4H-02　ユーザレベル・リアルタイムスレッドの実装と Java 仮想マシンを用いた評価

黒岩 実　高野 陽介
NEC　C&C メディア研究所

1　はじめに

組込みシステム分野では多種類の OS が使われているが、アプリケーションのリアルタイム性が OS に依存するのは移植性の観点からは問題である。我々は、様々な OS 上で互換性のあるリアルタイム性を提供するために、リアルタイムスレッドをユーザレベルで実現した (以下、PRT-Thread)。本稿では、PRT-Thread の設計と実装、および、組込み向き Java 仮想マシンに組み込んでの評価結果について述べる。

2　ユーザレベルスレッドのメリット

リアルタイムスレッドをユーザレベルで実現することの第一のメリットは、様々な OS 上で同レベルのリアルタイム性を提供できることである。組込みシステムでは様々な OS が採用されているが、それらのカーネルスレッドが提供するリアルタイム性は大きく異なっている。最近では汎用 OS が組込みシステムに採用される例も多いが、そのような汎用 OS には、たとえば固定優先度のスレッドを提供するものの優先度継承機能をもたない場合もあり、リアルタイム OS 上で動作していたプログラムをそのまま移植しても期待したリアルタイム性が実現できない。その点、ユーザレベル・リアルタイムスレッドでは、OS が提供するスレッド仕様への依存度が少なく、移植性が向上する。

第二のメリットは、スレッド機能のカスタマイズ性が高いことである。組込みシステムでは対象システムごとの細かい最適化が求められることが多いが、カーネルレベルスレッドではカスタマイズ項目が制限される。例えば、スケジューラとして固定優先度方式しか提供しない OS では、Earliest Deadline First 方式を実現するのは困難である。ユーザレベルスレッドでは、必要な機能を OS が提供していない場合でも機能拡張が容易である。

3　PRT-Thread の設計と実装

PRT-Thread は、スレッドの実行中断・再開などの実行制御や排他制御、同期といった基本機能に加えて、リアルタイム処理に関連する以下の機能を提供しており、ベース OS にもよるが、ミリ秒精度でのソフトリアルタイム性を実現する。

- 周期スレッド
  通常のスレッドに加えて、一定周期で起動される周期スレッドを提供する。周期スレッドは、時間属性として開始時刻、周期、デッドラインをもっており、それらの値はスレッド生成時に指定するほか、起動後も変更可能である。
- リアルタイム・スケジューラ
  全てのスレッドはプリエンプティブな固定優先度方式でスケジューリングされるが、固定優先度が等しいスレッド間のスケジューリングは、FIFO、ラウンドロビン、Rate Monotonic、Earliest Deadline First のいずれかの方式で行うことができる。
- デッドラインオーバーランの検出
  時間属性としてデッドラインをもつスレッドのデッドラインオーバーランを他のスレッドに通知することができる。この機能を使用して、過負荷時にスレッド周期を増加するなどの QOS 制御が可能となる。
- 優先度継承
  Mutex によるロック待ちで優先度継承を行うことによって、リアルタイム性を損ねる要因となる優先度逆転を低減することができる。

PRT-Thread を実現するために必要なベース OS の機能は、インターバル・タイマとカーネルスレッドコンテキスト情報の取得・設定機能である。これらの機能は小型のリアルタイム OS や汎用 OS の大部分で提供されているため移植は容易である。

現在、FreeBSD/Pentium、Linux/Pentium、および、LynxOS/PowerPC603e 上で PRT-Thread を実装している。これらの OS は POSIX 準拠であり、インターバルタイマを用いて一定間隔で SIGALRM シグナルを発生させることによって時間情報を管理している。そのため、時間精度は OS のインターバルタイマの時間精度に依存しており、10 ミリ秒程度である。また、OS 依存部と非依存部は分離して実装しているため、POSIX に準拠しない OS への移植も容易である。

4　組込み向き Java 仮想マシンへの適用

プロセッサや OS に依存しない移植性の高いプログラムを開発できるため、組込みシステム分野でも Java の利用が注目されており、組込みシステム用にリアルタイム性を向上した Java 仮想マシンが提供されつつある [1][2][3]。しかし、これらの多くは Java スレッドをリアルタイム OS が提供するカーネルスレッドで実現してい

Implementation and Evaluation of User-level Real-time Thread
Minoru KUROIWA and Yosuke TAKANO
C&C Media Research Laboratories, NEC Corp.

るため、第2節で述べたようにJavaプログラムのリアルタイム性がOSに依存することになり、移植性に問題が生じる。そこで我々は、リアルタイム性を必要とするJavaプログラムの移植性を向上するために、PRT-ThreadによってJavaスレッドを実現するJava仮想マシンを試作した。その特徴を以下に列挙する。

- **事前コンパイル方式の採用**
  組込みシステムではリアルタイム性とともに処理速度も重要な条件であることが多いため、リアルタイム性と処理速度を両立可能な事前コンパイル方式を採用した。これは、Javaバイトコードを Cプログラムに変換してからCコンパイラによってネイティブコードを生成しておく方式であり、Java-to-Cトランスレータとしてアリゾナ大学で開発されたToba[4]を利用している。
- **ミリ秒レベルのリアルタイム性の実現**
  JavaスレッドをPRT-Threadで実装するため、固定優先度方式スケジューラや優先度継承が提供される。加えて、Tobaに含まれるインクリメンタルGCを利用することで、様々なOS上でミリ秒レベルのリアルタイム性を実現できる。
- **周期スレッドクラスの提供**
  Javaの標準APIクラスでは、組込みシステムで頻繁に使用される周期スレッドが提供されていない。そこで、PRT-Threadの周期スレッドやリアルタイムスケジューラ、デッドラインオーバーラン検出などを利用可能な周期スレッドクラスを提供する。これはThreadクラスを継承しており、基本的なスレッド操作はThreadクラスと同様である。

5 Javaプログラムによる評価

PRT-Threadによって、十分なリアルタイム機能を提供していない汎用OS上でもミリ秒レベルのリアルタイム性を実現できることを確認するために、前節で述べた組込み向きJava仮想マシン上を用いて、Javaプログラムのスレッド間同期の待ち時間を測定した。リアルタイム・アプリケーションでは、スレッド間同期における待ち時間が設計時に見積もり可能である必要がある。

実験はFreeBSD2.2.8/Intel PentiumIII(450MHz)上で行い、待ち時間はSystem.currentTimeMillis()で測定した。また、他のプロセスやネットワーク処理の影響は排除して実験した。

図1に評価実験におけるスレッド構成を示す。高優先度(100)で周期200ミリ秒の周期スレッドAと、低優先度(10)で周期195ミリ秒の周期スレッドBが、各周期において共有オブジェクトSを20ミリ秒間ロックする。また中間優先度(99-n)の周期スレッドCnは各周期において20ミリ秒のビジーウェイトを行う。実験では、スレッドCnの個数を変化させながらスレッドAのロック待ち時間の最悪値を測定した。なお、測定は1000周期(200秒間)行った。

実験結果を図2に示す。優先度継承を行わない場合に

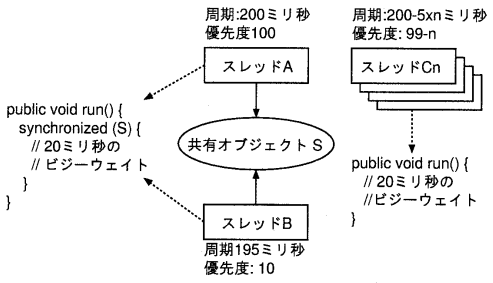

図1: 評価実験のスレッド構成

は、スレッドCnの個数が増加するに従ってロック待ち時間の最悪値が増大している。このロック待ち時間はスレッドCnの処理時間やタイミングに依存するため、事前に見積もることは困難である。それに対して、優先度継承機能を行うPRT-Threadによる実装では、スレッドCnの個数が増加してもロック待ちの最悪時間は20ミリ秒程度に抑えられている。この値は、オブジェクトSを共有しているスレッドBのロック時間であり、事前に見積もることは容易である。

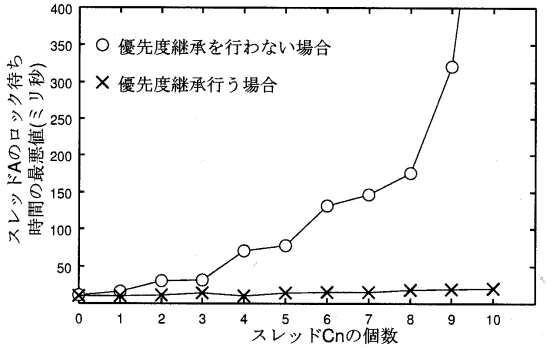

図2: スレッドAのロック待ち時間の最悪値

6 おわりに

本稿では、組込みシステムで用いられる様々なOS上で互換性のあるリアルタイム性を提供するPRT-Threadと、PRT-Threadを用いた組込み向きJava仮想マシンについて述べ、Javaプログラムによってリアルタイム機能を持たないFreeBSD上でもミリ秒レベルのリアルタイム性を実現できることを示した。

より多くの組込みシステムに適用するためには、アドミッション制御機能の実現や、ハードリアルタイムを提供するカーネルスレッドとの協調などが課題である。

参考文献

[1] Kelvin Nilsen: "Java for Real-Time", Real-Time Systems Journal, Vol.11, No.2 (1996).

[2] "Chai", http://www.chai.hp.com/

[3] "The Jeode Platform", http://www.insignia.com/

[4] "Toba: A Java-to-C Translator", http://www.cs.arizona.edu/sumatra/toba