プラント運転支援用リアルタイムトレンドデータベースの構築

6M−8

Development of Real-time Trend Database System for Support of Plant Operation

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

Mitubishi electric corporation
Hideji Ohnishi　Hiromitsu Shimakawa　Morikazu Takegaki

ABSTRACT: We propose a method of real-time trend database systems for large scale plants. We define two agents. One agent brings data to a ring-buffer. The other transmits data from the ring-buffer to discs in order to preserve data. Recent short data are found in the ringbuffer, while long term data are retrieved in the discs.

1．はじめに

著者らは、リアルタイム運転支援用データベースシステムを開発している。プラントにおける観測対象の時間経過の中での状態遷移を時系列と呼ぶことにしている。プラント運転支援システムは、適切な運転操作のガイダンスをオペレータへリアルタイムに提供しなければならない。また、プラントの故障等における原因追求や対策立案等の作業も支援しなければならない。これらの作業に際し、プラントの状態遷移が容易に把握できることが必要である。故に、システムには以下に述べる機能が不可欠となる。それは、

(1)運転支援のモジュールへの時系列の即時な提供。
(2)モジュールの要求に応じた形での時系列の提供。

以上2点である。

そこで本システムでは、プラントよりサンプルされたデータを、一時的に格納しておくリングバッファと、リングバッファ内のデータをディスクへ転送するエージェントを定義する。リングバッファ内のデータにより、上述(1)の機能を、ディスク内のデータにより、上述(2)の機能を実現する。

本稿では、上述の機能を有効に活用した、プラント運転支援用リアルタイムトレンドデータベースの構築について紹介する。

2．運転支援システムにおけるリアルタイム性

プラント運転支援システムの向上のためには、プラントの状態通知や運転操作のガイダンスにおけるリアルタイム性と、プラント故障解析を行なうためのプラントの時系列の記録の保持が、共に必要となる。運転支援がリアルタイムに行なわれるためには、最新の短い期間のサンプルデータを、細かい間隔で取り込まなければならない。また、状態通知や運転支援は、最新の要求時点におけるデータをリアルタイムに取り込まなければならない。故に、データベースは常時要求されるデータに対して、アクセス時間を短くすることが必要である。対して、長期間細かい間隔でサンプルされるデータは、故障解析など、非常時に必要なデータである。故に、アクセス時間は考慮しなくもよいが、大容量のデータベースを必要とする。

ここで、プラントよりサンプルされたデータを、運転支援などのリアルタイムエージェントに必要なデータを蓄えるデータベースと、長期間のデータを蓄えるデータベースとに分ける。まず、サンプルされるデータを時系列としてリングバッファに格納する。リングバッファは、最新の短期間の時系列を蓄える。時系列は、蓄えられている間にディスクへバックアップされる。運転支援に必要な時系列は、リングバッファにアクセスして取り込む。この時系列を用いて行なえば、オペレータへの運転支援に対してリアルタイム性を持つ動作をすることができる。プラントの長期間の時系列を必要とする場合は、ディスクにアクセスすれば確保できる。

3．Multi Thread の Data Base サーバ

データベースに処理を行なうスレッドとして、

(1)プラントよりサンプルされるデータを書き込むスレッド
(2)ディスクにバックアップを行なうスレッド
(3)エージェントからのメッセージを受け取るスレッド
(4)リングバッファへのリードを実行するスレッド

が挙げられる。(図1．参照)(1)のスレッドは、周期100 msで1レコードの書き込みを実行する。(1)のスレッドのプライオリティは最大である。(2)のスレッドは、周期1secで、リングバッファから10レコードを読み出し、読み出したレコードをディスクへ格納する。(2)のスレッドは(1)のスレッドと相互排除をすることなく実行されることが必要である。(3)のスレッドはエージェントからのリングバッファへの要求のメッセージを受け付ける。受け付けられたメッセージは、プライオリティごとに設けられた待ち行列に格納される。待ち行列の中のメッセージは、到着順に並べられる。(4)のスレッドは優先度の高い待ち行列の先頭から、順に実行を行なう。(4)のスレッドはメッセージが入っ

ていた待ち行列のプライオリティで実行される。現在実行されているスレッドのプライオリティより高いプライオリティの待ち行列にメッセージが入れられると、その待ち行列のプライオリティを、実行されているスレッドは継承する。

4．スレッド間の相互排除

リングバッファに対する処理のスレッドは、1つの書き込みのスレッドと複数の読み出しのスレッドである。読み出しのスレッド同志では相互排除の必要はないが、書き込みのスレッドと読み出しのスレッドが同じレコードをアクセスした場合に相互排除する必要がある。ここでは、データの読み出しを行なうタスクに対して、リングバッファのレコードに関する情報を持たせる。情報の内容は、読み出しのタスクが現在指しているレコードread_recordと、read_recordから書き込みのタスクが現在指しているレコードwrite_recordまでのレコード数r1である。r1は、書き込みが行なわれるごとにインクリメントされ、読み出しや移動が行なわれるごとにデクリメントされる。第一の排他制御例は、r1≦0、すなわち、読み出しのアクセスが、書き込みを行なうレコードを越えて行なわれた場合である。この時、読み出しのタスクは次の2つのうちのどちらか一方をとる。

(1)アクセスしたレコードへの書き込みが終了するのを待ってから、処理を再開する。

(2)既に読み出されたレコード分のデータをエージェントに転送して、終了する。

第二の例は、r1＝(リングバッファのレコードサイズ)、すなわち、読み出しを行なおうとしているレコードに対して、書き込みのアクセスが行なわれた場合である。この場合は、データの書き込みを優先させ、そのレコードを上書きする。読み出しを要求したエージェントに対しては、無効の通知をする。

5．タスクスケジューリング

周期タスクのスケジューリング法として有名なものに、Rate Monotonic scheduling algorithm(RM)[1]がある。この方法では、周期が短いタスクから順に高プライオリティを設定する。各タスクのプライオリティはシステムの起動時に一度設計者が設定する必要があるだけで、スケジューリングのためのオーバーヘッドのあまり大きくない現実的な手法である。RMを用いて、m個の周期タスクをスケジューリングした場合、

$$U = m\left(2^{1/m}-1\right)$$

で表される稼働率よりも小さな値を持つタスクセットについては、各タスクのデッドラインが保証される。タスクが十分に多い場合には、U≒0.7となり、稼働率Uが70%を越えない場合には、タスクセットはデッドラインを越えることなくタスクの実行を終了できる。

この方法を用いて、周期タスクによる稼働率を計算し、残りをユーザインタフェース部より発生するデータ要求のイベントなどの非周期タスクに割り当てる。

6．おわりに

本稿では、リングバッファとディスクを用いて、最新の短期間のデータのリアルタイムの必要にも、長期間のデータの必要にも対応できるプラント運転支援用リアルタイムトレンドデータベースを紹介した。

今後は、データベースの信頼性、バックアップファイルの改善、ならびに、バックアップされるデータの管理方法の検討を行なう予定である。

【参考文献】

[1] C.L.Liu and W.James Layland. Scheduling algorithms for multiprogramming in a hard realtime environment. Journal of ACM,pages 46-81,1973.

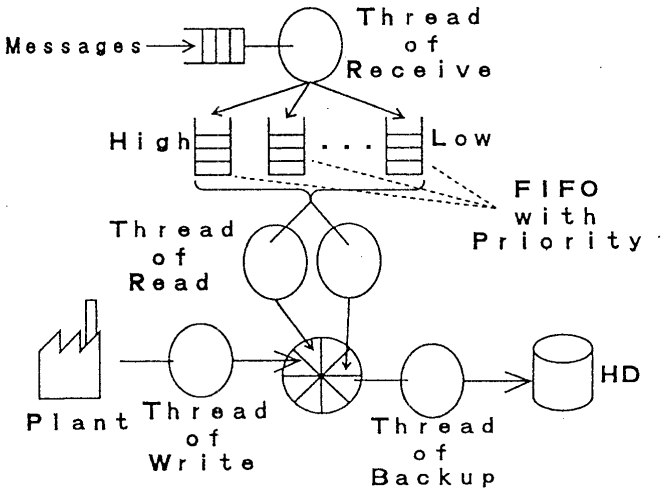

図1．リングバッファへのアクセス