3N-1 リアルタイムJava VMにおけるスレッドマネージャの設計

奥山 玄† 瀧本 栄二† 芝 公仁† 大久保 英嗣††
†立命館大学大学院理工学研究科 ††立命館大学理工学部情報学科

1 はじめに

インターネットの普及にともない，ネットワークに接続される家電や携帯端末への注目が高まっている．それとともに，ネットワークを通してオンデマンドにプログラムをダウンロード可能であるJavaを，組み込みシステムに適用しようとする動きが活発になってきている．Javaは，マルチプラットホーム，ソフトウェア生産性の向上などの利点を持つ．そこで我々は，リアルタイムオペレーティングシステム (以下RTOS) へのJava VM(Java Virtual Machine) の実装を検討している．しかし，Javaの欠点として，使用メモリが多い，実行速度が遅い，リアルタイム性がないなどが挙げられる．その要因は，バイトコード処理系のオーバヘッドや，ガーベジコレクション (以下GC) の必要性などである．我々は，以上の問題点を考慮し，現在，開発を進めているRTOS Easelへ Java VMを実装中である．

以下，本稿では，2章でJava VMの実装モデル，3章でJava VMにおけるスレッドマネージャについて述べる．また，最後に4章でまとめについて述べる．

2 Java VMの実装モデル

RTOSにおけるJava VMの実装モデルを図1に示す．Java VMは，インタプリタ，クラスローダ，ガーベジコレクタ，スレッドマネージャから構成される．

従来のJava VMでは，時間的制約を持つJavaアプリケーションを実行させることが困難であった．リアルタイム環境におけるJavaの問題点として，以下が挙げられる．

- クラスのダイナミックローディング
- GC
- スレッド管理

Java VMは，プログラムの実行に必要最小限のクラスを必要にあわせて動的にロードし実行する [1][2]．そ

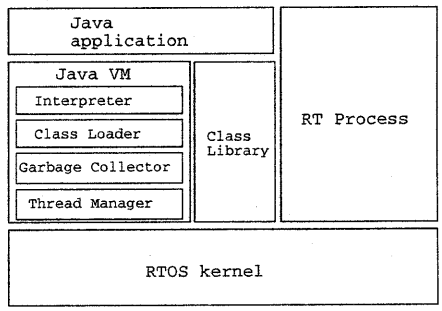

図1 RTOSにおけるJava VMの実装モデル

の際，Java VMは，プログラムを中断し，クラスのロードに専念する．また，クラスのロードにかかる時間は予測が困難であるため，アプリケーションの最悪実行時間を予測することは困難となる．

本Java VMでは，ロード時にかかる時間を削減するために，アプリケーション実行前に，その中で使っているクラスやそのメソッドの参照情報を整理する．これらの処理を静的なリゾルブという．静的なリゾルブを行うことにより，アプリケーション実行時に，リゾルブにかかる予測が困難な時間を削減することが可能である．このことから，アプリケーションの最悪実行時間を予測することが容易となる．

GCは，ヒープ領域に割り付けられたオブジェクトがどこからも参照されていない場合に，そのヒープを解放するための機構である．しかし，従来のJava VMでは，GC中は割り込み禁止状態としている．また，GCにかかる時間を予測することは困難である．これらのことから，リアルタイム応答性を要求するシステムにおいては，デッドラインミスなどの問題が発生する．

この問題を解決するために，GCをスレッドとして実現し，プリエンプト可能なリアルタイムGCを設計，実装する．GCのアルゴリズムとして，マーク&スイープ方式を採用する．この方式は，参照されているオブジェクトにマークを付け，参照されていないオブジェクトを解放するという2つのフェーズから構成される．

しかし，GCが動作中に割り込みが発生し，他のスレッドにプリエンプトされた場合，GC再開時にGCの状態と，メモリ割当て状態の不整合が発生する可能性がある．この問題を解決するために，オブジェクトにマークする際，GCだけでなく参照するスレッドもそのオブジェクトにマークする方法をとる．これにより，GCが

Thread Manager for a Real-Time Java VM

Gen Okuyama†, Eiji Takimoto†, Masahito Shiba†, and Eiji Okubo††
†Graduate School of Science and Engineering, Ritsumeikan University
††Department of Computer Science, Faculty of Science and Engineering, Ritsumeikan University

中断している際にスレッドが新しいオブジェクトを参照しても，メモリ割り当て状態の不整合が発生することはない．

Javaスレッドの管理を行うスレッドマネージャについては次章で述べる．

3 スレッド管理

JavaスレッドとRTOSのネイティブスレッドとの関係を図2に示す．Java VMは1つのプロセスとし，その中で複数のJavaスレッドが動作する．RTOSカーネルは，Javaスレッドを意識することなく，ネイティブスレッドをそのデッドラインに基づいてスケジューリングする．図に示すように，Javaスレッドは，RTOSのネイティブスレッドに1対1にマッピングされる．これにより，JavaスレッドをRTOSのネイティブスレッドと同等に管理することが可能であり，リアルタイム性が向上する．

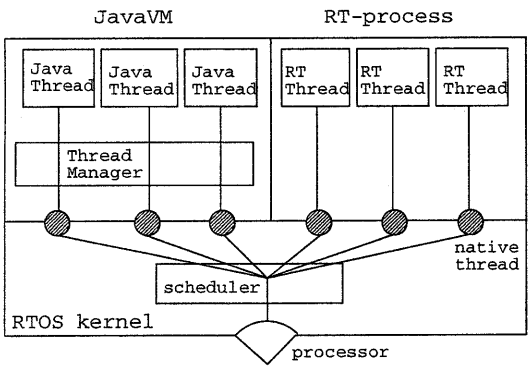

図2 JavaスレッドとRTOSのネイティブスレッドとの関係

本研究では，先に述べたリアルタイム環境におけるJavaの問題点を考慮し，Java VMのリアルタイム拡張を行った．以下では，Javaスレッドの管理を行うスレッドマネージャについて述べる．スレッドマネージャは，以下の機能を提供する．

- リアルタイムJavaスレッドの生成
- 例外処理機構
- スレッド間の同期機構

3.1 リアルタイムJavaスレッドの生成

Javaスレッドにリアルタイム性をもたせるため，RTThreadクラスを新たに追加した．リアルタイムJavaスレッドの生成手順を以下に示す．

(1) RTThreadクラスがnewされたとき，そのコンストラクタの中でネイティブメソッドを呼び，スレッドマネージャにスレッド生成を要求する．

(2) スレッドマネージャは，clone()システムコールによりスレッドを生成する．このスレッドは，Javaスレッドにマッピングされる．

(3) 生成されたJavaスレッドは，RTThreadクラスのメソッドであるsetDeadline()により，デッドラインを設定する．

また，デッドラインミス時の処理は，Java言語のtry-catch文を用い，例外処理として実現する．スレッドマネージャは，カーネルからデッドラインミスのシグナルを受け，カレントスレッドへ例外をthrowする．例外ハンドラは，ユーザが記述することが可能であり，またユーザは，例外処理を行う際の優先度を設定することが可能である．

3.2 スレッド間同期

オブジェクトのロック，アンロックは，Java VMがサポートするmonitorenter，monitorexitを利用する．本研究では，その際に発生する優先度逆転を考慮し，優先度継承プロトコル[3]を採用する．

ある優先度の高いスレッドT_Hが，あるオブジェクトOをロックしようとする場合を考える．このとき，Oがより優先度の低いスレッドT_Lによってロックされている場合，T_Lは，T_Hの優先度を継承する．これにより，T_LがOをロックしている際に実行可能状態となった中間の優先度をもったT_Mによってプリエンプトされることはない．また，優先度逆転となる時間は，T_LがOを解放するまでの時間となる．

スレッドマネージャは，ロックしているスレッド，ロックしようとするスレッドの優先度を考慮し，継承する機能を提供する．

4 おわりに

本稿では，主にスレッド管理に焦点を当て，RTOS上でのJava VMのリアルタイム化手法について述べた．Javaをリアルタイム環境で実行させるには，Java VMレベルで何らかの拡張が必要である．

参考文献

[1] Jon Meyer, Troy Downing 共著，鷲見 豊 訳: “Javaバーチャルマシン”，株式会社オライリー・ジャパン (1997).

[2] ティム・リンドホルム，フランク・イェリン 共著，野崎 祐子 訳: “The Java 仮想マシン仕様”，バリエ社 (1997).

[3] L. Sha, R. Rajkumar, and J. P. Lehoczky: “Priority inheritance protocols: An approach to real-time synchronization”, IEEE Transaction on Computers, Vol. 39, No. 9(1990).