3Z－5

MKngプロジェクトにおけるマイクロカーネル移植技法: マルチプラットフォームへの対応†

緒方 正暢[1]　光澤 敦[2]　斉藤 鉄也[3]　舩渡 大地[3]　徳田 英幸[3]

[1]日本IBM(株) ES事業　[2]NTT 情報通信研究所　[3]慶應義塾大学

1 はじめに

モーバイル，実時間，超分散/並列アプリケーションのような，高度なアプリケーションを構築していくには，様々なコンピュータアーキテクチャに対応でき，かつ，アプリケーションに対してハードウェアもソフトウェアも効率良く最適化されたコンピューティング環境を迅速に実現できなければならない．

慶應義塾大学が中心になって研究開発しているMKngプロジェクト [1] では，次世代のオペレーティングシステム (OS) に要求される動的構成/再構成法 [2] や，新しいシステムソフトウェアの動的適応可能型アーキテクチャ[3] に関する研究開発を行なっている．図1にソフトウェア構成の概要を示す．現在，Real-Time Machマイクロカーネル，および，各種サーバをSPARC[4], MIPS[5], PowerPC[6], Pentiumプロセッサを搭載した各種コンピュータ上に実装している．本稿では，マルチプラットフォームへの対応について具体的に検討する．

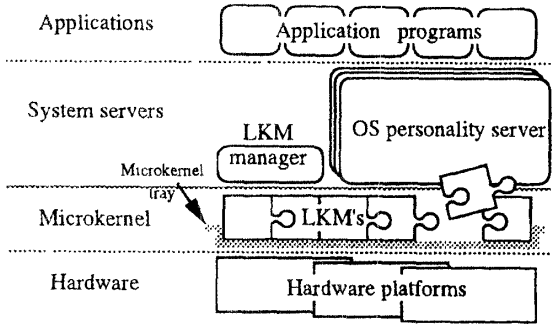

図 1: MKngにおけるソフトウェア構成

2 マイクロカーネルの課題

マイクロカーネル技術は，従来のUNIXでは，対応が困難であった新しいハードウェアや新しいアプリケーションの要求にこたえるための新しいOSを作る研究の成果として生まれた．カーネルとは，直接ハードウェアを管理，制御し，全てのソフトウェアの要求にこたえる共通部分を指す．マイクロカーネルでは，カーネル内に仮想記憶，プロセス間通信，タスク/スレッド，デバイスドライバといった基本機能のみを残し最小化した．それ以外の部分は，カーネルの外部にサーバとして実装した．サーバは，ユーザモードで走行し，他のアプリケーションと同様に扱われる．サーバは独自のアドレス空間を持ち，保護される．この方法は，UNIXとの互換性を保ちつつ，新しいOSの開発を可能にしていたためOS研究者に広く受け入れられたが，いくつかの欠点を持っていた [7]．

2.1 機能面

例えば，携帯端末では，2次記憶を仮定しない実メモリで動作することが要求される [8]．また，アプリケーションを限定して，サーバやOSの提供する機能を限定し，限られた資源を有効に活用したい．このように，特定の用途に適したOSを自由に組み立てることができなければならない．

固定された構造のマイクロカーネルでは，モーバイル環境のように計算機自身の構成を含め動的に変化するコンピューティング環境には適応できない．マイクロカーネルが備える基本機能そのものが動的に再構成できなければならない．

2.2 性能面

マイクロカーネル方式のカーネルは，サーバをユーザモードの独立したタスクとして実現しているため潜在的な性能の問題を持っている．それは，ユーザとカーネルモードの切替えが多い，アドレス空間の切替えが多い，プロセス間通信によってキャッシュミスが増加することなどである．このため，単層構造のカーネルに比べてマイクロカーネル構成のOSの性能が低いことが報告されている [13]．特にプロセス間通信の性能がボトルネックとなることがわかった [14]．しかし，プロセス間通信のオーバヘッドを軽減するように，アルゴリズム，データ構造といったカーネルの基本構造そのものを見直すことで性能を向上させた実装例も報告されている [15]．

マイクロカーネル構造のOSを移植する上では，単に論理的に動作するだけでは不十分で，アーキテクチャに応じた最適化を行ない性能的に満足するような方法を十分に検討しなければならない．

3 各種ソフトウェア環境への適用

3.1 UNIX 互換環境

RT-Mach上では，4.4BSD Lite Server(以下，Litesと呼ぶ)がUNIX環境を提供する．この場合，Litesが提供するUNIX環境が複数あっても構わない．すなわち，Pentium環境では，LitesがFreeBSDとLinuxのそれぞれのシステムについて，バイナリ互換性を提供することが可能である．また，X-Windowsを稼働させることもできる．この環境では，RT-Machの機能を使って書かれたアプリケーションが動作する．もちろん，従来のアプリケーションもそのまま動作する．動的なPCカード環境，モーバイル環境も提供される．

3.2 リアルタイムサーバ環境

RT-Machの実時間機能を使って実装されたリアルタイムサーバとしてReal-Time Server(RTS)とNetwork Protocol Server(NPS)がある．現在，この上にJava VMを実装中である [9]．

3.3 Java OS 環境

Java VMをOSパーソナリティとして実現し，Java OSに特化したカーネルの実現を行なっている [10]．

Microkernel Porting Technology in the MKng Project: Porting RT-Mach Microkernel and OS Personality Servers to Multiple Computer Platforms

Masanobu Ogata[1], Atsushi Mitsuzawa[2], Tetsuya Saito[3], Daichi Funato[3] and Hideyuki Tokuda[3]

[1]Embedded Systems Business Unit, IBM Japan, Ltd.
1623-14, Shimotsuruma, Yamato, Kanagawa 242, Japan
E-Mail: <ogata@yamato.ibm.co.jp>

[2]NTT, [3]Keio University

†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．

4　各種アーキテクチャへの適用

4.1　プロセッサ依存，独立部分の境界

Machでは，アーキテクチャ依存，非依存を区別して実装され，ソースコードがきちんとわかれており，他のアーキテクチャへの移植性が高いのが特徴であった．しかし，性能の向上のため，アルゴリズム，データ構造といったカーネルの基本構造そのものをアーキテクチャに応じた最適化を行なった場合，他のシステムやプロセッサへのカーネルの移植性が低くなってしまう．メカニズムに相当するところはアーキテクチャ依存でもいいが，ポリシに関するところはアーキテクチャに依存しないように実装しなければならない．

近年の高速プロセッサを搭載しているシステムでは，キャッシュの性能への影響は非常に大きい．オペレーティングシステムが，メモリ構成やキャッシュの詳細を理解し，うまく対応できれば性能向上が期待できる．例えば，物理アドレスキャッシュを使う場合と仮想アドレスキャッシュを使う場合ではカーネルの実装方法が異なってくる．

4.2　プロセッサ依存部分の構成

最近の高性能マイクロプロセッサは，プログラマから見えるアーキテクチャ(命令セットアーキテクチャ) を変更することなく，プロセッサを実現するアーキテクチャ（マイクロアーキテクチャ）をより高度なものにして性能向上を達成している．このようなプロセッサでは，カーネルモードの命令はプロセッサ毎にかなりの違いがある．プロセッサ依存部分はさらにもう少し細かく分類すべきである．

4.3　64ビットプロセッサのサポート

従来，Machマイクロカーネルでサポートされてきた Alpha は 64 ビットのみ，i386 は 32 ビットのみだった．本プロジェクトで実装中の MIPS や PowerPC には，32 ビットと 64 ビットの両方のモードで動作させることができるプロセッサがある．このため，プロセッサ依存部分のコード，特に，アセンブラで記述する部分に，32 ビットと 64 ビットのコードが混在するので実装をよく検討しなければならない．一方，C 言語で書かれた部分に関しては，64 ビットモードにおける C 言語の規定を使って記述する．

4.4　バイト順序

本プロジェクトで実装中のプロセッサでサポートしているバイト順序は，ビッグエンディアンとリトルエンディアンの両方がある．また，エンディアンを切替えることのできるプロセッサもある．

Machカーネルのプロセッサ独立部分は，どちらのエンディアンでコンパイルしても正しく動作する．一方，プロセッサ依存部分やデバイスドライバでは，実行時のエンディアンモードを考慮した実装が必要である．

4.5　資源管理と資源の仮想化

動的適応を可能にするためには，カーネルによる資源管理が動的にできなければならない．そのための，資源管理機能をカーネル内，あるいは，サーバとして実装しなければならない．また，マイクロカーネルでサポートされる資源を仮想化して上位のサーバやアプリケーションに提供するか，あるいは，そのまま仮想化せずに実現するか，検討しなければならない．仮想化は性能低下の原因になったり，利用する環境によって資源管理の方法が異なるからである．

4.6　デバイスドライバ

異なるチップを使用したデバイスドライバであっても，ほとんどの部分は共通で使えることが多いので，チップに依存する部分と依存しない部分を切りわけることができる．例えば，SCSI は，SCSI コントローラチップに依存した部分と，SCSI バスの制御のようにコントローラチップに依存しない部分がわかれている．移植する場合は，チップ依存部分のみをコーディングすればよい．Machでは，デバイスドライバをプラットフォーム独立に階層的に記述する枠組が既に提供されている [11]．この枠組により，MIPS, Alpha, i386, PowerPC, および SPARC でデバイスドライバの共有が可能になっている．

動的なデバイスドライバの置換，削除，追加を行なうためにはこの枠組は十分でないので，現在，デバイスドライバなどが動的にロード可能なカーネルを研究開発中である．

5　おわりに

本稿では，次世代マイクロカーネルプロジェクトにおけるマルチプラットフォーム対応について検討した．次世代 OS は，動的に構成/再構成な仕組みを備え，かつ，性能を低下させないことが必須である．

参考文献

[1] 徳田 他: “MKng: 次世代マイクロカーネル研究プロジェクトの概要,” 第 55 回情処全大論文集, 1Z-2 (1997).

[2] 盛合 他: “MKng プロジェクトにおけるカーネルアーキテクチャ: マイクロカーネルトレイとカーネルウェア,” 第 55 回情処全大論文集, 1Z-2 (1997).

[3] 追川 他: “スクリプト言語によるカーネルレベルでの動的適応,” マルチメディア，分散，協調とモーバイル (DiCoMo) ワークショップ, pp.509–514 (1997).

[4] 舩渡 他: “MKng プロジェクトにおけるマイクロカーネル移植技法: SPARC 用マイクロカーネルの実装と性能評価,” 第 55 回情処全大論文集, 3Z-6 (1997).

[5] 光澤 他: “MKng プロジェクトにおけるマイクロカーネル移植技法: MIPS 用マイクロカーネルおよび UNIX サーバの性能評価,” 第 55 回情処全大論文集, 3Z-7 (1997).

[6] 岩崎 他: “MKng プロジェクトにおけるマイクロカーネル移植技法: デバッグツールを利用した PowerPC への移植,” 第 55 回情処全大論文集, 3Z-8 (1997).

[7] Liedtke, J.: “Toward Real Microkernels,” CACM, Vol. 39, No. 9, pp. 70–77 (1996).

[8] 黒岩 他: “MKng プロジェクトにおける携帯端末サポート,” 第 53 回情処全大論文集, 5B-5, pp. 1-41–1-42 (1996).

[9] 三好 他: “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,” 第 55 回情処全大論文集, 3Z-3 (1997).

[10] 高野: “OS パーソナリティとしての JavaVM の利用,” 情処研報 97-OS-75-13, 73–78 (1997).

[11] Forin, A., Golub, D. and Bershad, B.: “An I/O System for Mach,” Proc. of the USENIX Mach Symp., (1991).

[12] Black, D. L. and Bernadat, P.: “Configurable Kernel Project Overview,” OSF Research Institute Collected Papers, Vol. 4, OSF (1994).

[13] Chen, J.B., et al.: “The impact of Operating System Structure on Memory System Performance,” In Proc. of the 14th ACM Symp. on Operating Syste Principles (SOSP-14), pp. 120–133 (1993).

[14] Condict, M., et al: “Microkernel Modularity with Integrated Kernel Performance,” OSF Research Institute Collected Papers, Vol. 4, OSF (1994).

[15] Liedtke, J.: “On Microkernel Construction,” In Proc. of the 15th ACM Symp. on Operating Syste Principles (SOSP-15), pp. 237–250 (1995).