2G－7

# 実リアルタイムアプリケーションへの並列プログラミング支援環境の適応

和泉 秀幸　　中島 毅　　萩原 正敏

三菱電機（株）情報技術総合研究所

## 1　はじめに

近年、汎用WS上などでスレッドライブラリを利用した並列プログラムの開発が可能になってきている。しかし、一般に並列プログラムの開発は逐次プログラムと比較して難しく、開発を効率良く行うための環境が十分整っていない。このため我々は、デバッグやタイミング検証などを支援するためのマルチスレッドプログラム作成支援環境を開発しており、これらの実システムへの適用を進めている[1] [2]。

本稿では、ある実リアルタイムアプリケーションに対して時間制約条件の判定とそのふるまいの把握を容易するための並列プログラミング支援環境の機能拡張とその適用方法について述べる。まず対象であるリアルタイムアプリケーションについて説明し、そこでの並列プログラミング支援環境の適用上の問題点を述べ、その後問題を解決するための機能拡張と適用方法を述べる。

## 2　対象アプリケーションと問題点

### 2.1　対象アプリケーション

対象とするリアルタイムアプリケーションは監視・制御システムの操作端末用S/Wであり、1) システム内で採取した情報の表示と2) ユーザ操作をシステムへの制御命令として発行する処理を行う。各処理ごとに異なった時間制約条件が設定されている。

アプリケーションは、HP-RT[1]上のpthreadライブラリを利用したマルチスレッドプログラムとして実現している。アプリケーションの特徴として、監視対象の変化通知やユーザ操作などの外部入力を受け取った時に対応する処理を実行する機能がある。

次にこの機能を示す(この機能を“ディスパッチ機能”と呼ぶ。図1参照)。

- 外部入力を受け取る“登録スレッド”と対応する処理を行う“実行スレッド”が存在する。
- 登録スレッドは外部入力の投入時に対応する処理と優先度を“ディスパッチテーブル”へ登録する。
- 実行スレッドはディスパッチテーブルを周期的にチェックし、登録されている処理を優先度に従って実行する。
- 複数の実行スレッドが、1つのディスパッチテーブルをチェック・実行することが可能。

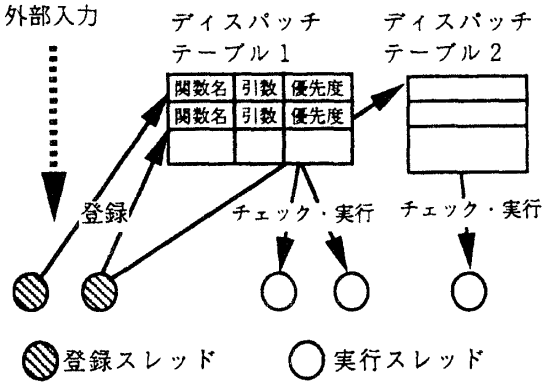


図1 ディスパッチ機能

### 2.2　適用時の問題点

対象アプリケーションに対してデバッグとチューニングを行うためには、各外部入力の投入時刻とその入力に対する処理の終了時刻と処理を実行したスレッドを把握することが重要である。また、時間制約条件を判定するためには、外部入力の投入時から処理終了までの時間を計測する手段が必要になる。

我々は、OSF/1[2]上でマルチスレッドプログラムの実行中のタイミングを評価するための“タイムログ表示ツール”を既に開発している[1]。このツールは次のような機能を持っている。

- 対象プログラムのソースコードの任意位置にイベント発生ポイントを設定。

---

Parallel Programming Support Environment for a Realtime Application,

Hideyuki IZUMI, Tsuyoshi NAKAJIMA, Masatoshi HAGIWARA,
Information Technology R & D Center,
Mitsubishi Electric Corp.
5-1-1 Ofuna, Kamakura, Kanagawa 247, Japan

[1] HP-RTはHEWLETT PACKARD社の商標です

[2] OSF/1はOpen Software Foundation,Inc.の商標です

- 対象プログラムを実行して各ポイント通過時に 1) 通過時刻と 2) 実行スレッドと 3) 位置 (ファイル名・行数) をログデータとして採取。
- 採取したログをグラフィック表示。

タイムログ表示ツールは、OSF/1 上のマルチスレッドプログラムの実行タイミングを評価するのに有効である。しかし、このツールを用いて本稿の対象アプリケーションを評価するには次のような問題がある。

本稿のアプリケーションは、外部入力を受け取る登録スレッドと対応する処理を実行する実行スレッドが別々である。このため、上記のタイムログ表示ツールでは次の問題があることが明らかになった。

- 通過時刻と実行スレッドと位置情報だけでは、外部入力に対する登録と実行の対応関係を把握しづらい。
- 外部入力の投入時から処理終了までの時間の計測手段として利用できない。

例えば、ディスパッチテーブルの処理を２つの実行スレッド (th1、th2) が行う。同じ外部入力が２回 (E1、E2) あり、これに対応する処理が別々の実行スレッドで実行されたとする。この時の処理終了時のログデータを表１に示す。

表１ 問題となるログデータの例

| 時刻 | スレッド | 場所 |
|---|---|---|
| 2045 | th1 | sample.c 80 |
| 3078 | th2 | sample.c 80 |

このとき、どちらのログが外部入力 E1、E2 に対応するかを判定する手段が存在しない。このため、外部入力の投入時から処理終了までの時間の計測手段として利用できない。また、外部入力に対する登録と実行の対応関係も把握できない。

## 3 適用方法

前記の問題を解決するため、ディスパッチ機能を pthread ライブラリ上の“ディスパッチルーチン”として提供する。ディスパッチルーチンは並列プログラミング支援環境の１部として実装する (図２参照)。アプリケーションはディスパッチルーチン内のディスパッチ機能を使用して実現する。

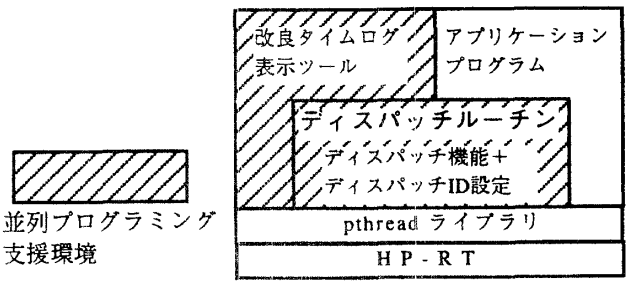


図２ ディスパッチルーチン

このディスパッチルーチンに対して、アプリケーションが本来の処理に必要とするディスパッチ機能に加えて、ログ情報採取時に次の情報を採取する機能を埋め込んだ。

- 各外部入力に“ディスパッチ ID”を設定。
- ログ出力時に ディスパッチ ID を付加。

表１の問題の例で、ディスパッチルーチンを使用した場合のログを表２に示す。ディスパッチ ID を加えることにより、各外部入力に対する処理を把握できるようになる。

また、タイムログ表示ツールについてもディスパッチルーチンに対応するための機能拡張を行い、外部入力に対する登録と実行の関係を追跡可能な“改良タイムログ表示ツール”とした。

表２ ディスパッチルーチン使用時のログデータの例

| 時刻 | スレッド | 場所 | ディスパッチ ID |
|---|---|---|---|
| 2045 | th1 | sample.c 80 | E2 |
| 3078 | th2 | sample.c 80 | E1 |

## 4 まとめ

ディスパッチ機能を持つアプリケーションに対して評価を行うためには、通過時刻と実行スレッドと位置情報だけでは十分でなく、外部入力に対する登録と実行の対応関係を把握することが必要であることを明らかにした。

この対応関係を把握するために、ディスパッチルーチンと改良タイムログ表示ツールを開発し、特定のアプリケーションで外部入力に対する登録と処理実行の対応づけを可能にした。これにより、アプリケーションの実行状況の把握を容易にし、時間制約条件の判定が可能になった。

ディスパッチルーチンを改良タイムログ表示ツールとともにプログラミング環境の一部として提供した。ディスパッチルーチンを使用することにより、ディスパッチ機能だけでなく改良タイムログ表示ツールも同種のアプリケーション開発で利用できると考えている。

## 参考文献

[1] 福地 雄史、石塚 章子、和泉 秀幸,“マルチプロセッサ対応 UNIX 上での並列プログラム開発支援環境の開発”, 本会第 48 回全国大会 2G-9, 1994.3.

[2] 和泉 秀幸、福地 雄史,“デッドロック解析・検知機能を持つマルチスレッド対応デバッガの開発”, 本会第 49 回全国大会 4U-1, 1994.9.