プラントシステム向けリアルタイム・データサーバにおける実時間データハンドリング

4Q－6

三菱電機　産業システム研究所
大西秀次　島川博光　竹垣盛一

1．はじめに

著者らは、プラントシステム向けリアルタイム・データサーバを開発している。リアルタイム・データサーバは、プラントからのデータのリアルタイムな獲得とともに監視制御を行なうワークステーションの要求に応じて、データの提供を行なう。

実時間データハンドリングにおいて、サンプルされるデータの種類は、プラントごとに異なる。また、データの獲得と提供は並行して実行されるので、データに対する排他制御が必要である。従来、このような問題に対しては、レコード長や最大レコード数を固定にしたライブラリが用意され、プログラマは1レコードの大きさを意識してプログラミングを行なってきた。しかし、リアルタイム・データサーバの機能はかなり限定されているので、機能実現の手続きは、1レコードのデータ構造による違いを除いてはほぼ同じである。著者らはこの点に着目し、プログラマが1レコードのデータ構造をスキーマとして定義することにより、レコード長や排他制御を考慮した手続きを表現した関数を生成する方法を採用した。この関数はデータの排他制御をプログラマより隠蔽する。プログラマは、この関数を指定すれば、レコード長を意識せずにプログラムを作成でき、さらに、排他制御を意識する必要がなくなる。これにより、リアルタイム・アプリケーションプログラムの生産性が向上する。

2．リアルタイム・データアクセス管理

図1.にリアルタイム・データサーバ[1]の構成を示す。リアルタイム・データアクセスに必要な手続きには、リングバッファへのデータの獲得、ディスク内のファイルへのデータの退避、リングバッファやファイルからのデータの検索がある。

プラントごとにサンプル・データを表す1レコードの構造が異なるので、リングバッファとファイルの作成には、レコードの大きさと数を意識する必要がある。最近のデータの検索は、過去のデータの検索よりも実時間性を必要とするので、最近のデータはリングバッファに、過去のデータはファイルに蓄える。ファイルには連続ファイルを用いる。連続ファイルへは、あるアクセス単位の整数倍ごとにしか読み書きできないので、任意の大きさのレコードを扱うためにはキャッシュが必要である。リングバッファとファイルに対して更新と参照の処理が存在するので、これらの間で、排他制御を行なう必要がある。著者らはこれらをリアルタイム・データアクセス管理と呼んでいる。

3．ライブラリ方式とコード生成方式との比較

実時間データハンドリングを行なうシステムにおける、多種多様なサンプル・データに対して柔軟に扱う方法として、一般に2つに分けられる。

ひとつは、あらゆるサンプル・データを扱うに十分な大きさのメモリ領域と、その管理の手続きをライブラリとする方法である。この方法をライブラリ方式とする。この方法ではメモリ領域に対する柔軟

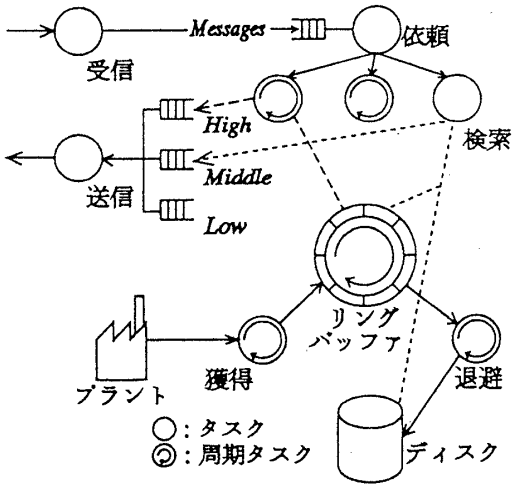

図1. リアルタイム・データサーバ

Real-time Data Handling in Data Server for Plant Monitoring
Hideji Ohnishi　Hiromitsu Shimakawa
Morikazu Takegaki
Mitsubishi Electric Corporation
8-1-1 Tukaguchihonmachi,Amagasaki,Hyogo,
661,Japan

性はプログラマのプログラミングにおける工夫により与えられる。その長所は、ライブラリが正しい限り、あらゆるサンプル・データに対して唯一のライブラリを用いてプログラムを作成できる点である。逆に短所は、プログラマが、メモリ領域へアクセスする際に、各レコードの大きさを意識する必要がある点など動的なメモリ管理が必要となる点である。このためメモリ領域を扱う手続きの作成が複雑となる。

もうひとつは、メモリ領域に対する柔軟性を静的に与える方法である。サンプル・データを表現するデータ構造をユーザにスキーマとして定義してもらい、必要なメモリ領域と、この管理の手続きをソースプログラムとして生成する。ソースプログラムはシステム生成時にコンパイルされ、他の部分とリンクされる。この方法をコード生成方式と呼ぶことにする。この方法の長所は、各サンプル・データに基づいたデータ構造と手続きが生成されるので、手続きが簡単となり、バグの発生確率を低くすることができる点である。短所は、サンプル・データの構造が変化するたびに、プログラムを再コンパイル・リンクする必要がある点である。著者らはコード生成方式を採用する。

4．スキーマからの変換

ユーザがプラントごとにスキーマを定義すると、スキーマ・トランスレータは、そのスキーマに基づいたデータサーバのリアルタイム・データアクセス管理に必要な手続きを、C言語で記述された関数として生成する。

例として、連続ファイルを用いたアクセス手続きの生成を図2.に示す.連続ファイルを用いることで、ディスクへ直接読み書きすることができ、ディスクアクセスにおける実時間性が保証されるが.連続ファイルへは512バイト単位のアクセスしか許されていない。そこで、スキーマ・トランスレータは、スキーマに応じた連続ファイルへのアクセス手続きを生成する。スキーマでは、変数の型と名前が指定される。まず、スキーマから1レコードあたりのバイト数aが計算される。1回で退避されるレコード数をkとして、1回のアクセスにおける必要バイト数(k×a)以上で、512バイトの整数倍であるbを大きさとする構造体をブロックと考える。データサーバは、このブロックを単位として連続ファイルにアクセスする。リングバッファ上のデータを連続ファイルに書き込む場合、1ブロックの大きさをもつキャッシュにレコードの内容をコピーしていき、kレコードコピーされればキャッシュの内容を連続ファイルへ書き込む.連続ファイルからあるレコードを読み出す場合、そのレコードが含まれているブロックを検索し,キャッシュに読み出す。

リアルタイム・データアクセス管理の手続きは、ユーザが定義したスキーマに基づき、自動的に生成される。目的のレコードを得るための手続きを作成する際のレコードの構造の配慮は、ユーザから隠蔽される。これにより、プログラム作成の負荷を軽減できる。

5．おわりに

本稿では、データサーバがプラントのデータへのアクセスをリアルタイムに実行するために必要であるリアルタイム・データアクセス管理と、プログラマが1レコードのデータ構造をスキーマとして定義してもらうことにより、データ構造ごとにリアルタイム・データアクセス管理の手続きを生成するスキーマ変換について述べた。

【参考文献】

[1]大西,水沼,島川,竹垣 プラントシステム向けリアルタイム・データサーバの構築 第37回システム制御情報学会研究発表講演会予稿論文集 P.117-P.118 1993

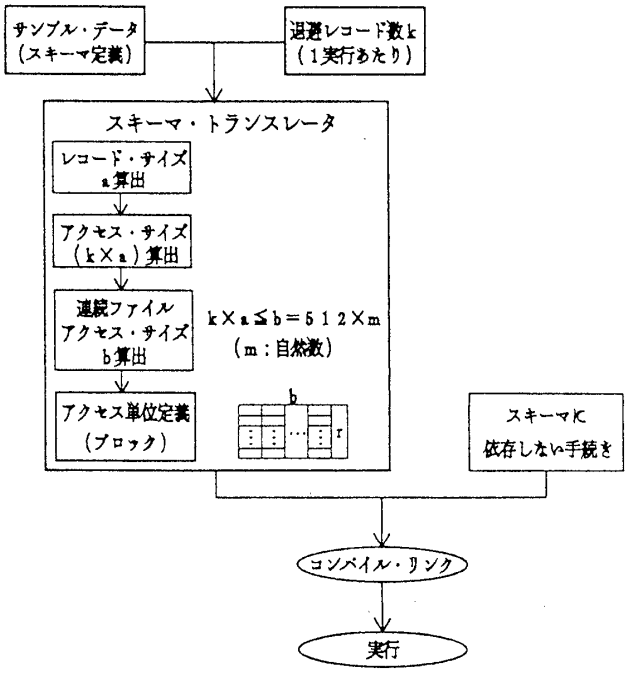

図2　スキーマからの変換による生成