データベースシステム 104−38
（1995. 7. 20）

周期・非周期データを統合する実時間データサーバ

井戸 譲治　島川 博光　浅野 義智　竹垣 盛一

三菱電機 産業システム研究所
〒661 尼崎市塚口本町 8-1-1

本稿では，プラント制御システムで取り交わされるデータを周期的 / 非周期的に獲得し上位の情報システムに提供する，実時間データサーバについて述べる．実時間データサーバは，制御システム間に張られた実時間ネットワークと，情報システムが接続されたオープンなネットワークとの間に位置し，制御システムの実時間制約を情報システムから隠蔽した上で，プラントデータを一元管理しオンラインで情報システムに提供する．実時間データサーバを用いることで，情報システムは制御システムの実時間制約を考慮する必要がなくなり，より柔軟で効率の良いシステム開発が可能となる．

Real-Time Data Server to Acquire and Serve Periodic/Aperiodic Data in Plant

Jouji IDO , Hiromitsu SHIMAKAWA ,
Yoshitomo ASANO , and Morikazu TAKEGAKI

Industrial Electronics and Systems Laboratory, Mitsubishi Electric Co.,
8-1-1 Tsukaguchi Honmachi, Amagasaki, Hyougo 661, Japan

In this paper, we discuss Real-Time Data Server (RTDS) which acquires data in a plant system periodically / aperiodically and serves it to information systems. Placed between a real-time network for control systems and open network for information systems, like Ethernet, RTDS manages plant data in a real-time manner and serves it to information systems in a on line manner. It conceals timing constraints for control systems from information systems. Using RTDS, we will be free from timing constraints for control systems in designing information systems, so we can develop plant systems more efficiently and flexibly.

1 はじめに

プラント等の監視制御システムは，実世界の状況をセンサ等を通じて取り込み，これをもとに制御量を導出し，コントローラを用いて実世界に働きかけるというサイクルで動作している．対象である実世界は計算機システムの状況とは無関係に変化してゆくため，計算処理は，対象の変化の速度に対応して，ある時間的制約を守りながら実行されなければならない．このように，ある既定時間内に処理が完了することを要求されるシステムは，リアルタイムシステムと呼ばれる．

従来，リアルタイムシステムの設計は，個々のプラントに応じて作り込みという形で，システム設計の熟練者がad hocな方法で行ってきた．そのため，同様の機能を持つプログラムを重複開発することになり，開発効率は高くはなかった．

このような中で我々は，個々の監視制御システムに共通な機能をミドルウェアとして設定し，これらを 実時間性を保証できるように設計することで，リアルタイムシステムの効率的な開発に寄与できると考えている．この考えに基づき，監視制御システムにおけるデータ収集・管理機能を提供するものとして，実時間データサーバ RTDS (Real-Time Data Server)[1] の開発を進めている．

監視制御システムのような実世界を対象とするシステムでは，外界データの収集・管理機能は必須である．この際，データ収集は，データ利用側の負荷に関わらず，常に時間制約を守って稼働していなければならない．即ち，データ利用システムはデータ収集の実時間性に影響を与えないように設計される必要があり，この点がシステム設計上のネックとなっていた．しかし，実時間データサーバを用いることで，データ利用側はデータ収集側の持つ時間制約から開放され，より自由な設計が可能となる．

以下，2章ではプラントデータ処理における問題点について指摘し，3章ではこれを克服するために開発中の実時間データサーバの全体像に触れる．4章ではデータのリアルタイムな獲得・管理方法について，5章ではデータの提供方法について説明する．最後に5章で課題等について述べる．

2 プラントデータ処理

監視制御システムでは一般に，外界プロセスの持つ固有周波数をもとに決定した最適のスキャン間隔でセンサにプロセス変数値を読ませ [9]，各センサ・コントローラで得られたデータはネットワークを経由してデータ収集系に格納，管理される．また，プロセス変数値のように周期的に送られるデータ以外にも，状態変化の通知などのように非周期的に送られてくるデータも存在する．

プラントデータ処理を獲得 / 保存 / 利用という3つの処理からなると考える．データ獲得処理は，保存 / 利用行なう上で前提となる処理であり，発生するデータを洩れなく獲得できなければならない．データの発生源は計算機システムとは無関係に変化するプラントであり，一度獲得に失敗した（ある時刻の）データは二度と獲得できないため，実時間要求は厳しいといえる．このようなタスクはハードリアルタイムタスク [5] と呼ばれる．データ保存はデータ獲得と連動して行なわれるべきものであり，これも実時間要求が厳しいと言えよう．

一方データ利用処理では，その内容によって実時間要求の強さに違いがある．例えば，刻々と変化するプラントの状況をグラフィカルに監視員に提示するような処理の場合，実時間要求は比較的厳しいものの，万一あるデータを提示できなかったとしても，そのデータがデータベースに保管され利用可能であれば，大きな問題にはならないであろう．このようなタスクはソフトリアルタイムタスク [5] と呼ばれる．また，一定期間のプラントの状態を帳票として印字するような処理は，特に実時間制約はない，即ちノンリアルタイムタスク [5] と言える．

このようにデータ獲得 / 保存 / 利用処理には，実時間要求に違いがある．センサ / コントローラからのデータを直接取り込んで利用するようなシステムの場合，データ利用処理プログラムの実行時間がデータ獲得 / 保存処理に大きく影響する．そのため，データ利用処理は，たとえそれ自身が実時間性を要求されないものであっても，データ獲得 / 保存処理に要求される実時間性を考慮した設計を余儀なくされる．このことが，従来リアルタイムシステム構築の上で大きな障害となっていた．

また，獲得及び保存といったデータ収集を行なうモジュールとしてデータロガーと呼ばれるものがあるが，多くの場合，獲得 / 保存処理の実時間性に影響を与えることなく，保管されたデータにオンラインでアクセスすることはできない．

そのため，プラントデータの獲得 / 保存を，その実時間制約をデータ利用処理に与えることなく行なうことができ，かつ保存されたデータをオンラインで提供できる機構が望まれている．

3 実時間データサーバ

前章で述べたようなリアルタイムシステム構築の際の問題点に対処するため，我々はプラントデータ獲得・提供システム実時間データサーバRTDSの開発を行なっている．

RTDSは，下位のコントローラ系と上位計算機系の中間に位置し，ネットワークを介して下位系から送られてくるプラントデータをリアルタイムに獲得・保持し，これらのデータを要求に応じて上位系に提供するものである．リアルタイムデータ収集・提供機能をサーバとして切り出すことで，リアルタイム性を保証したプラントシステムの構築が容易になる．

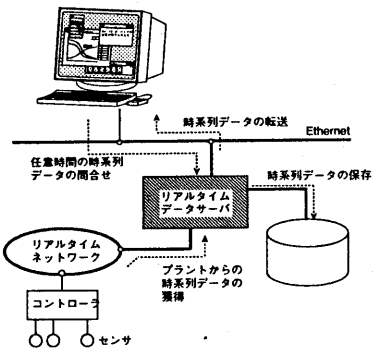

図 1: RTDSの位置付け

3.1 機能

履歴データ獲得・保存　プラントから送られてくるデータを周期的・非周期的に獲得し，時刻印を押印して，主記憶上に格納する．データは一定期間主記憶上に保持された後，ディスクに退避される．

履歴データ提供　獲得された履歴データは，以下の3種のタイミングで，上位システム（クライアント）に転送される．

- 周期転送
  定められた周期で最新データを転送する
- 要求駆動転送
  指定された期間のデータを指定された間隔で取りだし，転送する
- 事象駆動転送
  定められたイベント（状態変化）が起こった時にデータを転送する

状態変化通知　プラントの状態変化を検出し，上位システムに通知する．状態変化検出のための条件は，

- 新たに獲得された現時点のデータのみを参照するもの
- 履歴データを過去に遡って参照するもの

の2つに分類される．前者は，事故発生など即時性が要求されるものを対象としており，データ獲得の際に条件チェックがなされる．このため，この条件は獲得時条件と呼ばれる．後者はある期間におけるプラントの変化の兆候を判断するような，より複雑な条件を対象としており，提供プロセス内の状態変化駆動読みだしスレッド（図2）により条件チェックが行なわれる．

3.2 S/W構成

RTDSは，図2に示すように，マルチプロセス・マルチスレッド構成をとる．

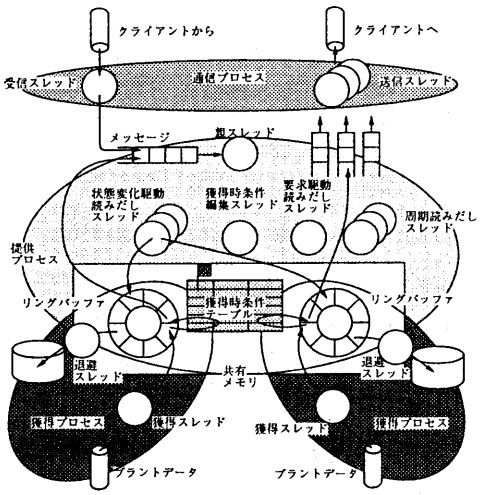

図 2: RTDSの構成

- 通信プロセス

受信スレッド クライアントからのメッセージの受信

送信スレッド クライアントへのメッセージの送信

- 提供プロセス

親スレッド クライアントからの要求解釈・他スレッドの起動管理など

周期読みだしスレッド 最新データの周期的な読みだし・通知

要求駆動読みだしスレッド 過去の指定期間のデータの読みだし・通知

状態変化駆動読みだしスレッド 状態変化判定条件のチェック及び条件成立時のデータ通知

獲得時条件編集スレッド 獲得時条件テーブルの書き換え

- 獲得プロセス

獲得スレッド プラントデータの獲得及び獲得時条件のチェック

退避スレッド データのディスクへの退避

3.3 リアルタイム性の保証

RTDS 内の各タスクは，Rate Monotonic Scheduling Analysis(RMA) [3] に基づいた優先度が設定され，また排他制御は優先度継承機能を持つセマフォを用いて行なわれる．これにより、設計時のリアルタイム性保証を可能としている．

4 データの獲得と管理

2章で述べたように，プラントでは様々な周期で，ないしは非周期的にデータが発生している．これらを並行に獲得するため，RTDS では，獲得の周期毎に獲得プロセス及び記憶構造を設けている．非周期的に発生するデータについては，その最小発生間隔を周期とみなすことで，同様に扱っている．また，データ管理機構については，本システムをプラントデータの獲得 / 提供に特化しているため，汎用データベースシステムのそれと比べて簡略化されている．

4.1 記憶構造

次々と発生するプラントデータを洩れなく獲得して一定期間保存するため，RTDS はリング状に構成された記憶構造を持つ．また，獲得されたデータを利用する際，最近のデータほど実時間要求が強く，過去のデータは比較的実時間要求が弱いため，リングバッファと呼ばれる主記憶上のリング構造と，リングファイルと呼ばれる，連続ファイルリング状に関連付けたディスク上のリング構造を用意した．

プラントネットワークから得られたデータは，タイムスタンプを付与された後，主記憶上のリングバッファに記録される．データは一定期間リングバッファ上に置かれた後，上書きされる前にディスク上のファイルに退避される．このファイルもリング上に連接されており，長い周期で上書きされていく．このようにして，最近のデータはリングバッファ上に，古くなったデータはリングファイル上に置かれるため，実時間要求の強い最近のデータは主記憶から，実時間要求の弱い過去のデータはディスクから読み出されることになる．リングバッファの大きさは，獲得周期と退避周期の関係から決定される．

4.2 排他制御

RTDS では，データの論理的正当性のみならず，各タスクの時間的正当性も保証されなければならない．獲得 / 退避処理は読みだし処理より優先して行なわれる必要があるが，読みだしデータの論理的正当性保証のためにロックを不用意に用いると，優先度の高い獲得 / 退避処理が長期間ブロックされ，時間的正当性を保証できなくなる恐れがある．

RTDS では，プラントから送られるデータの獲得・提供を目的としているため，保存しているデータに対する外部からの修正・削除は認めていない．また図 2に示すように，ひとつのリングバッファに対し対応する獲得スレッドが書き込みを，対応する退避スレッド及び提供プロセス内の複数の読みだしスレッドが読みだしを行なう．リングファイルには，対応する退避スレッドが書き込みを，提供プロセス内の複数の読みだしスレッドが読みだしを行なう．読みだしスレッドのうち周期読みだしスレッドは，獲得スレッドが更新した直後の最新のレコードをひとつ読み出すので，各々の周期に関連してリングバッファの大きさを適切に決めれば，上書きを回避できる．そのためデータの正当性が損なわれるのは，要求駆動及び事象駆動読みだしスレッドという，比較的優先度の低いスレッドが読み出し中のデータを獲得 / 退避スレッドが上書きしてしまう場合である．そこで，楽観的読みだし法を採用した．

データにアクセスする際，アクセスに必要なインデックスにはロックをかけるが，リングバッファ自体及びリングファイル自体にはロックをかけない．またこのロックには，優先度継承機能を有するセマフォを用いる．インデックスは小さいものであるため，ロックされる時間は無視できるほど小さくなる．

要求駆動及び事象駆動読みだしスレッドが過去のデータを読み出す際には，データの正当性を判別するため，最初にタイムスタンプを読み，各特徴変数を読んだ後，最後にもう一度タイムスタンプを読む．最初に読んだタイムスタンプと最後に読んだタイムスタンプの値が一致していなければ，そのレコードを読みだす過程で獲得スレッドに上書きされたと判断できる．このレコードは無効として破棄し，再度ファイルから読み出すことで補償される．

5 履歴データの提供

上位の計算機系でのプラントデータの利用方法としては，以下のようなものが挙げられる．

1. 最新データを一定周期で提示する（プラント状態の監視）
2. 特定の状態変化が生じた時にそれを通知する（プラント状態の監視）
3. 一定期間にわたるデータの推移を調べる（事後解析）

このうち 1. と 2. は実時間性が要求される，即ちデッドラインを持つが，最新の（獲得直後の）1 時刻分のデータの提供であるため，処理は比較的単純なものとなる．一方 3. は，明確なデッドラインを持たないものの，要求されるデータの量や内容が特定できないため，柔軟な処理が必要となる．

RTDS では，上記のようなデータ利用に対応するため，周期転送，要求駆動転送、事象駆動転送の 3 つのデータ提供法を用意する．周期転送では，定められた周期で定められた特徴変数の最新の値をクライアントに転送する．実時間性保証のため，最悪実行時間を見積もり，周期に応じた優先度を与えて RMA を適用する．事象駆動転送では，定められた条件の判定を行ない，条件が成立していれば，その時点でのデータを通知する．要求駆動転送では，指定された特徴変数を指定された期間分転送する．この際，要求駆動転送はデッドラインを持たないことから，この処理はバックグラウンドで行なわれる．

以下では，データモデルと要求駆動転送におけるデータの指定法，事象駆動転送で必要となる状態変化検出 / 通知機構について説明する．

5.1 履歴データと時系列

RTDS が扱うデータは，対象の状態を示す特徴変数の組と，それがサンプルされた時刻を表す時刻印から成るもので，履歴データと呼ばれる．履歴データを時間順に並べたものを時系列と呼び，時系列中の各履歴データを場面と呼ぶ．また，時系列中の場面が一定間隔で並んでいる時，この間隔を時系列のレートと呼ぶ．場面は時刻印を属性として持つ関係の組であり，時系列は関係表とみなすことができる．RTDS は，クライアントに指定された期間やレート等のパラメータに従って時系列データを作成，提供する．時系列の指定法については後述する．

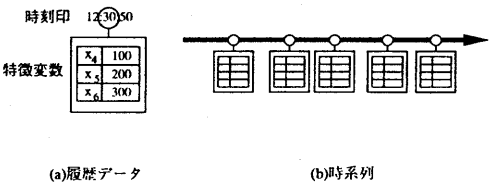

図 3: 履歴データと時系列

5.2 履歴データの分類

獲得される履歴データは以下の 2 つに分類できる．

サンプリングデータ コントローラの設定値やセンサの測定値などのプラントの状態を示すデータで，周期的に発生する．

イベントデータ プラントプロセスの開始・終了報告や事故発生報告など，状態変化を知らせるデータで，非周期的に発生する．

これらは時系列を作成する際，区別して扱われる．

周期（サンプリング）データの検索 周期データは，センサなどにより得られる物理量の測定値であり，測定の周期により観測精度が決まる．要求される観測精度すなわちサンプリング周期はデータを利用するクライアント

によって異なるため，RTDSはクライアントが要求した周期に従って履歴データを取り出せなければならない[1]．

非周期（イベント）データの検索　非周期データは，プラントの状態変化を知らせるデータで，基本的には離散値であり，非周期的に発生する．周期データがその値の変化に関わらず一定周期で獲得されるのに対し，非周期データはプラントの状態に変化が起こった時にのみ発生するデータであり，指定された期間の全てのデータを提供しなければならない．

5.3　時系列の問合せと応答

要求駆動転送では，以下のようなパラメータにより時系列を指定する．

- 時系列の基準 b
- 時系列のレート r
- 基準から見て過去方向への場面数 n_p
- 基準から見て未来方向への場面数 n_f
- 特徴変数 $v = \{v_1, \ldots, v_k\}$

時系列の基準は，時刻以外にも，特定の獲得時条件（後述）を指定することもできる．この場合は，指定された獲得時条件の成立が記録された履歴データのタイムスタンプを検索し，これを基準として用いる．各場面は，指定された各変数について，その時刻印の値が時区間

$$[b + ir, b + (i+1)r) \quad (-n_p \leq i \leq n_f)$$

に含まれるものをひとつ選びだし（代表値選択），これらを結合することで作成される．代表値選択としては，区間の先頭（最古），末尾（最新）など，いくつか考えられるが，これはユーザで定義するものとする．得られた場面のタイムスタンプには，その場面の時区間の開始時点である $b + ir$ を用いるものとする[2]．

ところで，既に述べたように，周期データと非周期データはその意味合いが異なるため，上記のような代表値選択は周期データに対してのみ行ない，非周期データは指定されたレート・場面数に関係なく，期間内のすべてのデータをそのまま用いて場面を作成する．そのため，結果として得られる時系列の場面数は指定されたものとは異なり，また空値を含むことになる．

例 1　以下の 2 つの時系列

s1

time	temp	pres
12:30:00	90	1.0
12:30:02	91	1.1
12:30:04	91	1.1
12:30:06	92	1.2
12:30:08	91	1.0
13:30:10	90	0.9

s2

time	temp	humid
12:30:00	18	60
12:30:01	18	60
12:30:02	18	60
12:30:03	19	61
12:30:04	20	63
12:30:05	20	65
12:30:06	20	65

に対し，b=12:30:00, r=3(sec), n_p=0, n_f=3, v= {s1.temp, s2.temp} を指定した場合，場面 1 を作成するため，それぞれの時系列から区間 [12:30:00, 12:30:03) に含まれるものをひとつ選択する．今，区間の先頭を選択するものとすると，s1 からは (12:30:00, 90, 1.0) が，s2 からは (12:30:00, 18, 60) が選択され，これらを用いて場面 (12:30:00, 90, 18) が作成される．場面 2 に対しては区間 [12:30:03, 12:30:06) に含まれるものとして

s1: (12:30:04, 91, 1.1) , s2: (12:30:03, 19, 61)

が選択され，場面 (12:30:03, 91, 19) を，場面 3 に対しては区間 [12:30:06, 12:30:09) から

s1: (12:30:06, 92, 1.2) , s2: (12:30:06, 20, 65)

を選択し，場面 (12:30:06, 92, 20) を作成する．結果として

result

time	s1.temp	s2.temp
12:30:00	90	18
12:30:03	91	19
12:30:06	92	20

を得る．

[1] もちろんクライアントの指定した周期が RTDS のデータ獲得周期より短い場合はこの限りではない

[2] この時には保管された履歴データのタイムスタンプと検索結果のそれとに誤差が生じる．この誤差が許容できるか否かは対象に依存するため，タイムスタンプを補正せずにそのまま用いる検索方法も用意し，選択可能としている．それについての説明は省略する

例2　以下のような周期データの時系列s3と，非周期データの時系列s4

s3

time	temp	pres
12:30:00	90	1.0
12:31:00	91	1.1
12:32:00	91	1.1
12:33:00	92	1.2
12:34:00	91	1.0
13:35:00	90	0.9
13:36:00	89	0.8

s4

time	batch#
12:31:20	1
12:33:40	2
12:36:00	3

に対し，b=12:30:00, r=2(min), n_p=0, n_f=4, v= {s1.temp, s2.batch#} を指定した場合，

result

time	temp	batch#
12:30:00	90	NULL
12:31:20	NULL	1
12:32:00	91	NULL
12:33:40	NULL	2
12:34:00	91	NULL
13:36:00	89	3

を得る．

このように要求駆動転送処理は，データ量を前もって特定できるものではないため，最悪実行時間を，実際の実行時間とそれほどかけ離れることなく見積もることは困難である．しかし，このような過去のデータの検索には明確なデッドラインが存在しないことから，これをバックグラウンドで処理することで，実用上問題にならないと考えている．

5.4　状態変化の検出

RTDSは，プラント状態変化を示すユーザ指定の事象を検出・通知する機能を持つ[2]．プラントの状態変化を検出するための条件には，事故発生を知らせるデータが送られてきた場合などのような，瞬時値に対して即座に判定されるべきものと，ある値が数秒間閾値を越え続けると異常とみなすといったような，一定期間の兆候を見て判定されるべきものがある．ここで後者は，常に条件判定を行わなくとも，前者がある条件を満たしたことが判明した時に条件判定すれば良い事が多い．

そこでRTDSでは，瞬時値に対する条件の判定を獲得スレッドに，一定期間の兆候に対する条件を事象駆動読みだしスレッドに行わせることにした．こうすることで，それほど頻繁には成立しない複雑な条件の判定にCPUを浪費することなく状態変化を検出することが可能となる．

5.4.1　獲得時条件判定

ある時点での瞬時値に対する条件の判定は獲得スレッドが行う．この条件は，獲得スレッドがデータを獲得する毎に，新たに獲得されたそのデータに対して検査されるため，獲得時条件と呼ばれる．このような条件は，RTDS稼働中，すなわちデータの獲得を継続しながら変更できる必要がある．また，データ獲得に付随して条件判定が行われるため，獲得処理の妨げにならぬよう処理時間が見積もれなければならない．そこで，獲得時条件としては獲得された変数値に対する線形の等式/不等式のみを許し，これらを記述したテーブルを共有メモリに置き，このテーブル中の条件を評価する機構を獲得スレッドに，編集する機構を提供プロセス内に置くこととした．

獲得時条件テーブルの1レコードは1条件に相当し，以下のようなフィールドを持つ．

- 条件名
- 項数
- 項（係数，変数）
- (不) 等号
- (最小) 値
- 不等号
- 最大値
- メッセージ（識別子，転送先，引数）

獲得スレッド内の獲得時条件テーブル評価機構は，対応するレコードを読み出して全項の線形和を求めた後，(不)等式として評価し，成立していれば，メッセージフィールドの記述に従って，親スレッドないしは転送スレッドにメッセージを転送する．

獲得時条件テーブルについても，排他制御のためのセマフォを設ける．獲得時条件を修正する際には獲得時条件編集スレッドが起動され，セマフォを獲得し，獲得時条件テーブルを書き換えた後，セマフォを開放する．一方獲得スレッドについては，編集スレッドによるデータ獲得処理のブロックを避けるため，セマフォ獲得に失敗すれば，待たずに次の処理を行なう．即ち，編集スレッドが獲得時条件テーブルにアクセスしている間は条件判

定が停止することになる．しかし，編集スレッドが獲得時条件テーブルに対しロックをかけるのは，テーブルを構成するレコードのリンクを繋ぎ替えるごく短い期間だけであり，さらに条件変更自体もそれほど頻繁に起こるものではないため，実用上問題はないと思われる．

5.4.2 一般条件の判定

新たに獲得されたデータのみならず，異なる周期で獲得されたデータや過去のデータをも対象とする条件（一般条件）の判定は，提供プロセス内の事象駆動読みだしスレッドにより行う．一般条件はある獲得時条件と関連づけられており，それを検査する獲得スレッドは，この獲得時条件が成立している場合，対応する一般条件を検査するための事象駆動読みだしスレッドの起動を親スレッドに要請する．親スレッドにより起動された事象駆動読みだしスレッドは，必要なデータをリングバッファあるいはファイルから読みだして時系列を作成し，条件を判定，結果をクライアントに通知する．

6 おわりに

本稿では，プラントデータ獲得・提供システムである実時間データサーバについて述べた．本システムにより，実時間性に厳しいデータ収集系からデータ利用系が切り離され，制御システムへの実時間性での悪影響を考慮することなくより自由にデータ利用システムを設計することが可能となる．

リアルタイム性の保証については現在，非周期的なデータ獲得・提供についても最小発生間隔を周期とみなしてRMAを適用している．しかしこれらの非周期タスクが支配的となる場合，RMAによりスケジュール可能とされるものと実際にスケジュール可能となるものの間に大きな開きが生ずるため，このような場合の実時間データサーバの挙動を予測する手法が必要である．非周期タスクのスケジューリングとしては非周期タスクサーバとRMAの併用[6]などが提案されており，それらの利用などが考えられる．

参考文献

[1] H.Shimakawa, H.Ohnishi, I.Mizunuma, M.Takegaki: Acquisition of Temporal Data for Real-Time Plant Monitoring, *Proc. of RTSS'93* (1993)

[2] 島川 博光，井戸 譲治，浅野 義智，竹垣 盛一: 実時間データサーバへのECA機能の導入，電子情報通信学会研究報告 CPSY94 - 116 ~ 124 , pp.49–56 (1995)

[3] L.Sha, S.S.Sathaye: Architectual Support for Real-Time Computing using Generalized Rate Monotonic Theory, 計測と制御, Vol.31, No.7 (1992)

[4] C.J.Date: *An Introduction to Database Systems*, 5th Edition, Addison Wesley (1990)

[5] K.Ramamritham: Real-Time Databases, *International Jounal of Distributed and Parallel Database*, Vol.1, No.2 (1993)

[6] B.Sprunt, L.Sha, J.Lehoczky: Aperiodic Task Scheduling for Hard-Real-Time Systems, *The Journal of Real-Time Systems*, Vol.1, No.1, pp27–60 (1989)

[7] 三巻 達夫，桑原 洋 編著: 制御用計算機におけるリアルタイム技術, コロナ社 (1986)

[8] 森下 巖 編: ディジタル計装制御システム, 計測自動制御学会 (1983)

[9] 松本 吉弘: リアルタイムシステム, 昭晃堂 (1984)