3Z−3

MKngプロジェクトにおけるリアルタイム技術の応用: リアルタイム環境下でのJavaの使用について†

三好 昭彦　　喜多山 卓郎　　徳田 英幸
慶應義塾大学 SFC 研究所　慶應義塾大学 SFC 研究所　慶應義塾大学 環境情報学部

1 はじめに

慶応大学を中心に行なわれている次世代マイクロカーネル（MKng）研究プロジェクトでは，RT-Machマイクロカーネルをベースに各種処理のための拡張を行っている [10]．このプロジェクトの一貫として実時間制御が可能なJava仮想マシンの実装を行なっている [5]．

2 Javaの特徴と問題点

Java[2] はSun Microsystemsにより開発されたC++に類似しているオブジェクト指向の言語である．分散環境下での利用を意識しアーキテクチャーに依存しないバイトコードを使用し，セキュリティの機能を持つなどが特徴としてあげられる．

これらの特徴からJavaはWWW上のプログラミング言語としてだけではなく，新しい分野のプラットフォームとしても注目を集めている．例えばSmartCard，組み込みシステムや情報家電などである．JavaOS, Network ComputerやJava Cardなどの商用の製品もこのトレンドの一例としてあげられるだろう．

従来のプログラミング言語においては，そのターゲットに応じたライブラリなどにリンクし再コンパイルをしなければならなかった．それに対してJavaはアーキテクチャに依存しないためプログラマはターゲットのアーキテクチャを事前に知る必要がないなどの利点がある．このため，分散環境下での様々な利用がされやすい．

また組み込みシステムや，機械制御などの分野では定期的もしくはランダムにおこる外部イベントを効率良く処理しなければならないため，スレッドなどの機能が必要だが，CやC++などの従来の言語では各プラットフォームごとに対応したスレッドパッケージを使用しなければならないという問題があった．Javaでは統一したスレッドのインタフェイスがあるためプログラムの移植性が高くなる．

このような利点があるが，これらの分野で使用するには考慮されなければならない問題もある．機械の制御には指定された時間制限や必要な資源が必要な時に保証されなければならないという条件がある．現在のJavaの実行環境および言語仕様ではこのような条件を表現したり，実行時に満たすことが難しい．

現在我々はこのようなリアルタイム処理における問題点を解決すべくリアルタイムJava環境を構築している．本稿では我々のJava環境の設計と実装について説明する．

3 システム構造

我々のリアルタイムJavaの環境はRT-Machマイクロカーネル [9] 上のユーザレベルサーバとして実現されている．RT-Machは米国カーネギーメロン大学にて開発され，Mach3.0マイクロカーネルを拡張したものである．分散環境下でのリアルタイム処理をサポートし，Pentium, SPARC, MIPS, Power PCなどの幅広いアーキテクチャに移植されている．RT-Machにて提供されているリアルタイムの機能としてリアルタイムスレッド，リアルタイムの排他制御機能 [8]，リアルタイムIPC[3]，プロセッサ予約機能 [4] などがある．

我々のJavaサーバはRTS(Real-Time Server)[6] という同じくRT-Mach上のユーザレベルサーバを基に実装された．RTSはオブジェクトベースのサーバでタスク管理，ファイル管理やネームサービスを行なう機能を持っている．JavaサーバはRTSを拡張しネイティブコードだけではなくJavaのバイトコードをも実行できるようにしたものである．またインメモリファイルシステムを持ち，フロッピー，ハードディスクやRAMディスクなど様々なファイルシステムをマウントすることが可能である．ファイルはサーバのインメモリファイルシステムに連続したメモリ領域としてコピーすることも可能である．現在の実装では，Javaサーバは初期化時にローカルにあるハードディスク上のUnixファイルシステムをマウントし必要なクラスファイルなどをインメモリのファイルシステムにコピーを行なう．このことによりJavaのプログラムを実行中にディスクI/O処理でブロックしてしまうことを防いでいる．しかし，十分なメモリを持たないデバイスの場合はメモリ内に必要なファイルを保持するよりもマウントされたファイルシステムから直接読むことも可能である．Javaのバイトコードを解釈し実行するイ

Application of Real-time Technology in the MKng Project: Using Java for Real-time

Akihiko MIYOSHI[1], Takuro KITAYAMA[1], and Hideyuki TOKUDA[2]
[1]Keio Resreach Institute at SFC, Keio University
5322 Endoh Fujisawa Kanagawa 252, Japan
E-Mail:{miyos,takuro}@sfc.keio.ac.jp
[2]Faculty of Environmental Information, Keio University
E-Mail:hxt@sfc.keio.ac.jp

†この研究は，情報処理振興事業協会（IPA）が実施している創造的ソフトウェア育成事業「次世代マイクロカーネル研究プロジェクト」のもとに行われた．

ンタプリタとして我々はkaffe[1]という仮想マシンを使用した．

4　リアルタイムJavaスレッド

RT-Machにて提供されているリアルタイムスレッドを基にリアルタイムJavaスレッドを実装した．カーネルで提供されているRT-Threadはカーネルのスケジューラでスケジュールされ短い間隔でインタラプトをかけるカーネル内のクロックデバイスを使用し時間制御を行なっている．

プログラマは時間の属性をスレッドに対して新たに指定することによりリアルタイムスレッドを作成する．リアルタイムJavaスレッドに指定できる時間属性は開始時刻，周期，デッドラインである．開始時刻はスレッドが起動される時間を指定できる．同時にいくつかのスレッドを起動したい時などに利用できる．周期はスレッドが再び起動されるまでの時間である．資源が充分でなかったなどの理由でリアルタイムJavaスレッドがデッドラインに指定された時間内に処理を終えられなかった場合にその通知をプログラムに送るようにすることも可能である．

Javaサーバ上でユーザがリアルタイムJavaスレッドを新たに作成すると，仮想マシン内ではRT-Threadを作成しJavaスレッドをこれにマッピングする．つまりJavaのスレッドとRT-Threadの間には一対一の関係がある．また，予測できないプライオリティインバージョン[7]の問題を避けるためにスレッド同士の同期を行なう場合，仮想マシン内で`rt_mutex_lock`と`rt_mutex_unlock`というプリミティブを使用している．

5　これからの課題

より予測性の高いリアルタイムJava環境を実現するためには，スレッド以外にも様々な拡張を行なわなければならない．例えば，リアルタイムの処理では事前に使用する資源を必要な時に保証しなければならないという場合もある．このとき，現在のJavaではCPU資源，ネットワークやI/Oの帯域などの資源を表現するということは難しい．資源の保証や予約を行なうためには現在のガーベージコレクションのメカニズムなどの改良が必要である．

またGUIの部分でもさまざまな問題がある．Javaの環境で使用されている多くのウィンドウシステムでは優先度逆転が起こってしまう．例えばX11の場合，あるウィンドウをドラッグしている間は，他のウィンドウの表示が止まってしまう．レーダーからの情報をモニタしている場合を考えると，モニタ情報を表示するウィンドウがたとえ高いプライオリティを持つリアルタイム処理であっても，低いプライオリティの活動のウィンドウをドラッグしている間はウィンドウへの情報の更新ができないのである．現在我々はこれらの問題を解決するための作業を行なっている．

6　おわりに

本稿では組み込みシステム，機械制御などのJavaが従来使用されなかった分野でJavaを使用する利点を述べた．しかしこのような分野で利用するためには時間制約などリアルタイムの機能を言語仕様および実行環境で持たなければならない．

我々はこの問題を解決するため，リアルタイムのJava環境をRT-Machマイクロカーネル上のJavaサーバとして実装している．Javaサーバの実装の説明をした後，Javaスレッドの拡張であるリアルタイムJavaスレッドについて述べた．リアルタイムJavaスレッドはJavaスレッドに新たに開始時刻，周期，デッドラインなどの時間属性を与えることにより作られる．

また，より予測性の高いシステムにするにはスレッドだけではなくガーベージコレクションや描画などの処理系，資源を予約または保証するために言語での考慮が必要である．現在我々はこれらの課題を解決するための作業を行なっている．

謝辞

本研究を進めるにあたり，MKngプロジェクトの皆様から多大な助言をいただきました．ここに感謝の意を表します．

参考文献

[1] T. J. Wilkinson & Associates. Kaffe A free virtual machine to run Java code, 1997. http://www.kaffe.org.

[2] J. Gosling, B. Joy, and G. Steele. *The Java Language Specification*. Addison Wesley, 1996.

[3] T. Kitayama, T. Nakajima, and H. Tokuda. RT-IPC: An IPC extension for Real-Time Mach. In *Proceedings of the USENIX Symposium on Microkernel and Other Kernel Architecture*, September 1993.

[4] C.W. Mercer, S. Savage, and H. Tokuda. Processor Capacity Reserves for Multimedia Operating Systems. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, May 1994.

[5] A. Miyoshi and H. Tokuda. Real-Time Java Server for Real-Time Mach. In *Fifth International Workshop on Parallel and Distributed Real-Time Systems*, April 1997.

[6] T. Nakajima, T. Kitayama, and H. Tokuda. Experiments with Real-Time Servers in Real-Time Mach. In *Proceedings of USENIX 3rd Mach Symposium*, 1993.

[7] L. Sha, R. Rajkumar, and J. P.Lehoczky. *Priority inheritance protocols: An approach to real-time synchronization*. 1987.

[8] H. Tokuda and T. Nakajima. Evaluation of Real-Time Synchronization in Real-Time Mach. In *Proceedings of USENIX 2nd Mach Symposium*, October 1991.

[9] H. Tokuda, T. Nakajima, and P. Rao. Real-Time Mach: Towards a Predictable Real-Time System. In *Proceedings of USENIX Mach Workshop*, October 1990.

[10] 徳田, 追川, 西尾, 萩野, 斎藤. MKng: 次世代マイクロカーネル研究プロジェクト. 第*53*回情報処理全国大会論文集, September 1996.