5J-01

アスペクト指向プログラミングによる並列・分散アラーム管理機能を有するリアルタイムＯＳの実現

溝呂木裕規† 横山孝典† 兪明連†
東京都市大学‡

1. はじめに

組み込み制御システムは自動車や産業用機器等の多様なアプリケーションに使用される．しかし，リソース消費量の制約が大きい組込みシステムにおいて，多様なアプリケーションに要求されるすべての機能を備えたリアルタイム OS を搭載する事は困難である．そこで，アプリケーションの要求に応じて必要最小限の機能を備えたリアルタイム OS を提供することが望まれる．

そこで我々は，構成管理の容易なリアルタイム OS ファミリを実現することを目的にアスペクト指向プログラミングを用いて，既存のリアルタイム OS に対して必要な機能の追加・変更を可能とするリアルタイム OS のファミリ化の研究を行ってきた．そしてこれまでに OSEK OS 仕様[1]に基づくリアルタイム OS の TOPPERS/ATK1 [2]を基に，タスク管理及びイベント制御のマルチコア対応や分散処理対応[3]を行ってきた．

本研究では，CPU コア間及び、ノード間でアラーム管理を可能とするアスペクトを提案する．

2. アラーム管理機能

アラームは特定のタイミングで発生するイベントを扱う機能である．アラームの代表的な使い方として，OS のシステム時刻をカウントするシステムカウンタを用いて，設定された時刻に周期タスクの起動やイベントの設定を行うことが挙げられる．

本研究では，ほかの CPU コア上及びほかのノード上のアラームを制御するための拡張を行う．拡張対象のシステムコールは，指定するアラームが満了するまでの相対時間を取得する GetAlarm()，指定するアラームの構造体に格納する GetAlarmBase()，相対時間でアラームを設定する SetRelAlarm()，絶対時刻でアラームを設定する SetAbsAlarm()，アラームをキャンセルする CancelAlarm()の５つである．

A Real-Time Operating System with Parallel and Distributed Alarm Management Functions Based on Aspect-Oriented Programming
† Hiroki Mizoroki , Takanori Yokoyama and Myungryun Yoo
‡ Tokyo City University

3. アスペクト

3.1. アスペクト指向プログラミングによるカスタマイズ

アスペクト指向プログラミングは，横断的に散在する関心事を分離してモジュール化する手法である．図１に示すように追加・変更する機能をオリジナルのソースコードと分離してアスペクトで記述し，織り込むことで，ソースコードを直接修正することなく，ソフトウエアの機能の追加・変更が可能になる．必要機能のアスペクトを取捨選択して織り込むことで，アプリケーションの要求に応じて必要最小限の機能を備えたリアルタイム OS を提供できる．本研究では，アスペクト指向プログラミング言語として ACC[4]を用いる．

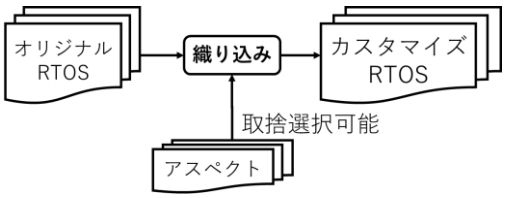

図１：アスペクトによる機能の追加・変更

3.2. アラーム管理機能用アスペクト

SetRelAlarm を例に，コア間及びノード間の処理を追加するアスペクトコードを図２に示す． execution ポイントカットを用いて SetRelAlarm の関数が実行されるときを指定し，around アドバイスを用いて，代替して要求対象のアラームが存在する CPU に応じたアラーム設定処理を織り込む．

```
StatusType around():
execution(StatusType SetRelAlarm()){
    switch(位置){
        case ローカル:
            アラーム設定();
        case コア間:
            コア間アラーム要求();
        case ノード間:
            ノード間アラーム要求();
}
```

図２：アスペクト記述

図 3 に，アラームを設定するシステムコール SetRelAlarm を発行したときの処理の流れをシーケンス図で示す．図 3 の赤い円形はポイントカットで指定したジョインポイントであり，図 4 に示すアスペクトの処理を織り込む位置を示す．橙色の枠部では要求対象のアラームが存在する CPU によって分岐する．

図 5 に，アスペクト織り込み後の SetRelAlarm の処理の流れをシーケンス図で示す．

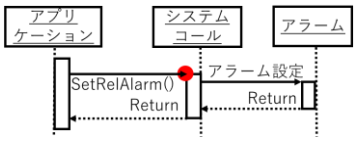

図 3：オリジナルの処理

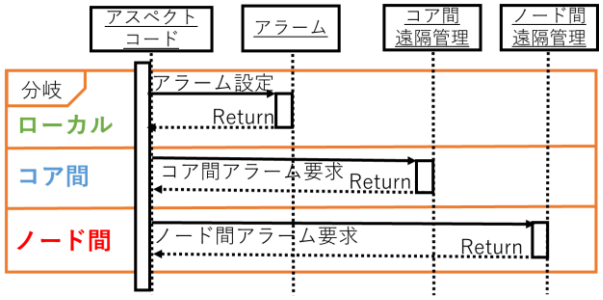

図 4：アスペクトの処理

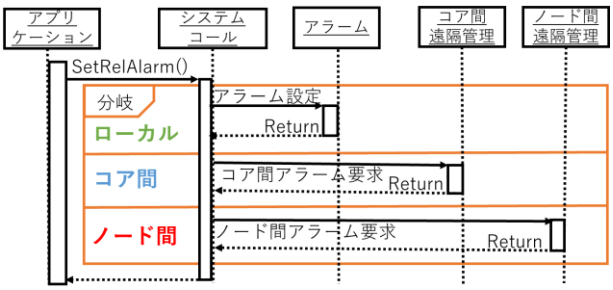

図 5：織り込み後の送信側の処理

4. 評価

評価には，デュアルプロセッサ SH7205 を搭載するルネサス製の評価ボード M3A-HS50 を用いる．100 回ずつ測定し，その平均を求める．ソースコードを直接修正して機能を拡張した直接修正版リアルタイム OS と，アスペクト指向プログラミングを用いて機能を拡張したアスペクト版リアルタイム OS の 2 つの処理時間を比較する．

CPU コア間の応答時間は，システムコールが発行されてから値が返されるまでを計測する．ノード間の応答時間は，通信時間がネットワーク状態により大きく変動するために，ネットワーク通信時間を除いた値を測定する．測定結果を表 1 に示す．

表 1 ：アラーム管理機能の評価

システムコール名	処理時間[μsec]			
	直接修正版		アスペクト版	
	コア間	ノード間	コア間	ノード間
GetAlarm	3.52	4.82	3.55	4.82
GetAlarmBase	4.61	6.30	4.61	6.33
SetRelAlarm	4.06	10.73	4.09	10.76
SetAbsAlarm	4.06	10.85	4.09	10.91
CancelAlarm	3.45	4.79	3.45	4.79

表 1 より直接修正版とアスペクト版で約 0.03μ 秒の差が生じている．マルチコア並列・分散リアルタイム OS が適用対象と考えている多くの制御システムでは，タスクの周期は最小で約 1m 秒であるが，アスペクト版の処理時間の増大は 1m 秒に対して十分に小さいため，実用上問題のない性能であると考える．

ノード間の場合の応答時間はネットワーク通信時間が加わる．ネットワークのトラフィックが少ないときのデータ転送速度 500kbps の CAN の通信時間は約 700μ 秒である．したがって，アスペクト版の応答時間は最大で約 710.91μ 秒である．一般に，自動車制御用アプリケーションの場合，CAN 通信を用いるタスクの制御周期は 10m 秒から 100m 秒程度で，それと比較して応答時間は十分に小さく，実用上問題のない性能であると考える．

5. おわりに

構成管理の容易なリアルタイム OS ファミリを開発することを目的に，アスペクト指向プログラミングを用いてアラーム管理機能の並列，分散化を実現する機能を追加する手法を提案した．また，各システムコールの実行時間を計測し，アスペクト指向プログラミングによるオーバヘッドは十分に小さく，実装したリアルタイム OS が実用上問題のない性能であることを確認した．

謝辞

本研究で使用した ACC および TOPPERS/ATK1 の開発者に感謝する．本研究は JSPS 科研費 JP18K11225, JP21K11815 の助成を受けたものである．

参考文献

[1] OSEK/VDX: Operating System, Version 2.2.3 (2005).
[2] TOPPERS Project:https://www.toppers.jp/
[3] Y. Harada, M. Yoo, T. Yokoyama, A Distributed Multicore Real-Time Operating System Family Based on Aspect-Oriented Programming, 19th IEEE ICIT, pp.1389-1394(2018)
[4] AspeCt-oriented C : https://sites.google.com/a/gapp.msrg.utoronto.ca/aspectc/