C-004

アスペクト指向プログラミングを用いたリアルタイム OS のカスタマイズにおけるオーバヘッドの評価

An Evaluation of Aspect-Oriented Customization for Real-Time Operating Systems

原田 祐輔 * Yusuke Harada　**兪 明連** * Myongryun Yoo　**横山 孝典** * Takanori Yokoyama

1. はじめに

組込みシステムは様々な用途に使用されるため，必要となるリアルタイム OS(RTOS) の機能もアプリケーションによって異なることが多い．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，それら全ての機能を備えた RTOS を搭載する事は困難である．このため，単一の RTOS ではなく，アプリケーションの要求に応じて必要最小限の機能のみを備えた RTOS を選択可能な RTOS ファミリーが望まれている．そしてその実現には，効率的に RTOS のファミリー化を行う手法が必要となる．

横断的関心事を分離してモジュール化することのできるアスペクト指向プログラミング [1] を用いて，RTOS をカスタマイズする研究がいくつかなされている [2]．例えば Lohmann らは，AUTOSAR OS 仕様に基づき必要な機能のみを，アスペクトによって取捨選択できる RTOS ファミリーを提案している [3]．アスペクト指向プログラミングを用いることで，ソースコードを直接修正せずに，RTOS の機能の追加・変更が可能になり，RTOS のソースコードの保守性を向上できる．

アプリケーションに応じて取捨選択したい機能のひとつにタスクスケジューリングがある．既存の RTOS の多くは固定優先度スケジューリングを採用しているが，アプリケーションによっては動的スケジューリングが求められる場合もある．しかし，スケジューリングのような大きなアルゴリズム変更を行う研究は少ない．

そこで我々は，アスペクト指向プログラミングを用いてスケジューリングアルゴリズムをカスタマイズする手法を提案した [4]．本発表では，アスペクト指向プログラミングによるオーバヘッドを，コンパイラによる最適化を含めて評価し，実用上問題ない程度に抑えられることを示す．

2. アスペクト指向プログラミングによるカスタマイズ

RTOS ファミリーが提供する機能を表したフィーチャモデル (一部) を図 1 に示す．自動車制御分野の標準である OSEK OS 仕様を対象に拡張し，固定優先度スケジューリングのみでなく動的優先度スケジューリングである EDF(Earliest Deadline First) スケジューリングとまたは，RMCL(Rate Monotonic Critical Laxity) スケジューリングを選択可能とする．また，スケジューリングアルゴリズムに応じてリソースアクセスプロトコルを置き換える．

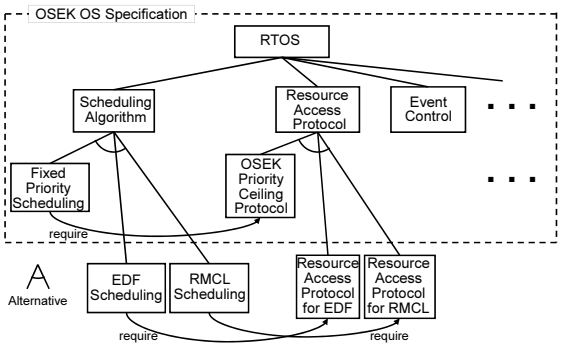

図 1: RTOS ファミリーのフィーチャモデル

表 1: カスタマイズに用いるアスペクト

Aspect		EDF	RMCL
Scheduling	Ready Queue Operation	x	x
	Selection of Task to Run	x	x
	Preemption	x	x
	Scheduler Call	x	x
	Remaining Execution Time Maintenance	-	x
	Absolute Deadline Update	x	
	Task Initialization	x	x
Resource	Resource Management	x	x
	Ceiling Priority Maintenance	x	-
	Search Shared Resource	-	x
	Resource Occupation Maintenance	x	
	Resource Initialization	x	x

固定優先度スケジューリングを EDF スケジューリングまたは RMCL スケジューリングにカスタマイズするためのアスペクトの一覧を表 1 に示す．EDF スケジューリング向けに，デッドラインが近いタスクの選択やキューイングなどのためのアスペクトを 5 個定義し，EDF 向けリソースアクセスプロトコルのためにタスクのデッドラインを優先度としたリソースの上限優先度管理などのアスペクトを 4 個定義する．また，RMCL スケジューリング向けには，タスクのデッドラインまでの時間と残り実行時間の差である余裕時間を用いた優先度管理などのためのアスペクト 6 個定義し，RMCL 向けリソースアクセスプロトコルのために，余裕時間を優先度としたリソースの上限優先度管理などのアスペクトを 5 個定義する．うち，タスクのデッドライン管理とリソース使用状況管理を行う 2 個のアスペクトは共通である．

カスタマイズする RTOS として TOPPERS/ATK1

*東京都市大学

表 2: システムコールの実行時間

System Call	Optimizing Option	Execution Time[μsec]				
		EDF		RMCL		Fixed
		AOP	Rewrt	AOP	Rewrt	
Activate	none	143.7	115.80	280.1	232.8	67.9
Task()	O2	64.7	57.2	149.5	142.9	40.5
Get	none	57.8	45.8	61.3	49.2	36.6
Resource()	O2	27.1	27.1	31.1	31.7	20.6
Release	none	67.3	55.3	213.5	176.1	38.0
Resource()	O2	36.2	36.2	118.4	113.2	21.7

表 3: メモリ消費量

RTOS		Memory Consumption [Byte]					
Scheduling	Optimizing Option	Code		Data (ROM)		Data (RAM)	
		AOP	Rewrt	AOP	Rewrt	AOP	Rewrt
EDF	none	20236	18340	840	840	145	149
	O2	15080	13672	832	832	144	145
RMCL	none	21896	19612	856	856	143	143
	O2	15948	14124	848	848	139	139
Fixed	none	-	16718	-	820	-	137
	O2	-	12436	-	812	-	133

Release 1.0[5] を用いた．アスペクト指向言語には ACC ver0.9[6] を C コンパイラは GCC 3.4.6 を用いる．

3. オーバヘッドの評価

3.1. 実験環境

システムコールの CPU 実行時間とメモリ消費量の評価を行う．C コンパイラの最適化オプションによる影響を考慮し，TOPPERS/ATK1 の makefile で指定している最適化オプション O2 の場合と最適化を行わない場合でについてオーバヘッドを評価する．

本評価に用いた実験環境として，マイクロコントローラ H8S/2638F を搭載した評価ボードを使用する．H8S/2638F は 256kB の ROM, 16kB の RAM を内蔵し，クロック周波数は 20MHz である．

3.2.CPU 実行時間の評価

アスペクト指向プログラミングによる CPU 実行時間のオーバヘッドを評価するため，カスタマイズした機能に関わるシステムコールについて実行時間を測定し，ソースコードを直接書き換えて実装した同一機能の RTOS の実行時間と比較する．実行時間の測定には 5MHz のハードウェアタイマを用いる．

表 2 にシステムコールの CPU 実行時間を示す．表に示した値は，タスク管理については最もよく使用する ActivateTask()，リソース管理については GetResource() と ReleaseResource() についてのシステムコールの発行から終了までにかかる時間を 100 回計測した平均値である．参考のため，オリジナルの TOPPERS/ATK1 での固定優先度の場合の実行時間も示す．

スケジューリングの Fixed，EDF，RMCL はそれぞれ固定優先度スケジューリング，EDF スケジューリング，RMCL スケジューリング搭載の RTOS を表す．また，AOP はアスペクト指向プログラミングによりカスタマイズした場合，Rewrt はソースコードを直接修正してカスタマイズした場合を表す．AOP の場合と Rewrt の場合の実行時間の差がアスペクト指向プログラミングによるオーバヘッドを示す．

最適化を行わない場合において最もオーバヘッドの大きい RMCL スケジューリングの ActivateTask() では，20%程度，47μsec 程度増大する．しかし，最適化を行うと，4%程度，6μsec 程度の増大となり，オーバヘッドは十分小さく抑えられることがわかる．また，最適化を行わない場合でもオーバヘッドが小さいリソース管理のシステムコールにおいては，同一のコードに最適化されオーバヘッドが発生しない．

以上の評価から，広く用いられている O2 程度の最適化を行える状況において，アスペクト指向プログラミングのオーバヘッドは実用上問題ない範囲と考える．

3.3. メモリ消費量の評価

アスペクト指向プログラミングによるメモリ消費量の増大を評価する．

表 3 に，コード領域，ROM 上のデータ領域，RAM のデータ領域のメモリ消費量を示す．

最適化を行わない場合は，コード領域で 10〜11%程度，1.8kB〜2.2kB 程度増加する．最適化を行うと，コード量は 10〜12%程度，1.4kB〜1.8kB の増加となる．一方データ量についてはほぼ同程度で，増大はない．アプリケーションを含めたシステム全体のメモリ消費量 (H8S/2638F の場合，コードが書き込まれる ROM 容量は 256kB) に対する増加量は 1%以下であることから，実用上問題ない範囲と考える．

4. おわりに

OSEK OS 仕様に基づく RTOS の固定優先度スケジューリングを，EDF または RMCL スケジューリングにカスタマイズする場合についてアスペクト指向プログラミングのオーバヘッドをコンパイラによる最適化を含めて評価し，実用上問題ない程度に抑えられることを示した．

今後の課題として，他の最適化オプション等も含めて評価を行い，より詳細なオーバヘッドの原因を明らかにすることが挙げられる．

謝辞

本研究で使用した TOPPERS/ATK1 の開発者と ACC の開発者に感謝する．本研究は JSPS 科研費 2450046 および 15K00084 の助成を受けたものである．

参考文献

[1] Kiczales, G. et al., Aspect-oriented programming, *ECOOP'97 — Object-Oriented Programming* (Akşit, M. and Matsuoka, S., eds.), Lecture Notes in Computer Science, Vol. 1241, Springer Berlin Heidelberg, pp. 220–242 (1997).

[2] Afonso, F. et al., Applying Aspects to a Real-time Embedded Operating System, *Proceedings of the 6th Workshop on* , ACP4IS '07, (2007).

[3] Lohmann, D. et al., Transactions on Aspect-Oriented Software Development IX, Springer-Verlag, Berlin, Heidelberg, chapter The Aspect-aware Design and Implementation of the CiAO Operating-system Family, pp. 168–215 (2012).

[4] 原田祐輔 ほか，アスペクト指向プログラミングによるリアルタイム OS スケジューラのカスタマイズ，情報処理学会論文誌，vol.57, No.8, (2016) to appear.

[5] *TOPPERS Project, http://www.toppers.jp/.*

[6] *AsceCt-oriented C, https://sites.google.com/a/gapp.msrg.utoronto.ca/aspectc/.*