Qスレッド: 連続メディアの動的QOS制御に適したプロセス実行モデル†

2F－2

河内谷 清久仁[1‡]　徳田 英幸[2]
[1]日本アイ・ビー・エム(株) 東京基礎研究所　[2]慶應義塾大学 環境情報学部

1 はじめに

我々は，Real-Time Mach 3.0 (以下 RT-Mach と略す)[1] を拡張する形で，連続メディア処理を行なうための基盤ソフトウェアの研究開発を行なってきている [2]．その研究テーマの一つとして，連続メディア処理の記述に適したプロセス実行モデルの実現があげられる [3, 4, 5]．

連続メディアの一つの特徴は，その名のとおり処理が連続的，周期的に行なわれることであり，そのためのプロセス実行モデルとしては，一定周期でエントリポイントが呼び出される「周期的実行モデル」が有効であると考えられる．我々が研究のベースとして用いている RT-Mach でも，周期的実時間スレッドが提供されており，毎回の呼び出し時に一まとまりのメディアデータ (例えばビデオの１フレーム) を処理することで連続メディア処理が行なえる [6]．

しかし，周期的実時間スレッドは元来実時間処理のためのプリミティブとして用意されたものであり，必ずしも連続メディア処理に適していない部分がある．例えば，タスクセットが定義されているような静的な実時間処理とは違い，連続メディア処理では，ユーザの操作などによりシステム環境が動的に変化していくことが普通である．このような環境でも処理の実時間性を保証するためには，資源使用の保証 (Reservation) がある程度できることがのぞましい．また，連続メディア処理では，処理のサービスの質 (QOS: Quality of Service) を調整することで，メディアのもつ時間的制約を守りつつ処理量を変更することが可能であり，そのためには，実際に使用可能な資源に応じて処理量を適応 (Adaptation) させていける仕組みがあることがのぞまれる．

RT-Mach の周期的実時間スレッドのモデル自体には，これらの機構は組み込まれていないため，連続メディア処理における動的 QOS 制御などを実現するためには，モデル外の機構を用いてユーザプログラム側で対処する必要があった [7]．我々は，この点を改良し，周期的実時間スレッドに資源使用の保証と適応のための機構を組み込んだ，動的 QOS 制御に適した新しいプロセス実行モデル「Qスレッド」を開発した．

2 新しいプロセス実行モデル「Qスレッド」

使用可能な資源量に応じて連続メディアの QOS を調整する代表的な方法として，時間的解像度の調整と空間的解像度の調整が考えられる．これは，上述の周期的実行モデルにおいては，それぞれ周期および一回あたりの処理量を調整することに対応している．Qスレッド実行モデルは，この周期と処理量の調整を組み込んだプロセス実行モデルである [8]．

Qスレッドの基本的アイデアは以下のとおりである．まず，システム側で資源使用の保証を行なうために，ユーザプログラム側からは起動の周期に加えて一回あたりの処理量 (必要な CPU 時間) もスレッド属性として申告するようにする．この周期と処理量はある特定の値を指定するのではなく，ユーザプログラムが対応可能な「範囲」で指定される．この範囲内の一点が，使用可能な CPU 資源量に応じて動的に決定される．ユーザプログラムが指定したエントリポイントはこの周期で呼び出され，その際，可能な処理量 (使用可能な CPU 時間) などの情報がユーザプログラム側での処理量調整のヒントとして渡される．ユーザプログラムがこのヒント中の処理量の指定を守っている限り，その動作は保証される．表1は，周期的実時間スレッドとQスレッドを比較したものである．

	周期的実時間スレッド	Qスレッド
機能	ユーザ指定のエントリポイントが定期的に呼び出される	
属性	周期などを指定	周期と処理量の「範囲」，QOS 制御ポリシなどを指定
周期	固定 (変更は可能)	指定された範囲内で，ポリシにしたがって変動
呼出時の引数	ユーザ指定の引数	さらに，可能な処理量などの情報もヒントとして提供
処理の保証	なし (リザーブにより可能)	可能な処理量の制限を守っている限り保証
用途	(ソフト/ハード) リアルタイム処理	(動的 QOS 制御を考慮した) 連続メディア処理

表 1: 周期的実時間スレッドとQスレッドの比較

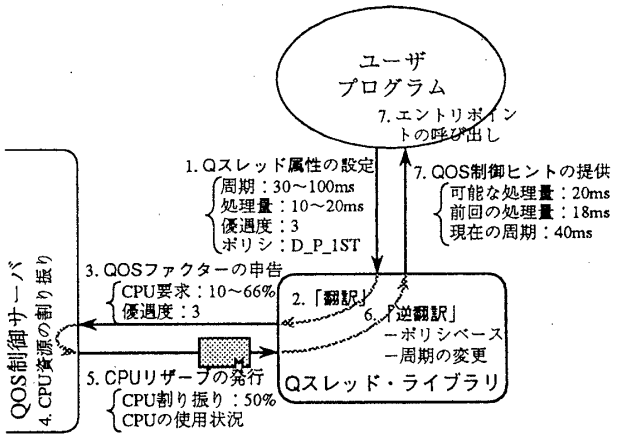

図 1: Qスレッド・ライブラリによる QOS 翻訳

現在Qスレッドは，RT-Mach 上に，周期的実時間スレッドをベースとしたユーザレベル・ライブラリとして実現されている．ユーザプログラムからQスレッドの生成を指示されると，ライブラリはまず，Qスレッドの属性中で指定された周期と処理量の範囲を必要な CPU 資源量の範囲 (QOS ファクター) に「翻訳」し，システム内に存在する「QOS 制御サーバ [8, 9]」に対して申告する．QOS 制御サーバは，この情報から CPU 資源の割り振りと予約を行ない，ライブラリにそれを通知する．ライブラリは，この CPU 割り振り量を可能な周期と処理量に「逆翻訳」し，この周期でエントリポイントを定期的に呼び出す．この結果は QOS 調整のためのヒントとしてユーザプログラムにも渡される．図1は，この「QOS 翻訳」の流れを示したものである．なお，この処理は最初の一回だけの静的なものではなく，システムの状況などに応じて随時動的に行なわれる．

QOS 制御サーバから割り振られた CPU 資源量を可能な周期

Q-Thread: A New Execution Model for Dynamic QOS Control of Continuous-Media Processing
Kiyokuni KAWACHIYA[1] and Hideyuki TOKUDA[2]
[1]IBM Research, Tokyo Research Laboratory
1623-14, Shimotsuruma, Yamato, Kanagawa 242, Japan
E-Mail: <kawatiya@trl.ibm.co.jp>
[2]Faculty of Environmental Information, Keio University

†この研究は，情報処理振興事業協会 (IPA) が実施している開放型基盤ソフトウェア研究開発評価事業「マルチメディア統合環境基盤ソフトウェア」プロジェクトのもとに行なわれた．

‡開放型基盤ソフトウェア湘南藤沢キャンパス研究室の研究員としてIPAに登録されている．

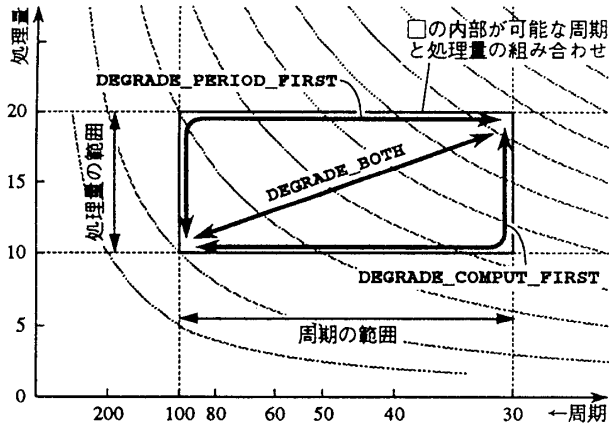

図 2: Q スレッドの属性と QOS 制御ポリシ

関数名	機　能
qoslib_init()	Q スレッド・ライブラリの初期化
qthread_attribute_init()	Q スレッド属性の初期化
qthread_create()	Q スレッドを生成する
qthread_get_comput()	既に消費された CPU 時間を得る
qthread_get_attribute()	Q スレッド属性を取得する
qthread_set_attribute()	Q スレッド属性を変更する
qthread_terminate()	Q スレッドを終了する

表 2: Q スレッド・ライブラリが提供する関数

と処理量に「逆翻訳」する際のポリシとして，現在の実装では次の 4 種類が提供されている．DEGRADE_PERIOD_FIRST は，資源が十分にない場合，まず周期をのばす空間的解像度優先のポリシである．DEGRADE_COMPUT_FIRST は逆に，まず一回あたりの処理量を減らす時間的解像度優先のポリシである．DEGRADE_BOTH では，周期と処理量の両方が均等に変更される．HINT_ONLY の場合，QOS 制御はすべてユーザプログラム側にまかされ，そのためのヒントのみが提供される．これらのポリシは Q スレッドの属性として指定可能である．図 2中の矢印は，各 QOS 制御ポリシにおける周期と処理量の変化を示したものである．

Q スレッドの周期は，(HINT_ONLY 以外のポリシの場合) ライブラリ側で自動的に変更される．しかし，一回あたりの処理量に対応した仕事量の調整はユーザプログラム側で行なう必要があり，ユーザ指定のエントリポイントの呼び出し時に，そのための「QOS 制御ヒント」が渡される．ユーザプログラムがこのヒント中の処理量の指定を守っている限り，その動作は RT-Mach の CPU リザーベーション機能 [10] により保証される．QOS 制御ヒントには，前回の呼び出し時に実際に使用された CPU 時間の情報なども含まれており，ユーザプログラムが仕事量を調整するためのヒントとして利用することができる．

Q スレッド・ライブラリ (と QOS 制御サーバ) は，既に RT-Mach 上に実装されている．表 2に，Q スレッド・ライブラリによって提供される関数の一覧を示す．

3　動的 QOS 制御の実験

我々は，Q スレッドの有効性と問題点を検証するために，動的 QOS 制御の実験を行なっている．「ダミープログラム」を用いた Q スレッドの基本機能の検証については，既に報告済み [8] であるので，今回はより実アプリケーションに近い実験を試みた．実験には，Q スレッドで書かれた QuickTime ビデオプレーヤを用いた．このプログラムでは，Q スレッド生成時に周期の範囲を指定することで，フレームレートが自動的に変化するようになっている．図 3は，周期の範囲指定と優先度が異なる 3 つの QuickTime プレーヤを用いた場合の実験結果である．上のグラフは各プレーヤの実際のフレームレートの変化を示しており，下のグラフは実験期間中の CPU の使用率とデッドラインミスの状況を示している．

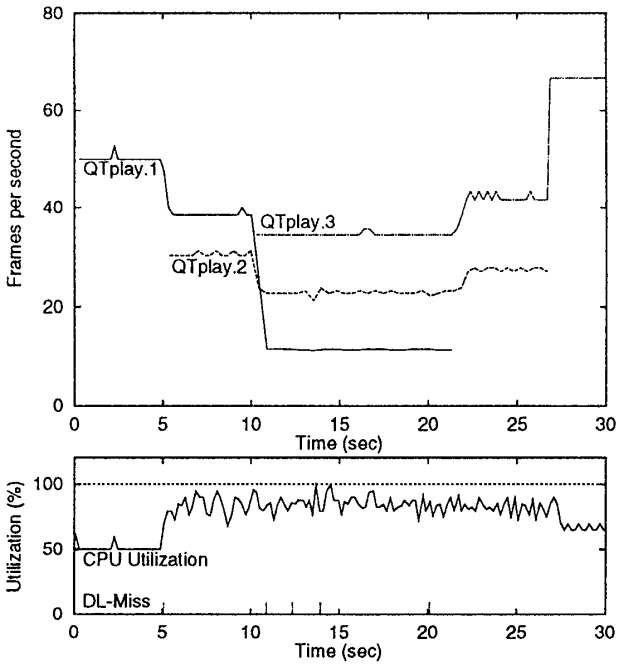

図 3: Q スレッドによる動的 QOS 制御の実験結果

4　おわりに

連続メディア処理では，時間的制約を守るために，使用可能な資源量に応じて，動的に QOS を変更していく必要がある．本稿では，このような動的 QOS 制御に適した新しいプロセス実行モデルである「Q スレッド」について述べた．これは従来の周期的実時間スレッドに，資源使用の保証と適応のための機構を組み込んだものである．Q スレッドでは，その属性として処理の周期 (時間的解像度) や一回あたりの処理量 (空間的解像度) の「範囲」を指定することができ，ユーザは，Q スレッドを用いることで，QOS を動的に調整していくようなプログラムを容易に作成することができる．本稿では同時に，実アプリケーションへの適用例として，Q スレッドで書かれた QuickTime プレーヤを用いた動的 QOS 制御の実験結果についても示した．

参考文献

[1] H. Tokuda et al.: "Real-Time Mach: Towards a Predictable Real-Time System," *Proc. USENIX Mach Workshop*, pp. 73–82 (1990).

[2] Keio-MMP プロジェクト: WWW ホームページ, URL: <http://www.mmp.sfc.keio.ac.jp/>.

[3] K. Jeffay and D. Bennett: "A Rate-Based Execution Abstraction for Multimedia Computing," *Proc. NOSSDAV '95*, pp. 67–78 (1995).

[4] J. Nieh and M. S. Lam: "Integrated Processor Scheduling for Multimedia," *Proc. NOSSDAV '95*, pp. 215–218 (1995).

[5] R. Yavatkar and K. Lakshman: "A CPU Scheduling Algorithm for Continuous Media Applications," *Proc. NOSSDAV '95*, pp. 223–226 (1995).

[6] H. Tokuda et al.: "A Real-Time Thread Model for Continuous Media Applications," *ART Group Tech. Report*, CMU (1993).

[7] K. Kawachiya et al.: "Evaluation of QOS-Control Servers on Real-Time Mach," *Proc. NOSSDAV '95*, pp. 123–126 (1995).

[8] 河内谷, 徳田: "「QOS チケット」モデルに基づく連続メディア処理の動的 QOS 制御," 第 7 回コンピュータシステム・シンポジウム論文集, pp. 141–148 (1995).

[9] K. Kawachiya and H. Tokuda: "QOS-Ticket: A New Resource-Management Mechanism for Dynamic QOS Control of Multimedia," *Proc. MM Japan '96*, to appear (1996).

[10] C. W. Mercer et al.: "Processor Capacity Reserves: Operating System Support for Multimedia Applications," *Proc. ICMCS '94*, pp. 90–99 (1994).