システムソフトウェアと
オペレーティング・システム
（２００１．６．２８）
87−2

適応的スケジューリングポリシ Adaptive Deadline Modification の性能評価

滝沢泰久[†]　芝　公仁[††]　大久保英嗣[†††]

動画や音声に代表される連続メディアを扱うストリーム処理タスクをスケジュールする場合，それらの時間制約を満たすために，最悪処理時間に基づいたリアルタイムスケジューラが利用されてきた．しかし，多くの連続メディアデータは，圧縮・伸長処理や差分処理などにより，処理時間は必ずしも一定とはならない．このような連続メディアデータを従来の最悪処理時間に基づいた方式で処理すると，過度な CPU 利用率の予約のために，システム全体の CPU 利用率が大きく低下する．我々は，以上の問題を解決する適応的スケジューリングポリシ Adaptive Deadline Modification の提案を行っている．Adaptive Deadline Modification は，連続メディア資源モデルである Linear Bounded Arrival Process に変動処理量の変更を加えたモデルを基本とし，そのモデルに Parallel Distributed Processing モデルと熱力学モデルを適用している．本稿では，Adaptive Deadline Modification の概要と性能評価について述べる．

A Performance Evaluation of An Adaptive Scheduling Policy: Adaptive Deadline Modification

Yasuhisa Takizawa,[†] Masahito Shiba[††] and Eiji Okubo[†††]

Real-time scheduling policies based on the worst-case execution time have been used for stream processing tasks which manipulate continuous media such as voice, audio, video and animation. However, the processing time for continuous media stream dynamically changes as the result of compressions, extensions and differences between data. Therefore, conventional schemes cause excessive reservations of the processing time since the processing time of each continuous datum is shorter than its worst-case execution time. In order for solution of those problems, we have been proposed a new scheduling policy which is adaptable for stream processing tasks with timing constraints and processing delay. The proposed policy is based on the model which modifies Linear Bounded Arrival Process, and applies Parallel Distributed Processing model and thermal dynamics model to this model. In this paper, a summary of the proposed policy and its performance evaluation are described.

1. はじめに

音声や動画に代表される連続メディアを扱うアプリケーションは，処理するメディアの特性上，周期タスクモデルに基づいた時間制約を持つ．マルチメディアシステムにおける連続メディア処理は，このようなタスクが複数実行され，かつ直列にデータ通信を行うストリーム処理により行われる場合が多い．ストリーム処理タスクをスケジュールする場合，その時間制約を満たすために，最悪実行時間と周期タスクモデルに基づいて CPU 利用率を予約した上で，リアルタイムスケジューラを用いる方式が有効とされている．

一方，連続メディアデータは，入出力装置から固定周期で生成／消費され，そのデータ量は動画データなどの圧縮／伸張や差分処理に見られるように可変量となる．すなわち，連続メディアデータは，Variable Bit Rate（以降 VBR）データであると言える．このような連続メディアデータのストリーム処理は，順次処理されるパイプラインモデルにより行われる（**図 1**）．パイプライン上のタスクは VBR データを処理するため，その処理遅延も可変となる．すなわち，VBR データを処理するタスクの処理遅延は可変となり，パイプライン上のタスクにおけるデータの到着は周期性を失う．

† （株）ATR 環境適応通信研究所
ATR Adaptive Communications Research Laboratories
†† 立命館大学大学院理工学研究科
Graduate School of Science and Engineering, Ritsumeikan Univ.
††† 立命館大学理工学部情報学科
Department of Computer Science, Facutly of Science and Engineering, Ritsumeikan Univ.

図 1 連続メディアデータのストリーム処理
Fig. 1 Stream processing for continuous media data.

このため，周期タスクモデルをストリーム処理に適用すると，タスクにデータ未到着による待ち時間が発生しスケジュール可能性が低下する．これを回避するためには，パイプライン上の隣接するタスク間に十分な初期位相が必要である．従って，エンド－エンドの処理遅延が大きくなる．また，最悪実行時間に基づき CPU 利用率を予約することは，VBR データ処理環境では過度な予約量となり，システム全体での CPU 利用率が大きく低下する．

我々は，以上の問題点を解決するために，以下の 4 つを前提として，従来の方式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）の提案 [7] を行っている．

- 最悪実行時間より短い時間を CPU 使用時間と想定する．
- 各タスクのメッセージ処理に必要な CPU 使用時間は，ランダムに変動する．
- 各タスクは，連続メディアデータをパイプラインモデルにより処理する．
- 連続メディアデータは，その入出力装置から VBR ／固定周期で生成／消費される．
- 複数のストリーム処理が混在する．

ADM では，ストリーム処理モデルを連続メディア資源モデル Linear Bounded Arrival Process[3]（以降 LBAP）に VBR データ処理のための変更を加えたモデルとし，Parallel Distributed Processing モデル [2]（以降 PDP モデル）と熱力学的モデル [1] を動作メカニズムとしている．本稿では，ADM の性能評価について述べる．

以下，２章で ADM のストリーム処理モデルを説明し，３章で ADM のスケジューリングポリシを述べる．４章で ADM の性能評価結果について述べ，その有効性を議論する．

2. ADM におけるストリーム処理モデル

本章では，LBAP と LBAP に基づいた従来のストリーム処理モデルについて説明する．その上で，ADM の VBR ストリーム処理モデルを説明する．

2.1 LBAP

LBAP では，処理すべきデータを以下の３つのパラメータにより特徴付けたメッセージとしている．

R^{max}： maximum message rate (messages/second)
W^{max}： maximum workahead (messages)
S^{max}： maximum message size (bytes)

LBAP では，パラメータ W^{max} により示されるバースト的なデータ到着により，データレートは短期間 $R^{max}S^{max}$ を超えることを許容している．このバースト的なデータ到着に伴う未処理メッセージ数を次のように定義する．

$$w_n(m_0) = 0$$
$$w_n(m_i) = max(0, w_n(m_{i-1}) - (a_n(m_i) - a_n(m_{i-1}))R^{max} + 1)$$

ただし，$w_n(m_i)$ はタスク n における i 番目のメッセージ到着時の未処理メッセージ数，$a_n(m_i)$ はタスク n における i 番目のメッセージ到着時刻である．従って，タスク n における W^{max}，すなわち W_n^{max} は次のように定義できる．

$$W_n^{max} = \max_i(w_n(m_i)) \quad (1)$$

LBAP では，メッセージが処理対象となる論理的時刻をタスク n における i 番目のメッセージのメッセージ論理到着時刻 $l_n(m_i)$ として，次のように定義する．

$$l_n(m_i) = a_n(m_i) + w_n(m_i)/R^{max}$$

また，LBAP では，タスク n における i 番目のメッセージにおける処理遅延時間 $D^a(m_i)$，論理処理遅延時間 $D^l(m_i)$ およびデッドライン時刻 $d_n(m_i)$ を，次のように定義する（**図 2** 参照）．

$$D_n^a(m_i) = a_{n+1}(m_i) - a_n(m_i) \quad (2)$$

$$D_n^l(m_i) = l_{n+1}(m_i) - l_n(m_i) \quad (3)$$

$$d_n(m_i) = l_n(m_i) + D_n^{max} \quad (4)$$

ただし，D_n^{max} は，前述のパラメータにより資源予約した場合の最大の論理処理遅延時間であり，次のように定義できる．

$$D_n^{max} = \max_i(D_n^l(m_i))$$

LBAP では，このように特徴付けられたメッセージストリームの処理を，次のように行う．

- タスクのデッドライン時刻に基づき，EDF により実行順を決定する．
- 前述のように定義されたタスクがパイプラインモデルにより複数接続される．

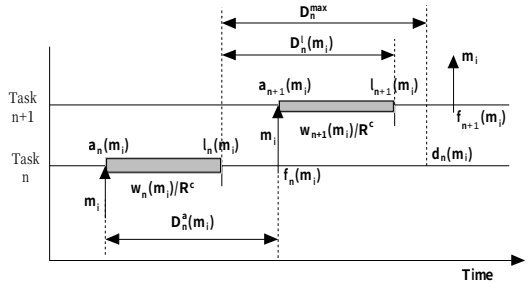

図 2 LBAP における時間属性
Fig. 2 Timing attributes on LBAP.

- タスクは，メッセージの到着時刻に起動される．
- タスクは，メッセージ処理完了時刻に未処理メッセージがある場合，処理完了したメッセージのデッドライン時刻を待たずに直ちに起動される．

2.2 LBAP に基づく従来の処理モデル

従来の方式 [4] では，ストリームデータを一定のメッセージの到着レートと最悪実行時間により特徴付け，CPU 利用率を予約している．すなわち，従来の方式は，最悪実行時間に基づいた Constant Bit Rate（以降 CBR）ストリーム処理として考えられる．このような CBR ストリーム処理を，LBAP のパラメータを用いて表すと，次のようになる．

R^c: constant message rate (messages/second)
W^{max}: maximum workahead (messages)
C^{max}: maximum processing time for a message (seconds/message)

この定義により，未処理メッセージ数，およびメッセージ論理到着時刻は，次のようになる．

$$\begin{aligned} w_n(m_0) &= 0 \\ w_n(m_i) &= max(0, w_n(m_{i-1}) \\ &\quad - (a_n(m_i) - a_n(m_{i-1}))R^c + 1) \end{aligned} \tag{5}$$

$$l_n(m_i) = a_n(m_i) + w_n(m_i)/R^c \tag{6}$$

その他の定義は，2.1 節と同様である．上記のように，定義された時間属性において連続メディアデータ出力装置の時間制約を保証するには，隣接するタスク間に十分な初期位相が必要である．その初期位相は，次のようになる．

$$a_n(m_i) \geq a_0(m_0) + \sum_{j=0}^{n-1}(W_j^{max} + 1)/R^c \tag{7}$$

従って，出力装置の時間制約を保証する場合，最悪実行時間に基づく CBR ストリーム処理モデルは，大きなエンドーエンドの遅延時間が必要となる．また，最悪実行時間に基づく CPU 使用量の予約は，過度な CPU 予約量となり，システム全体での CPU 利用率が低下する．

2.3 VBR ストリーム処理モデル

前節の問題を解決するために，ADM では，メッセージ処理時間を最悪実行時間 C^{max} より短い任意の時間 C^{req} とし，また，最大未処理メッセージ数 W^{max} の代わりに実測の未処理メッセージ数 b を用いる．そのパラメータを次に示す．

R^c: constant message rate (messages/second)
b: actual workahead (messages)
C^{req}: request processing time for a message (seconds/message)

レート R^c で最悪実行時間より短い時間 C^{req} により，CPU 利用率として予約している場合，すべてのメッセージの処理時間が $1/R^c$ 以下であることは保証されない．従って，式（5）,（6）のような予測は困難である．そこで，VBR データ処理モデルにおける時間属性は，変動する処理時間を考慮して定義する．

未処理メッセージ数の定義は，$w_n(m_i)$ に代わり，任意の時刻に実測された未処理メッセージ数を用いる．タスク n において，時刻 t に実測された未処理メッセージ数を $b_n(t)$ と表記し，実効未処理メッセージ数と呼ぶ．

メッセージ論理到着時刻 $l_n(m_i)$ に代わり，メッセージ実効到着時刻 $e_n(m_i)$ を定義する．$l_n(m_i)$ は，メッセージが初めて処理対象と予測される論理時刻である．このことから，メッセージ実効到着時刻 $e_n(m_i)$ は，タスク n において，i 番目のメッセージが，初めて処理対象となった実際の時刻とし，次のように定義する．

$$\begin{aligned} e_n(m_0) &= a_n(m_0) \\ e_n(m_i) &= max(a_n(m_i), f_n(m_{i-1})) \end{aligned}$$

ただし，$f_n(m_{i-1})$ は，タスク n における $i-1$ 番目のメッセージの処理完了時刻である．

次に，デッドライン時刻について定義する．時刻 $e_n(m_i)$ では，$D_n^l(m_i)$ は不明であるが，パイプラインモデルの隣接するタスク n と $n+1$ において，タスク n は，i 番目のメッセージの処理を，$f_{n+1}(m_{i-1})$ までに完了すればよいと考えられる．従って，デッドライン時刻は，次のように考える．

$$d_n(m_i) = f_{n+1}(m_{i-1})$$

時刻 $e_n(m_i)$ では，$f_{n+1}(m_{i-1})$ もまた不明である．しかし，周期 $1/R^c$ 内でメッセージの処理を完了しなければ，後続タスクの周期的時間制約を満たせなくなる可能性が高まる．このことから，メッセージ処理が $1/R^c$ 内で完了すると仮定し，時刻 $e_n(m_i)$ におけるタスク

$n+1$ の実効未処理メッセージ数から，$f_{n+1}(m_{i-1})$ を次のように想定する．

$$\bar{f}_{n+1}(m_{i-1}) = e_n(m_i) + b_{n+1}(e_n(m_i))/R^c$$

従って，デッドライン時刻 $d_n(m_i)$ を，次のように定義する．

$$d_n(m_i) = e_n(m_i) + b_{n+1}(e_n(m_i))/R^c \qquad (8)$$

その他の定義は，2.1 節と同様である．

3. ADM のスケジューリングポリシ

本章では，VBR ストリーム処理モデルの特性を列挙し，それを考慮することにより，スケジュール可能性を高めるスケジューリングポリシについて述べる．

3.1 VBR ストリーム処理モデルにおける特性

最悪実行時間より短い処理時間に基づき CPU 利用率を予約する場合の VBR ストリーム処理モデルの特性は，著者らの報告 7) により，次のようになる．

[特性 1] VBR ストリーム処理モデルにおいて，最悪実行時間より短い時間を，パラメータ C^{req} と R^c により CPU 利用率として予約する場合，メッセージの処理完了時刻は，以下の条件で，デッドライン時刻を満たす．

$$b_{n+1}(e_n(m_i))/R^c \geq D_n^a(m_i)$$

[特性 2] 特性 1 を満たす VBR ストリーム処理モデルにおいて，以下の条件を満たすならば，メッセージ処理に必要な CPU 利用率は，予約した CPU 利用率を超えない．

$$b_{n+1}(e_n(m_i)) \geq C_n(m_i)/C_n^{req}$$

ただし，$C_n(m_i)$ は，タスク n における i 番目のメッセージ処理に使用する CPU 時間である．

[特性 3] VBR ストリーム処理モデルにおいて，最悪実行時間より短い時間を，パラメータ C^{req} と R^c により，CPU 利用率として予約している場合，パイプライン上の隣接するタスクのメッセージ処理遅延時間が以下を満たすならば，連続メディアデータ出力装置の時間制約が満たされることが保証される．

$$\begin{aligned}\sum_{k=0}^{i}\{&\sum_{j=0}^{n-1}(D_j^a(m_{k+1}) - D_{j+1}^a(m_k)) \\ &+ (D_{in}^a(m_{k+1}) - D_0^a(m_k)) \\ &+ (D_n^a(m_{k+1}) - D_{out}^a(m_k))\} \leq 0\end{aligned}$$

ただし，$D_{out}^a(m_k)$ は出力装置における k 番目のメッセージ処理遅延時間，$D_{in}^a(m_{(k+1)}$ は入力装置における $k+1$ 番目のメッセージ処理遅延時間である．

3.2 ブロッキング時間の抑制

ストリーム処理において，パイプライン上の隣接するタスク間でのメッセージ通信は非同期通信により行われる．そのため，隣接するタスク間にはメッセージを保存する FIFO のキューが用意される．この各タスク間のキューもまた，使用量には限度がある．タスク間のキューが使用量の限度に達している場合，送信側タスクのメッセージ送信はキューが空くまで待機する．これにより，タスクにはブロッキング時間が発生する．

ADM では，ブロッキング時間をメッセージ実効到着時刻 $e_n(m_i)$ からメッセージ処理完了 $f_n(m_i)$ までの期間で，当該タスクの優先度より低い優先度のタスクが実行した時間とアイドル時間の和をブロッキング時間とする．このブロッキング時間は優先度逆転 (priority inversion) を引き起こし，スケジュール可能性を低下させる．

ADM では，著者らが文献 8) に示したブロッキング時間を抑制する制御を用いる．すなわち，前メッセージ処理でブロッキング時間が発生したタスクは相対的に実行開始時刻が早いと考えられるので，実行開始時刻を遅くするために優先度を低くする．これにより，スケジュール可能性の低下を防ぐ．

3.3 スケジュール可能性を高める制御

スケジュール可能性を高めるため，VBR ストリーム処理モデルの特性とブロッキング時間の抑制を満たすように優先度を次のように制御する．

- パイプライン上のタスクの実行機会が同じになるように，各タスクの優先度を連動させる．
- 特性 3 を満たすため，タスク n のメッセージの実効到着時刻における前周期のメッセージ処理遅延時間と後続タスク $n+1$ の同じ時刻における前周期のメッセージ処理遅延時間を比較する．前者の値が大きい場合は，タスク n の優先度を高める．また，後者の値が大きい場合は，優先度を低める．
- 前メッセージ処理においてブロッキング時間が発生している場合は，実行開始時刻を遅くするため，優先度を低くする．
- メッセージ処理完了後に，ブロッキング時間が未発生であり，かつ，実測された処理遅延時間が特性 1 を満たさないならば，処理遅延時間を小さくするため，優先度を高める．
- メッセージ処理完了後に，ブロッキング時間が未発生であり，かつ，実測された処理遅延時間が特性 2 を満たさないならば，処理遅延時間を大きくするため，優先度を低める．

以上により，予約 CPU 利用率に基づき，連続メディア出力装置の時間制約を満たし，かつ，エンドーエンドの処理遅延時間を小さくすることが可能となる．

3.4 適応デッドライン時刻

ストリーム処理するタスクは，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られる．従って，デッドライン時刻に許容遅延幅 h_n を持たせる．これにより，タスクの優先度として用いるデッドライン時刻は，式（8）で求められる時刻に，許容遅延幅内 $[-h_n, h_n]$ で，スケジュール可能性を高める制御による優先度の修正を行った時刻とする．この時刻を適応デッドライン時刻 $d_n^{adap}(m_i)$ と呼ぶ．

3.5 複数のストリーム処理が混在する環境と多重制約

3.3 節では，スケジュール可能性を高める制御について述べた．しかし，この制御は単一のストリーム処理における制御である．マルチメディア環境においては，多地点のテレビ会議システムに見られるように，複数のストリーム処理が混在する場合が多い．すなわち，マルチメディア環境におけるストリーム処理は，パインラインモデルにより接続されたタスク群が複数混在するタスクセットとして考えられる．このようなタスクセットにおいて前節における制御を行うスケジュール問題は，複数のパイプラインのタスク群において前節における制御，すなわち制約を同時に満たすような状態を探し出す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 モデルと熱力学的モデルを組み合わせた適応メカニズム [8] を VBR ストリーム処理環境に適用する．以下にその概要を示す．

PDP モデルでは，多くの単純な情報処理ユニットが相互に結合し，それぞれが他のユニットから信号を受け取る．その入力信号が正の値である場合は，ユニットの状態値を高める．負の値である場合は，その状態値を低める．さらに，その状態値に応じた信号を他のユニットに送る．この相互結合ネットワーク（以降，ネットワーク）に何らかの外部条件を与え，ユニット毎に非同期に相互作用を繰り返すと，各ユニットはある状態に落ち着く．その状態がある条件での制約を最大に充足した状態を表す．

以上のことから，複数のパイプラインのタスク群における制約を充足する状態を算出するために，タスク間の依存関係を PDP モデルのネットワークに次のように対応づける．

- 各ユニットの状態値（0 から 1 までの連続値）は，各タスクの重要度とする．
- ユニット間の結合は，通信するタスク間の場合正の重みを，通信しないタスク間の場合負の重みを持たせる．これにより，同一パイプライン上のタスク群の重要度が連動し，タスクの実行機会が同じになる．
- 各ユニットに与えられる外部条件は，変動するタスク実行状況に対応してネットワークの動作を修正する力とする．すなわち，外部条件は，各タスクの変動する処理遅延時間に応じて，3.3 節で述べた制御を，次のようにネットワークに作用させる．
  - タスク n のメッセージ実効到着時刻における前周期のメッセージ処理遅延時間と後続タスク $n+1$ の同じ時刻における前周期のメッセージ処理遅延時間を比較し，前者の値が大きい場合は，正の入力とする．また，後者の値が大きい場合は，負の入力とする．
  - メッセージ処理でブロッキング時間が発生しているならば，負の入力とする．
  - メッセージ処理でブロッキング時間が未発生であり，実測された処理遅延時間が特性 1 を満たさないならば，正の入力とする．
  - メッセージ処理でブロッキング時間が未発生であり，実測された処理遅延時間が特性 2 を満たさないならば，負の入力とする．

以上により，複数のパイプラインのタスク群における制約をネットワークを動作させて処理する．しかし，PDP モデルは，ネットワークを動作させると，その状態は初期値に依存してある制約充足度の高い状態に至り，動作は静止する．一方，ストリーム処理環境において，各タスクの処理遅延時間は，処理するメッセージにより，常に変化する．ネットワークは，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 モデルに熱力学的モデルを加える．熱力学的モデルは，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，高い充足状態から離れ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，高い充足状態の近傍にあ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 モデルを各タスクの変動する処理遅延時間に連続的に動作するようになる．従って，ブロッキング時間を抑制し，特性１と２を満たし，かつ，タスク間の処理遅延時間を均一にする適応デッドライン時刻を探し続けることを可能とする．

4. 性能評価

ADM を，我々が開発している分散 OS Solelc[5] 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

4.2 評価方法と評価タスクセット

図 1 に基づき，パイプラインは次のように構成する．

- 周期的にメッセージを生成する入力装置
- 周期的にメッセージを消費する出力装置
- メッセージ毎に処理時間が変動する入力サーバタスク，出力サーバタスク，アプリケーションタスク

ADM は複数のストリーム処理が混在することを想定していることから，評価タスクセットを，上記のパイプライン処理を行うタスクの組を 3 組（**表 1** の task1〜9）と，ランダムな負荷を生成する周期タスク（表 1 の task10）より構成する．

評価方法は，周期タスクモデルによる方式，LBAP による方式および ADM の 3 つの方式を，上記タスクセットの負荷タスクとパイプライン上のタスクのスケジューリング成功率で比較評価する．また，各タスクの時間属性を表 1 に示す．ただし，タスクの実行時間は，メッセージの処理毎に，最大実行時間と最小実行時間の間をランダムに変動する時間とし，スケジューラには未知とする．

4.3 スケジューリング成功率

評価タスクセットに関して，負荷状況，初期位相，キュー使用量限度を変動させて，スケジューリング成功率を前述の 3 つのスケジューリング方式について比較評価した．スケジューリング成功率（以降成功率）を，次のように定義する．

$$SuccessRaito = \frac{Success(Load)+Success(Msg)+Lost(Msg)}{Finish(Load)+Finish(Msg)+Lost(Msg)}$$

$Success(Load)$ はデッドライン時刻までに処理を完了した負荷周期タスク数，$Success(Msg)$ は出力装置の処理開始時刻までに出力装置に到着したメッセージ数，$Lost(Msg)$ は入力装置がキューの使用量限度のため破棄したメッセージ数，$Finish(Load)$ は処理完了した負荷周期タスク数，$Finish(Msg)$ は入力装置から生成され出力装置に到着したメッセージ数である．

負荷状況に応じた成功率を**図 3** に示す．負荷は，すべての実行タスクにおける周期（$1/R^c$）に占める平均実行時間の割合の総和とした．また，評価時の負荷量は，負荷周期タスクの最大実行時間と最小実行時間により調整した．各タスク間の初期位相は，各タスクの周期の 16 倍の時間，キュー使用量限度は 32 メッセージとした．さらに，LBAP による方式の最大論理処理遅延時間は 4 つのケース（周期の 4 倍，16 倍，32 倍，64 倍）を用い，それぞれにデッドライン時刻を設定した．

図 3 から分かるように，ADM は他の方式より成功率が常に高い．しかし，高負荷（負荷が 1.07）時に，成功率が悪化する．これは，高負荷のためキューが枯渇することによる．そこで，初期位相とキューの使用量制限を大きくした場合（初期位相は周期の 32 倍，キュー使用量限度 64 メッセージ）の負荷状況に応じた成功率を**図 4** に示す．図 3 と図 4 の高負荷時（負荷が 1.07）の場合における ADM と LBAP(x64) の成功率を比較すると，図 3 では，ほぼ同じであるが，図 4 においては ADM の成功率の改善が著しく，ADM が LBAP(x64) を大きく上回る．従って，高負荷時において，ADM のスケジューリングポリシが初期位相とキューを活用し，スケジュール可能性を高めるのに有効であることが分かる．このことを，裏付けるため，初期位相とキュー使用量限度に応じた成功率を示す．

先ず，初期位相に応じた成功率を**図 5** に示す．負荷とキュー使用量限度はそれぞれ 1.03，32 メッセージとし，初期位相は各タスクの周期の倍数で設定した．

図 5 から分かるように，すべての方式で初期位相を大きくすると，成功率が改善されるが，周期タスクモデルによる方式は，初期位相がキュー使用量限度と比較して大きな値になると，成功率が改善されなくなる．これは，高負荷により処理遅延がキュー使用量限度を越えるためである．一方，LBAP による方式と ADM

表 1　評価タスクセットの時間属性
Table 1　Timing attributes for example taskset.

	$1/R^c$	h	C^{max}	C^{min}	C^{ave}
task1	$100ms$	$10ms$	$12ms$	$2ms$	$7ms$
task2	$100ms$	$10ms$	$28ms$	$2ms$	$15ms$
task3	$100ms$	$10ms$	$13ms$	$3ms$	$8ms$
task4	$70ms$	$10ms$	$8ms$	$2ms$	$5ms$
task5	$70ms$	$10ms$	$20ms$	$2ms$	$11ms$
task6	$70ms$	$10ms$	$8ms$	$2ms$	$5ms$
task7	$40ms$	$10ms$	$5ms$	$1ms$	$3ms$
task8	$40ms$	$10ms$	$10ms$	$2ms$	$6ms$
task9	$40ms$	$10ms$	$5ms$	$1ms$	$3ms$
task10	$30ms$	–	–	–	–

h: 遅延許容幅　C^{max}: 最大実行時間
C^{min} : 最小実行時間　C^{ave}: 平均実行時間

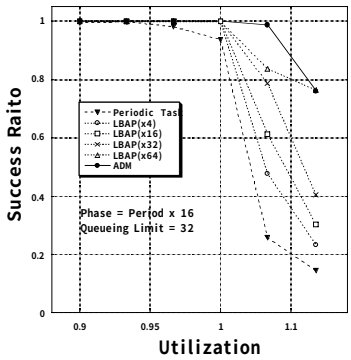

図 3 負荷状況に応じたスケジューリング成功率 (1)
Fig. 3 Scheduling success raito by utilization (1).

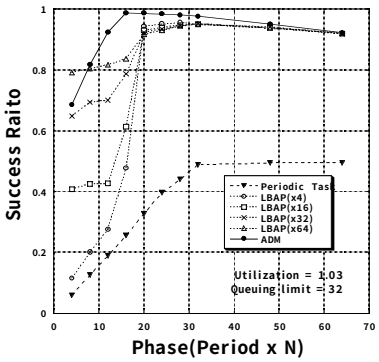

図 5 初期位相状況に応じたスケジューリング成功率
Fig. 5 Scheduling success raito by initial phase.

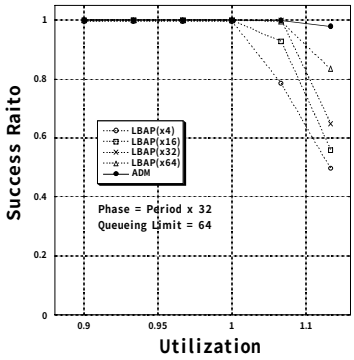

図 4 負荷状況に応じたスケジューリング成功率 (2)
Fig. 4 Scheduling success raito by utilization (2).

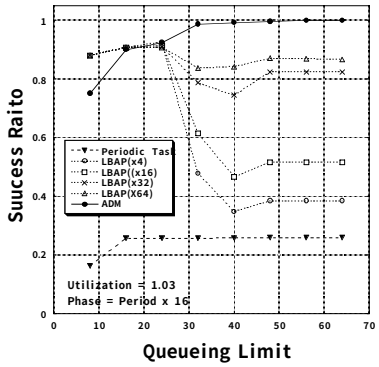

図 6 キュー使用量限度に応じたスケジューリング成功率
Fig. 6 Scheduling success raito by queueing capacity.

は，成功率がほぼ 1 まで改善されるが，両者を比較すると，ADM は少ない初期位相で成功率が改善される．これは，特性 3 を満たす制御が式 7 を満たす条件より有効に機能しているためと考えられる．また，LBAP による方式は，設定したメッセージ最大論理処理遅延時間が小さいほど，初期位相が少ない状態での成功率が低い．これは，メッセージ最大論理処理遅延時間を周期の倍数としているため，倍数値が小さいほど，周期の異なるタスクのメッセージ最大論理処理遅延時間の差が小さくなり，設定されるデッドライン時刻に差がなくなる．従って，メッセージ最大論理処理遅延時間の倍数値が小さい場合は，倍数値が大きい場合と比較して，周期の短いタスクの実行機会が減少し，遅延時間が大きくなる．そのため，成功率が低下すると考えられる．この観点で考えると，ADM が少ない初期位相で成功率を改善することは，ADM の多重制約を解く適応メカニズムが有効に機能した結果と考えることもできる．

次に，キュー使用量限度に応じた成功率を**図 6** に示す．負荷と初期位相はそれぞれ 1.03，初期位相の 16 倍とした．

図 6 から分かるように，キュー使用量限度を大きくすると，ADM だけが成功率を 1 へ上昇させる．これは，ADM が未処理メッセージ数に応じてデッドライン時刻を変動さることにより，負荷をキュー使用量で吸収しているためである．一方，LBAP は，一時的にはスケジュール可能性が改善されるが，キュー使用量限度が増えつづけると，逆にスケジュール可能性が低下する．これは，キュー使用量限度が増えることで，LBAP の固定のメッセージ最大論理処理遅延時間を超える未処理メッセージ数が発生し，式 7 を満たせなくなるためである．従って，LBAP の固定のメッセージ最大論理処理遅延時間により設定するデッドライン時刻より，ADM の未処理メッセージ数により設定するデッドライン時刻の方が，キューを有効に利用することができると考えらる．

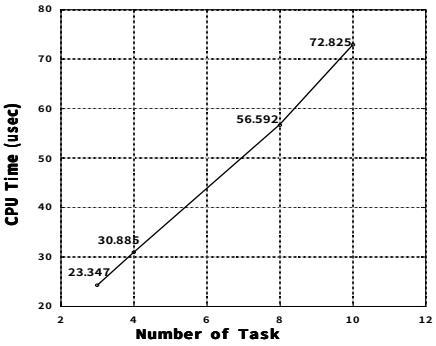

図 7 スケジューリングオーバーヘッド
Fig. 7 Scheduling overhead.

以上のことから，ADM は，従来の方式と比較して，次のような特性をもつ．

- スケジューリング可能性が高く，特に高負荷時に他の方式のまさる．このことから，CPU 利用率を高めることが可能である．
- 少ない初期位相でスケジューリング成功率が改善される．このことから，少ないエンド－エンド処理遅延時間でスケジューリングすることが可能である．
- キューを有効に利用することによって，スケジュール可能性を高めることができる．

4.4 スケジューリングオーバヘッド

スケジューリングオーバヘッドの実測評価結果を**図 7** 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．以上のことから，スケジューリングオーバヘッドはタスク数に比例すると考えられる．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 72μsec 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 36μsec 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時には不要である．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 6) によると，最短周期が 75Hz, すなわち約 13$msec$ 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 36μsec 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，提案ポリシ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，提案ポリシは，連続メディアをストリーム処理するタスクのスケジューリングに十分に適用できると考えられる．

5. おわりに

本稿では，ADM の性能評価結果について述べた．ADM が VBR ストリーム処理タスクのスケジューリング方式において，高い CPU 利用率環境でも，連続メディア出力装置の時間制約を満たし，かつ，エンド－エンドの処理遅延時間を小さくするポリシであることを示した．また，スケジューリングオーバヘッドが微量であることも明らかにした．

参考文献

1) R. Yavatkar and K. Lakshman: Optimization by Simulated Anneling, *Science*, Vol. 220, pp. 671–680 (1983).
2) D. E. Rumelhart, J. L. McCelland and the PDP Research Group: *PARALLEL DISTRIBUTED PROCESSING*, The MIT Press (1986).
3) D. P. Anderson: Metascheduling for continous media, *Trans Comput Syst ACM*, No. 11, pp. 226–252 (1993).
4) R. Govindan and D. P. Anderson: Scheduling and IPC mechanisms for continuous media, *Proceeding of the 13th ACM on Operating Systems Principles*, pp. 68–80 (1991).
5) 芝公仁, 大久保英嗣:分散オペレーティングシステム Solelc の構成, **情報処理学会研究会報告** *2000-OS-84*, pp.237-244 (2000).
6) B. Furth: Multimedia Systems: An Overview, *IEEE Multimedia*, Vol. 1, No. 1, pp. 47–59 (1994).
7) 滝沢泰久, 芝公仁, 大久保英嗣:連続メディア処理における時間制約と通信遅延に適応するタスクスケジューリング, **情報処理学会研究会報告** *2000-OS-86*, Vol. 2001, No. 21, pp. 123–130 (2001).
8) 滝沢泰久, 芝公仁, 大久保英嗣:連続メディア処理における時間制約と通信遅延に適応するタスクスケジューリング, **情報処理学会論文誌：数理モデル化と応用**, Vol. 42, No. Sig5, pp. 29–41 (2001).