7R－7

Keio-MMPにおける Conductor/Performerアーキテクチャの協調性能評価†

西尾 信彦*　多田 征司**‡　徳田 英幸*　萩野 達也*　斎藤 信男*

*慶應義塾大学環境情報学部　**横河・ヒューレット・パッカード(株)

1 はじめに

慶應義塾大学では，情報処理振興事業協会(IPA)からの委託研究として10社の企業と共同研究として，分散マルチメディア統合環境プロジェクト(Keio-MMPプロジェクト)を行なっている．本プロジェクトでは動画や音声などの時間にしたがって連続的に変化する連続メディア(Continuous Media)を扱うために，実時間性を考慮した基盤ソフトウェアを開発/評価する研究を行なっている[2]．本稿では実時間カーネル上で連続メディアアプリケーションとの間で動作する連続メディア処理のための各種サーバ群をいかに実時間性を守り協調/同期して動作させるかの研究について述べる．

2 Conductor/Performerアーキテクチャ

われわれは全体モデルの構築のために以下の点を考慮した;(1)連続メディアのInter-/Intra-ストリームが同期する実時間性があること，(2)各種のアプリケーションの要求に応じてさまざまな形態のセッションを自由に組み立てられること，(3)セッション確立前(admission制御)と後のQOS制御(動的なサービスの質(資源)の制御)ができること，(4)効率がよく，資源のusabilityが高いこと．

そこでわれわれはストリームコンダクタ(以下conductor)と呼ばれる同期サーバを中心として，他の各種サーバ(パフォーマ(以下performer)と呼ぶ)からマイクロカーネル上にmiddle wareを構成するConductor/Performerアーキテクチャを採用した[3]．実際に各種メディア機器を制御しデータを転送するのがperformerで，それらに処理のタイミングを知らせる同期信号を送るのがconductorである．Conductorは(1)アプケーション，(2)performer，(3)他ホストのconductorとの間にI/Fをもつ．以下にconductorの役目をまとめる．

- 各ホストに1つ存在し，そのホストの資源を管理する．
- アプリケーションとのI/Fを集中管理する．
- アプリケーションから要求されたセッションに必要な資源の見積もりを立て，そのadmission制御をする．
- パフォーマを連結して生成したストリームの実時間性の保証をする．
- セッション内のストリーム間[1]の同期に責任をもつ．
- Conductorはアプリケーションから要求されるQOSを保証するために，ストリームの動的な性能評価をしてperformerとのI/Fにより制御する．
- アプリケーションライターにはperformerをメディアオブジェクトという「部品」で表象してそれを連結するprogramming abstractionを提供する．
- 分散ストリームのためにconductor間通信のプロトコルをもつ．

Conductorは自分への要求により，その他のperformerと通信をしてそれらを連結させたストリームを確立し，それに応じた周期スレッドをデッドラインハンドラつきで生成する．これで連続メディアデータの実時間性を保証しつつそのQOS制御を行なう．

Performerはconductorとの I/Fをもち，それぞれの内部の実装は自由である．Performerの例としては，ファイルサーバやビデオ/オーディオキャプチャやXウィンドウサーバやネットワークプロトコルサーバなどがある．

3 メディアオブジェクトと同期/非同期ストリーム

Conductor内にはストリームを構成する各performerとのセッションを代表するメディアオブジェクトが生成される．これに対応してperformer内にも一つのオブジェクトないしはセッション，スレッドなどの実体が生成される．各種performerによりどのような実体が生成されても構わない．これら二つの間でRPCを行ない全体として一つのオブジェクトのようにアプリケーションに抽象化して見せる．

Conductor内のメディアオブジェクトがgenericに持つべき代表的なメソッドとして，(1)QOSを指定して必要な資源を確保する(2)QOSを変更する(3)ストリームとして連結した次のperformerに指定されたタイムスタンプつきのデータフレームをマップする，などがある．

これらのメソッドをconductor内でストリーム毎に生成された周期スレッドが周期毎にRPCによってメソッド起動をしていく．この周期スレッドが自分の周期内で処理を完了できない場合はデッドラインハンドラが自動的に起動され，適切なQOS変更のメソッドを起動する．これはストリーム内のすべてのデータの移動を統一的に一つのメソッド起動により行なう単純化ができ見とおしがよい．再生するすべてのフレームに対して毎回conductorがそのタイムスタンプを指定する方式であり，ストリームとconductorが同期して動作することから同期ストリームと呼ぶ．

"Cooperativity Performance Evaluation on Conductor/Performer Architecture in Keio-MMP Project"†

NISHIO, Nobuhiko*, Seiji Tada**, Hideyuki Tokuda*, Tatsuya Hagino*, Nobuo Saito*

*Keio University, 5322, Endo, Fujisawa-shi, Kanagawa, 252 Japan, **Yokogawa-Hewlett-Packard Ltd., 1-3-2, Murotani, Nishi-ku, Kobe,Hyogo 651-22, Japan

† 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．

[1]例えば1セッション内には動画ストリームと音声ストリームをもつものがある．

一方各ステップでの遅延が絶対的に大きいか，挙動が予測不可能であるとスムーズな再生には先読みが有効となるが，その先行制御は conductor 内の周期スレッドとテンポとは独立で，単純な位相のずれでは対応できない．また同期ストリームでは処理の並列性を出せない．そこで conductor 内のスレッドの処理のテンポに間に合う (予測可能 and/or 遅延の小さい) 部分と，(予測不可能 and/or 遅延の大きい) 先行制御が必要な部分とに分けてそれぞれに異なった処理を施すことにした．前者は同期ストリームを後者は先行制御を伴ない，最初の QOS 指定以降は conductor とは非同期に自律的な一定レートのストリームを確立して実現する．後者を**非同期ストリーム**と呼ぶ．

各種 performer は期待された QOS を満たすのに必要な量のバッファを conductor に用意してもらい，それをアクセスするすべてのサーバのアドレス空間に共有させる．その共有バッファの中から各 performer が自分の使用可能領域を割当ててロックし，自分の処理を行なう．自分の処理が完了すると，どの処理が終ったかをロックにマーカとして書き込み解放する．今度は自分の下流側の performer がそれを用いて処理を始める．このようにして，常に一定の量しかメモリ資源を使わないように限定でき，かつメモリのマップの手間も節約できるという利点がある．この共有バッファを **Cyclic Shared Buffer(CSB)** と呼ぶ．各 performer は処理状況に応じてバッファにマーカをつけ，conductor がそれを参照してストリームの全体性能を評価できるようにしている．

4 動的 QOS 制御

非同期ストリームではいずれかの performer が処理すべきデータが特定量より滞留し始めることをトリガとする．その performer は conductor に QOS 変更を要求し，conductor が最上流オブジェクトに，そこから順に下流まで伝播させ全体での調整をとる．また，同期ストリームについても conductor 内の周期スレッドが実行時性能評価を行ない，それをトリガとした QOS 制御も同時に進行させる．周期スレッドの実行時性能評価としては，(1) デッドラインハンドラをトリガとする，(2) 自らの周期内での処理時間を実測して判定する，(3)CSB の各単位フレームに各 performer がつけたマーカで全体状況を把握して行なう，といったものがある [4].

5 QuickTime Player による実験

Conductor/Performer アーキテクチャに基いて QuickTime Player を実装した．これは，Apple 社の QuickTime ファイルを UNIX ファイルシステム上に格納し，X Window やオーディオ performer で再生するものである．

われわれは，RT-Mach 3.0[1] 上で QuickTime Player を conductor (qtplay) に見立てて，MIG を用いて UX サーバ上に作成した簡易なファイルサーバ (qt_server)，MediaVision 社の ProAudio16 サウンドカードを駆動できるオーディオ performer(pas_serv)，拡張された X Window performer(machX) といった各種 performer と上記の方式で協調動作するプロトタイプを実装した．アプリケーションと conductor とは未分化な実装である．

Qtplay は一つの QuickTime ファイルの再生に関して，音声と動画の二つのストリームを二つの周期の異なった (音声 1 秒，動画 1/fps 秒) 周期スレッドではる．音声スレッドの方を優先度を上げて，FixedPriority ポリシーでスケジューリングした．ここではファイルサービス performer と qtplay の間が非同期ストリームで CSB があり，qtplay から X Window performer が同期ストリームである．

実験では，160x120，8bit256 色カラー，10fps，7 秒間の QuickTime ファイルをローカルな HD から再生させた．このとき前述の CSB の各バッファにつけられたマークの統計を取った．われわれは先行処理用に 1 セッションで 4 フレーム読み込める CSB を用意した．この内，conductor 内の周期スレッドがリズームしたときには常時 2 つが先読みされていた．

動的 QOS 制御時の Conductor/Performer に及ぼす協調性を調べるためにファイルの毎回の再生の途中で conductor が再生 fps を 10fps と 5fps とで行き来させ，その要求に先行制御をしているファイルサービス performer が追随できるかを計測した．図は再生開始時から換算してそのときに再生されるべきフレームと実際に再生されたもののタイムスタンプのギャップを示したものである．ギャップは 1 フレーム以内に収まれば制御できているものとした．結果はほぼ 1,2 フレーム程度のズレが生じるのみで QOS 変化要求を伝播することができている．

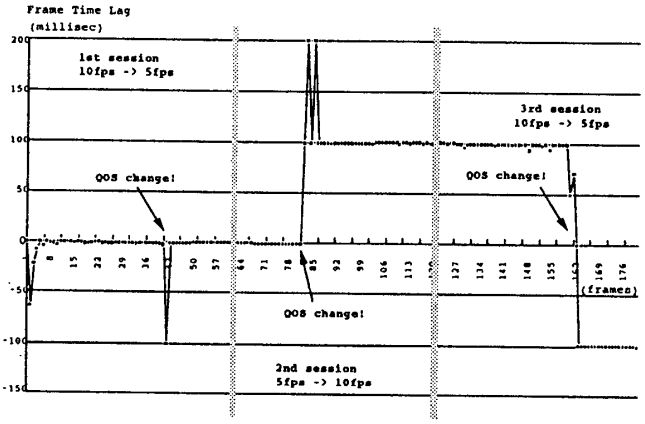

図 1: 動的 QOS 制御での協調性能

6 おわりに

本稿では，連続メディア処理のサーバ構成を実時間性を守って協調/同期させることを目的とする Conductor/Performer モデルを設計し，そのプロトタイプとして実装した QuickTime Player で，各種動的 QOS 制御機構をとりいれ，それらの協調動作を確認した．

参考文献

[1] H. Tokuda, T. Nakajima and P. Rao: "Real-Time Mach: Towards a Predictable Real-Time System," *USENIX Mach Workshop*, pp.73–82 (1990).

[2] 西尾, 徳田他: "マイクロカーネルによる連続メディア処理の基盤技術", 第 5 回コンピュータシステム・シンポジウム論文集, pp17–24, October, 1993.

[3] 西尾信彦: "Real-Time Mach 3.0 上の連続メディア処理のための協調サーバ群の設計と実験," 情報処理学会第 63 回 OS 研究会 (1994).

[4] H. Tokuda and T. Kitayama. "Dynamic QOS Control based on Real-Time Threads". *Proc. of the 4th Int. Workshop on Network and Operating Systems Support for Digital Audio and Video*, pp.113–122. 1993.