6Z－2

# メディアサーバ制御のための並行スクリプト言語と軽量コルーチンを用いた効率良い実行方式

前田 誠司　　金井 達徳　　岐津 俊樹　　矢尾 浩　　鳥井 修

(株) 東芝 研究開発センター

## 1　はじめに

メディアサーバの制御は、複数のストリームの処理を同時に行なうため、複雑な制御プログラムが必要である。この時、制御プログラムを個々のストリームに着目して記述することで、開発を容易にすることができる。しかし、OSの提供するマルチスレッドなどの並行処理機構を用いると、共有資源の排他制御などの記述が繁雑であったり、コンテクストスイッチによる実行オーバヘッドが大きいなどの問題がある。そこで、C言語を拡張してイベントによる同期や並行処理を記述できる並行スクリプト言語を開発した。さらに、軽量コルーチンを用いて小さなオーバーヘッドで実行する方式を考案して実装した。

## 2　メディアサーバ

メディアサーバは、MPEG-2等の圧縮技術によって符号化された映像のような連続データの蓄積・配信を行なうサーバである。そのメディアサーバの一つであるスマートストリーマ[1][2]は、大量の映像データを蓄積する大容量のハードディスクと、データの転送速度差を吸収するバッファメモリと、データをネットワーク上に配信するネットワーク・インターフェースにより構成される。また、連続データを一定レートで確実に配信するために、タイムスロットと呼ばれる一定の短い時間間隔(数10ms ～ 100ms程度)で制御する。

このようなメディアサーバを制御するプログラムは、

- 各資源(ハードディスク・バッファメモリ・ネットワーク)の予約
- ハードディスクからのデータ読み出し
- ネットワークへのデータ配信
- クライアントからの指示(再生・停止等)への応答

といった処理を、メディアサーバ全体では数百～数千ストリームに関して同時に実行する。このとき、ハードディスクのデータ転送能力・バッファメモリ量・ネットワークバンド幅などの資源を確実にかつ有効に各ストリームに配分する必要がある。また、各ストリームが連続データである映像を滞ることなく配信するためには、タイムスロットに同期して時間的に正確に動作する必要がある。

## 3　制御プログラムの実装方式

メディアサーバの制御プログラムを、入出力の完了やクライアントから非同期に通知される要求などのイベントに従って動作する一つの大きなステートマシンとして記述した場合には、処理性能の面での問題はない。しかし、非常に複雑なプログラムになるので保守や拡張が困難であるという問題が存在する。これに対し、プログラム全体を、各ストリームの配信処理等の動作に着目してその手順を記述したプログラムと、共有資源の管理を行うプログラムの2階層に構造化することによって、拡張性や保守性さらには開発効率の向上を達成することができる。

OSの提供するスレッド等の並行処理機構を用いて、各ストリームの処理をそれぞれ別のスレッドで実行することで、このような実装を行うことができる。しかしこの方式では、共有資源の排他制御を行なう必要があり記述が繁雑になると共に、それによる実行時のオーバーヘッドが大きくなってしまう。さらに、プロセスに比べて軽量であるスレッドを用いたとしても、各ストリームの動作を個別のスレッドで実行するため、メディアサーバ上で数百～数千ストリーム分のスレッドが同時に動作している場合でも、各タイムスロットにおいて全てのスレッドの実行制御を切り替える必要がある。実際にはこのコンテクストスイッチの負荷が非常に高く、制御プログラムを実行するのに能力の高い計算機が必要となる。

そこで筆者らは、このようなメディアサーバ制御プログラムの実装上の問題点を解決するために、イベント駆

Concurrent Script Language and Efficient Implementation with Light-weight Co-routine for Media Servers
Seiji MAEDA, Tatsunori KANAI, Toshiki KIZU, Hiroshi YAO, Osamu TORII
TOSHIBA R&D Center
1, Komukai Toshiba-cho, Saiwai-ku, Kawasaki 210, Japan

動および並行処理の双方を記述できる能力を持つスクリプト言語 CSL(Concurrent Script Language) を開発した。また、その実行系を軽量コルーチンを用いて実装し、コンテクストスイッチや排他制御のオーバーヘッドを無くして効率良く実行できるようにした。

## 4　並行スクリプト言語 CSL

並行スクリプト言語 CSL は、C 言語を拡張し、並行処理とイベント待ちを容易に記述できるように設計した言語である。CSL の文法は基本となる代入文・実行制御文 (for 文や while 文)・関数呼び出しなどは C 言語と同じ文法を採用している。C 言語からの拡張点をまとめると以下のようになる。

- 並行動作を記述する parallel 文
- 複数のイベント発生の待ち合わせを記述する guard 文
- スクリプトが受け取るイベントの集合を定義するスクリプトクラス宣言

parallel 文によって、ハードディスクからバッファメモリへのデータ読み込みと、バッファメモリからネットワークへのデータ送信のような、並行に動作する複数の関連する処理の流れを記述することが出来る。また、guard 文によって、ディスク読み出しタイミングや入出力動作の完了通知、クライアントからの指示などの外部からのイベントが通知されるまで、スクリプトの実行を停止するような処理を容易に記述することができる。

## 5　軽量コルーチンを用いた実行方式

CSL で記述したスクリプトのプログラムは、スクリプトトランスレータで C 言語に変換して実行する。CSL は C 言語をベースにしているため、スクリプトトランスレータで変換するのは、parallel 文で記述された並行動作と、guard 文で記述されたイベント待ちを実現するコードである。この部分のコード生成にあたっては、メディアサーバの制御プログラムの特性を活かしたオーバーヘッドの小さな実行制御方式を考案した。

メディアサーバの制御プログラムにおいては、各ストリームが毎タイムスロットに実行すべき処理にはストリーム間で実行順序の依存関係はなく、タイムスロット内で全てのストリームに必要な処理を実行し終われば良い。トランスレータはこの性質を利用し、各ストリームの処理を実行するスクリプトを、コルーチンとして切り替えて実行するようなコードを生成する。各スクリプトはコルーチンとして実行されるため、同時に複数のプログラムが共有資源をアクセスすることが避けられるので、排他制御のロック変数の必要性を無くすこともできる。

スクリプトトランスレータは、ストリームの配信処理等を行うスクリプトを、ひとつの C 言語の関数に変換する。この関数は、各タイムスロットに置いて実行すべき処理を完了し、何らかのイベント待ち状態になると戻ってくるようになっている。毎タイムスロット、各ストリームに対応するこの関数を順に呼び出すことで、そのタイムスロットにおいて各ストリームに必要な処理が順次コルーチンとして実行される。コルーチン間の切替えをサブルーチンコールで行うことで、コンテクストスイッチが不要な軽量な実装が可能になっている。

## 6　評価

本稿で述べたメディアサーバ制御プログラムの実装方式を評価するため、CSL で記述したスクリプトプログラムを軽量コルーチンで実行する場合と、Solaris2.5.1 の Solaris スレッドで実行する場合の比較を行った。Soralis スレッド版では parallel 文を複数のスレッドとして実行し、event 文によるイベント待ちはセマフォを用いて実現した。その結果、複数のスレッド間で共有資源アクセスのための排他制御が必要になり、また、スレッドの切り替えの為のコンテクストスイッチが多発することになった。それぞれの方式について、各タイムスロットで必要な処理の実行時間を測定した結果、同時に 240 ストリームの配信を制御している時で、軽量コルーチン版は Solaris スレッドに比べて 1 桁以上少ない時間で処理することができた。

## 7　おわりに

メディアサーバの制御に適した、並行スクリプト言語 CSL を開発し、個々のストリームに着目して処理を記述することで、制御プログラムの開発を容易にした。さらに、軽量コルーチンを用いて実行時のオーバーヘッドを小さくする、効率の良い実行方式を実装した。実際に制御プログラムを実装し、プログラム記述が容易であること、および実行時のオーバーヘッドが少ないことを確認した。

## 参考文献

[1] 浅野ら, “マルチメディアサーバスマートストリーマ (1) アーキテクチャ概要”, 情報処理学会 第 54 回全国大会, March 1997.

[2] 岐津ら, “マルチメディアサーバスマートストリーマ (5) 制御ソフトウェア”, 情報処理学会 第 54 回全国大会, March 1997.