_r^k + C_{r,rcv}^k) - (s_s^n + C_{s,snd}^n) & d_s^n < d_r^k \\ s_r^k - (s_s^n + C_{s,snd}^n) & d_s^n \geq d_r^k \end{cases} \quad (2)$$

だだし，B_s^nは送信側タスクsのn周期目のブロック時間，$C_{s,snd}^n$は送信側タスクsのn周期目での送信開始時刻までの実行時間，$C_{r,rcv}^k$は受信側タスクrのk周期目での受信開始時刻までの実行時間，d_r^kは受信側タスクrのk周期目のデッドライン時刻，d_s^nは送信側タスクsのn周期目のデッドライン時刻である．

一方，受信側タスクrにブロッキング時間が発生する場合，そのブロッキング時間は，送信側タスクおよび受信側タスクのデッドライン時刻の関係により，次式となる．

$$B_r^k = \begin{cases} s_s^n - (s_r^k + C_{r,rcv}^k) & d_r^k \geq d_s^n \\ (s_s^n + C_{s,snd}^n) - (s_r^k + C_{r,rcv}^k) & d_r^k < d_s^n \end{cases} \quad (3)$$

ただし，B_r^kは受信側タスクrのk周期目のブロッキング時間である．

いずれの場合も，ブロッキング時間はタスクの実行開始時刻，送信開始時刻および受信開始時刻に依存し，式（2）（3）から次のように考える．

- 双方の実行開始時刻を近付けるとブロッキング時間が小さくなる．
- ブロッキング時間が発生しているタスクにおいて，実行開始時刻が相対的に遅くなるとブロッキング時間が減少する．
- ブロッキング時間が発生していないタスクにおいて，実行開始時刻が相対的に早くなると，通信相手側タスクのブロッキング時間が減少する．

（2）入出力処理におけるブロッキング時間

入出力処理は，送信側をタスクとし，受信側を入出力機器としたメッセージ通信と考える．従って，入出力機器でデータが準備できていない場合はブロッキング時間がタスクに発生し，次式となる．

$$B_s^k = IO_x - (s_s^k + C_{s,snd}^k) \quad (4)$$

ただし，IOxは入出力データ到着時刻である．この式から，実行開始時刻を遅くするとブロッキング時間が減少する．

（3）ブロッキング時間を減少させるデッドライン

前述のブロッキング時間は，式（2）（3）（4）より，通信するタスクの実行開始時刻に依存する．

一方，EDFでは，デッドラインが早いタスクほど優先度が高い．そのため，デッドラインが早い（優先度が高い）タスクは実行開始時刻が早まり，デッドラインが遅い（優先度が低い）タスクは実行開始時刻が遅くなる．すなわち，デッドラインによりタスクの実行開始時刻を制御できる．そこで，タスク実行により計測されたブロッキング時間に基づき，（1）（2）で述べたように実行開始時刻を制御するため，時間制約内でデッドラインを修正する．この修正されたデッドラインを適応デッドラインと呼び，次のように，実行－その結果から修正－再実行をくり返し，変動するブロッキング時間に対応してタスク到着時に算出する．

- 最初の周期では，ブロッキング時間は不明であるため，互いに通信するタスクの実行開始時刻を時間制約範囲内で，可能な限り近付ける適応デッドラインを算出する．
- 2周期目以降は，前周期の実行の結果生じたブロッキング時間に応じて適応デッドラインを修正算出する．すなわち，ブロッキング時間がある場合は，相対的に開始時刻が早いと判断し，実行開始時刻を遅らす適応デッドラインを算出する．ブロッキング時間がない場合は，相対的に開始時刻が遅いと判断し，実行開始時刻を早くする適応デッドラインを算出する．

以上により，算出された適応デッドラインはブロッキング時間を減少させ，スケジュール可能性を高める．

（4）PDPモデルの応用

以上の処理は2つのタスクにおける処理であるが，実際の実行環境ではこのような通信関係が複数混在する．すなわち，扱う問題は，2つのタス

クの通信関係が多重にからみ合う同時多重制約問題である．

ADMは，この多重制約問題を処理するために，認知科学におけるPDPモデルを応用する．PDPモデルは，相互に関連する複数の制約に対して高い充足状態を算出する処理モデルである．このPDPモデルを，タスク間の通信関係と計測されたブロッキング時間から次のような制約に対して動作さる．

・通信するタスク同士は協調して，重要度を上げ下げする．
・通信しないタスク同士は抑制して，一方が重要度が高い場合は低く，一方が重要度が低い場合は高くする．
・ブロッキング時間がある場合は，重要度を下げる．
・ブロッキング時間がない場合は，重要度を上げる．

これにより，上記制約を充足する各タスクの重要度を探し出し，この重要度と時間制約から適応デッドラインを算出する．

（5）熱力学モデルの付加

PDPモデルは，初期状態に依存してある充足状態（状態とは各タスクの重要度により表される）に静止する特徴を持っている．一方，動的なタスク実行環境において，ブロッキング時間は常に変化する．PDPモデルは，この変化に応じて，1つの状態に静止することなく，いくつかの制約充足度の高い状態間を移動する必要がある．このため，PDPモデルに熱力学モデルを付加し，温度による物質の熱揺動（高温で分子間の結合が弱まり不安定，低温で分子間の結合が強まり安定する性質）をPDPモデルの処理において擬似する．すなわち，ブロッキング時間が増大した場合は温度を高くし，PDPモデルの処理動作を大きく振動させ新しい状態を見つける可能性を高める．一方，ブロッキング時間が減少する場合は温度を低くし，PDPモデルの処理動作を現在の状態の近傍で小さく振動させ，その状態を維持する．

この温度による熱揺動制御により，PDPモデルを変動する環境に連続的に動作するようにし，変動するブロッキング時間に適応して，そのブロッキング時間を減少させるタスクの適応デッドラインを探し続けることを可能とする．

4． 性能評価

ADMを，我々が分散OSの構成法の研究のため開発している分散OS Solelc上に実装し，性能評価を行った．本章では，その結果について述べる．

4．1 性能評価環境

ハードウェアの環境としては，次のものを使用した．

・使用機種　エプソン社製 Endeavor ATX-7000
・プロセッサ　Pentium-S 200MHz
・キャッシュ　512KB
・主記憶　128MB

ソフトウェア環境としては，提案ポリシADMの実装環境であるOSにはSolelcを使用した．今回の評価では，Solelcのディスパッチ機能とメッセージ通信機能のみを使用し，スケジューリング分解能は1 msecとした．また，コンパイラはGNU C v2.4を使用した．

4．2 評価方法と評価タスクセット

評価はメッセージの到着順に処理するFIFO，メッセージの優先度順に処理するPRIQおよびメッセージの優先度順に処理し，受信タスクの優先度をメッセージキュー内の最も高い優先度にするPIPの3つのRT-IPCに適応機能を付加した場合と付加しない場合の合計6つのケースを比較する．

（1）評価タスクセット1

評価タスクセット1は，クライアント／サーバモデルに基づく要求時処理型である．これのタスクセットは，1つの負荷タスク，6つのクライアントタスク，入力データサーバタスク，出力データサーバタスク，入力および出力の擬似ドライバにより構成される（図3）．クライアントタスクは入力データサーバタスク通信し，データ処理後に出力データサーバタスクと通信する．サーバタス

クはクライアントタスクの要求の到着により起動され，擬似ドライバへ処理要求を行う．擬似ドライバはデータ量に応じて遅延を発生させる．サーバタスクのデッドラインは，適応機能を付加しない場合は要求クライアントタスクのデッドラインと同一とし，適応機能を付加した場合は要求クライアントタスクの時間制約から要求の到着時にサーバタスクの適応デッドラインを算出し，これを用いる．また，クライアントの処理時間はランダムに変動させ，クライアントの処理時間に連動しサーバタスクの処理時間および擬似ドライバでの遅延時間を変動させる．

各タスクの通信関係と時間制約を図３に示す．図中の{}はタスクの時間制約を示す．{}内の各項目は，左から基準周期，許容遅延時間，起動時刻，平均CPU使用時間，最小CPU使用時間，最大CPU使用時間を示す．単位はすべてmsである．また，各タスクはSoloelcのスレッドに対応付ける．

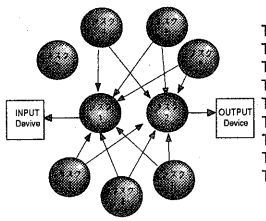

図３　評価タスクセット１

（２）評価タスクセット２

評価タスクセット２は，パイプラインモデルによる前処理型／予約処理型である．このタスクセットは，連続メディアストレージサーバ[8]などの処理において使用されるモデルである．最初のタスクがデータを生成し，次のタスクに渡して別の処理を行う．データはタスクからタスクへと流れ，順次処理される．クライアントタスクと入力サーバタスクの関係からみた場合，クライアントタスクとサーバタスクが１対１に対応し，サーバタスクは前処理を行い，これをクライアントタスクに引き渡す．出力サーバタスクの場合は，同様にクライアントタスクと１対１に対応し，クライアントタスクの処理後に処理を開始する．すなわち，サーバの処理は複数タスク（マルチスレッド）により構成される．このタスクセットは，１つの負荷タスク，３つのクライアントタスク，３つの入力サーバタスク，３つの出力サーバタスク，１つの入力擬似ドライバ，１つ出力擬似ドライバにより構成される（図４）．

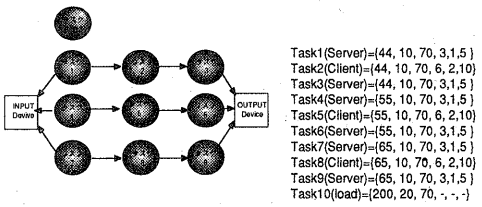

図４　評価タスクセット２

各タスクの通信関係と時間制約を図４に示す．

４．３　スケジューリング成功率

各評価タスクセットに関して，負荷状況，入出力遅延状況と起動位相状況に応じて周期タスクのスケジューリング成功率の変動を前述の６つのケースについて比較評価した．スケジューリング成功率は，到着総周期タスク数に対するデッドラインまでに処理が完了したタスク数の割合とする．

（１）評価タスクセット１の結果

６つのケースの負荷状況に応じたスケジューリング成功率を図５に示す．この図から分かるように，スケジューリング成功率はRT-IPCの種類によって異なり，PIPを用いたケースで適応機能の有無に関わらず，スケジューリング成功率が高い．これは，入出力遅延が比較的少ないため（入出力遅延時間はサーバタスクの処理時間の1.5倍），適応機能が資源競合を抑制する適応デッドラインを十分に算出できていないためと考える．すなわち，資源競合が多く発生し，そのためPIPの機能を有するケースがスケジュール成功率が高くなると考える．

６つのケースの入出力遅延に応じたスケジューリング成功率を図６に示す．入出力遅延時間はサーバタスクの実行時間に比例して設定し，横軸

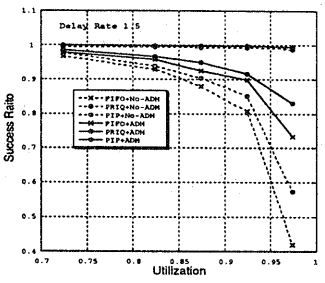

図5　評価タスクセット1の負荷状況に応じたスケジューリング成功率

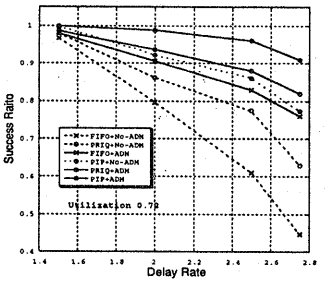

図6　評価タスクセット1の遅延状況に応じたスケジューリング成功率

のDelay Rateはその比例定数を示す．この図から分かるように，スケジューリング成功率は適応機能を用いたケースで高くなる．これは，入出力遅延時間が比較的大きくなると，適応機能が有効に動作し，資源競合を抑制する適応デッドラインを算出していると考える．

（2）評価タスクセット2の結果

6つのケースの負荷状況に応じたスケジューリング成功率を図7に示す．この図から分かるように，スケジューリング成功率はRT-IPCの種類に全く依存せず，適応機能を用いたケースで常に高い．スケジューリング成功率がRT-IPCの種類に全く依存しないのは，サーバタスクとクライアントタスクが1対1に対応している（マルチスレッドサーバ）ため，サーバタスクとの通信によりクライアントタスク間での競合が発生しないことによる．従って，RT-IPCの3つの機能は同一の結果となる．一方，サーバタスクと擬似ドライバとの通信で競合は発生する．しかし，擬似ドライバはカーネルタスクであり，常に最も高い優先度であることから，RT-IPCは擬似ドライバには機能しない．適応機能は，擬似ドライバとの通信により発生したブロッキング時間から適応デッドラインを修正することから，入出力遅延にも機能する．従って，適応機能を用いたケースで常にスケジュール成功率が高いと考える．

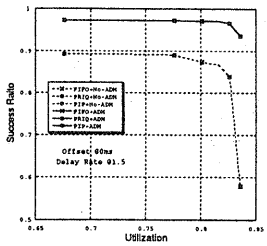

図7　評価タスクセット2の負荷状況に応じたスケジューリング成功率

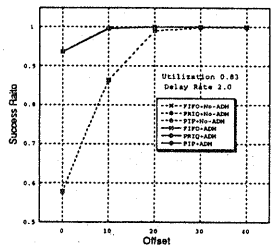

図8　評価タスクセット2の起動位相に応じたスケジューリング成功率

6つのケースの起動位相に応じたスケジューリング成功率を図8に示す．この図から分かるように，スケジューリング成功率は適応機能を用いたケースで常に高い．これは，図7の場合と同じ理由と考える．

以上の（1）と（2）結果から，適応機能は，従来の方式では考慮されていない入出力遅延やタスク間通信による競合に対して十分に機能し，PIP機能と組み合わせることにより高いスケジュール可能性を導くことが分かる．

5．4　スケジューリングオーバヘッド

ADMの計算量は，PDPモデルに依存し，スケジュール対象のタスク総数に比例する．従って，提案ポリシのスケジューリングオーバヘッドは，

タスク総数が増えると高くなる．そのオーバヘッドの評価結果を図９に示す．この図から分かるように，タスク総数に応じて，直線的にオーバヘッドは高くなる．プロセッサがPentium-S 200MHzを用いた場合，タスク総数１０で平均約72 μ secである．現在の一般的なプロセッサのクロックは評価環境の数倍である．このことから，一般的なPCは，評価環境の２倍のクロックを持つと仮定すると，１０個のタスクのスケジューリングで約36 μ secの計算時間が必要となる．ただし，この負荷は，適応デッドデッドラインを計算するタスクの到着時のみに必要であり，それ以外のコンテキストスイッチには不要である．一方，マルチメディア環境で扱う音声，アニメーションおよびビデオは文献[4]によると，最短周期が75Hz，すなわち約13msecとなっている．この最短周期のタスクを１０個スケジュールする場合，一般的なPC環境において１周期の適応デッドライン計算に必要となる時間は，上記で示した約36 μ secである．この時間のタスクの周期に占める割合は，高々0.28％に過ぎない．また，タスクを１００個スケジュールする場合でも，2.8％ほどである．

以上のことから，ADMはタスク総数に比例してオーバヘッドは高くなるが，タスクのスケジューリング周期に占める割合は極微量であると言える．従って，ADMは，周期的タスクのスケジューリングに十分に適用できると考える．

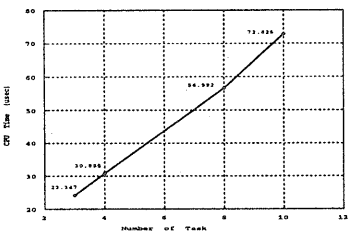

図９　スケジューリングオーバーヘッド

５．　おわりに

本稿では，従来方式において問題であった１）スケジュール環境が事前に予測が困難な動的な環境である，２）スケジュール問題がタスクの時間制約と通信依存関係により構成される，３）いくつかのタスクは入出力処理を行う，４）タスク間のデータ交換方式は疎結合モデルによるメッセージ通信方式である，といった４つの問題を解決しするADMを提案し，さらに，ADMは高いスケジューリング可能性を導くポリシであることを示した．

参考文献

[1]L. Sha, R. Rajikumar, and J. P. Lehoczky: Priority Inheritance Protocols: An Approach to Real-Time Synchronization, IEEE Trans. on Computer, Vol.39, No.9, pp 1175-1185 (1990).

[2]C. L. Liu and J. W. Layland: Scheduling algorithms for multiprograming in a hard real time enviroment, J. ACM, Vol.20, No.1, pp.46-61 (1973).

[3]R. Yavatkar and K. Lakshman: Optimization by Simulated Anneling, Science, Vol.220, pp. 671-680 (1983).

[4]B. Furth: Multimedia Systems: An Overview, IEEE Multimedia, Vol.1, No.1, pp. 47-59 (1994).

[5]H.Tokuda, T. Nakajima and P. Rao: Real-time mach: Towards a predictable real-time system, Proc. USENIX Mach Workshop, pp. 1-10 (1990).

[6]D. E. Rumelhart, J. L. McCelland and the PDP Research Group: PARALLEL DISTRIBUTED PROCESSING, The MIT Press (1986).

[7]H.Tokuda, T. Nakajima and H.Tokuda: RT-IPC An IPC Extension for Real-Time Mach, In the Proceedings of 2nd Microkernel and Other kernel Architectue, USENIX (1993).

[8]D.P.Anderson, Y.Osawa, R.Govindan: A File System for Continuous Media, ACM Transactions on Computer Systems, Vol.10, No.4, p311-337(1992).

[9]T.P.Baker: Stack-Based Scheduling of Realtime Processes, J.Real-time Systems, Vol.3, No.1, pp.67-99 (1991).