汎用ホモジニアスマルチコアにおける異種 OS の連携機構

磯部 泰徳† 佐藤 未来子‡ 並木 美太郎§
東京農工大学大学院工学府情報工学専攻 †/ 東京農工大学大学院 ‡/ 東京農工大学大学院共生科学技術研究院 §

1 はじめに

近年，マルチコアプロセッサは携帯電話などの様々な機器に搭載されている．マルチコアプロセッサの性能を引き出すためには，専用 OS による適切なリソース管理やスケジューリング制御などを行い，コアを効率よく利用することが有効である．しかし，そのような専用 OS だけでは従来の汎用システムのファイルシステムや API などのような利便性を望むことは難しい．

従来の汎用 OS の利便性を維持したまま，マルチコアプロセッサを効率よく利用できるステムとするために，それぞれの処理内容に特化した複数のシステムソフトウェアを連携させる手法が有用である．本研究では，共有メモリ型ホモジニアスマルチコアプロセッサ（以下，マルチコアプロセッサ）を対象として，その性能を活かすためのプログラム実行基盤の実現を目指している．

2 本研究の目標

本研究では，汎用 OS とマルチスレッドプログラムを軽量に管理・制御する並列演算向け専用 OS（以下 専用 OS）を連携させて動作させるヘテロジニアスハイブリッド OS 構成にすることによって，汎用 OS と専用 OS を適材適所に利用し，汎用 OS の利便性と専用 OS の演算性能を両立させることを目標とする．

3 システム構成

本システムでは，マルチコアプロセッサ上の異なるコア上で，汎用 OS と専用 OS を同時に動作させ互いに通信を行うことによって，一つのシステムとして動作する．汎用 OS である Linux がファイルシステムや I/O，ネットワークなどの機能を提供し，専用 OS である Future[1] はマルチスレッドライブラリ MULiTh(Userlevel Thread Library for Multithreaded architecture) と連携して高速なマルチスレッドプログラムの実行環境を提供する．二つの OS 共有メモリを介した OS 間通信で互いに通信を行い連携させる．これによって，専用 OS を肥大化・複雑化させることなく，汎用 OS の機能を専用 OS 向けプログラム中から利用可能となる．

図 1 にシステムの動作概要を示す．

3.1 Linux

Future のブートや Future 用プログラムのロードなど，Future の制御を行う．また，Future 側で発生した I/O 処理も Linux が代行する．また，Future との通信はデバイスドライバを介して行う．

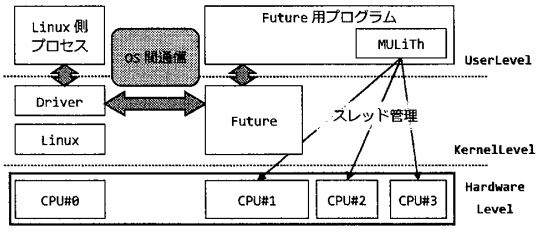

図 1: システムの概要

表 1: Linux と Future の主な役割の分担

Linux	Future/MULiTh
I/O の処理	スレッドプログラムの実行
Future の制御	複数の CPU コアの管理

3.2 Future

MULiTh のスレッド制御のサポート及び，Linux に対し OS 間通信を利用し高速なマルチスレッドプログラムの実行環境を提供するカーネルである．プロセス管理や例外・割込の処理，OS 間通信の送受信などを行う．

3.3 MULiTh

MULiTh は POSIX スレッドを管理するユーザーレベルのスレッドライブラリで，スレッドのコアへの割り当てや制御を行う．

4 連携機構

Linux と Future は互いに役割を分担し，連携することによって一つのシステムとして動作する．OS 間連携機構では，次の機能を提供する．

Linux からの Future の起動
あらかじめ起動した Linux から Future のカーネルをロードし，ブートする．

Future 用プログラムのロード及び実行
Linux のファイルシステム上で管理している Future 用プログラムを Linux からロードし，Future 上で実行させる．また，実行時にプログラムで利用する CPU コアの数を引数として指定できる．

Future 用プログラムのシステムコール代行処理
Future 用プログラム中で発行されたシステムコールを Linux に代行させることによって，Linux 側で管理している I/O などの資源を利用可能にする．

連携で用いる OS 間通信は CPU コア間の割込を用いて通知し，データの受け渡しは OS 間の共有メモリを利用する．

4.1 Future 用プログラムのロード及び制御

Future プログラムのプログラム制御ルーチンはプログラムのロード・実行依頼後は，終了までスリープして

Mechanism of Cooperation between Heterogeneous Operating Systems for General-purpose Multi-core Processor

† Hironori Isobe
Computer and Information Sciences, The Graduate School at Tokyo University of Agriculture and Technology
‡ Mikiko Sato
Graduate School of Engineering, Tokyo University of Agriculture and Technology
§ Mitaro Namiki
Institute of Symbiotic Science and Technology, The Graduate School at Tokyo University of Agriculture and Technology

待機する．システムコールの代行が必要な場合は，このプログラム制御ルーチンが依頼を受け取りシステムコールを代行する．Linux から Future 用プログラムを起動させるシーケンスを図 2 に示す．
(1) 起動した Linux から Future 用のメモリ領域にカーネルプログラムをロードする．
(2)OS 間通信で Future に対して実行依頼を行う．この際にロード先のアドレスや利用するコア数などを通知する．依頼を受けた Future は通知されたロード先アドレスから実行を開始する．
(3) システムコールが発行された場合は，Linux に対して代行依頼を行う．Linux は代行した結果を Future へ返す．
(4)Future のプログラムが終了したら Linux へ通知を行う．Linux は通知を受けると Future へプロセスの終了依頼を出す．

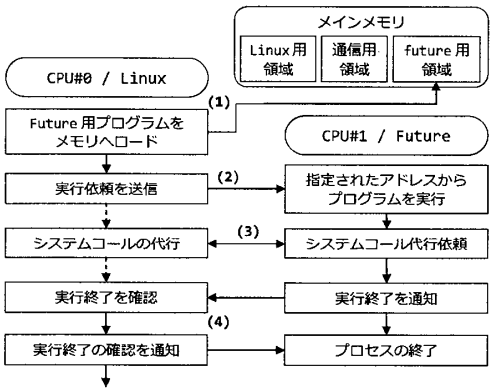

図 2: Future 用プログラムのロード及び実行

4.2 システムコールの代行処理

システムコール代行処理では Future プログラム内のファイル入出力に関するシステムコール (open,close,read,write など) とネットワークに関するシステムコール (socket,connect など) を Linux 側で代行する．現段階では，ファイルシステムに関するシステムコールの一部のみを実装している．

Future プログラム中のシステムコールの Linux での代行処理のシーケンスを図 3 に示す．
(1) 標準ライブラリ中のシステムコール呼び出しをフックし，Future のシステムコールを呼び出す．
(2)Future のシステムコールは OS 間通信を利用して，Linux 側のプログラムに対してシステムコール番号，引数を渡す．
(3)Linux 側のプログラムは OS 間通信を通して受け取った情報からシステムコールの種類を判別し，代わりにシステムコールを Linux へ発行する．

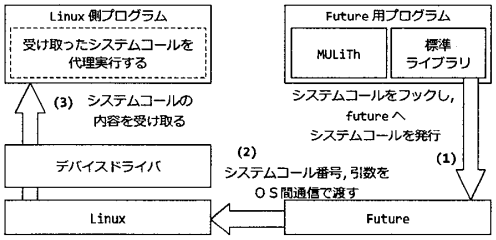

図 3: システムコールの代行処理

5 OS 間通信の基本性能

本システムを参考文献 [2] のマルチコアプロセッサ (表 2) へ実装し，Linux から Future 用プログラムを動作させ，終了するまでの所要時間を計測した．その結果を 3 に示す．また，参考データとして，SH-Linux(kernel-2.6.19) の fork によるプロセス生成の所要時間を併せて示す．

表 2: 情報家電用マルチコアプロセッサ仕様

項目	仕様
CPU コア	SH-4A(600MHz) × 4 core
システムバス周波数	300MHz

表 3: Future プロセス生成のオーバーヘッド

項目	所要時間
Future：ロード～プロセス生成～終了	11.53[msec]
SH-Linux：fork～execv～wait	6.37[msec]

現在の実装方法では，Future 用のプログラムをロードする際に Linux が管理するメモリ領域から Future 用のメモリ領域へプログラムをコピーする必要がある．そのため，このように差が開いてしまったと考えれる．今後はメモリコピーによるオーバーヘッドを極力減らすような実装方法を検討していく必要性があると考えられる．

6 おわりに

本稿では，汎用ホモジニアスマルチコアにおける異種 OS の連携機構について述べた．今後は，Future プログラムのロード方法の改善やシステムコールの代行処理の実装，Future のマルチプロセス化への対応，GUI の代行などの機能強化を図る予定である．

謝辞 本論文おけるマルチコアチップ及び実行環境は NEDO リアルタイム情報家電用マルチコアプロジェクトにて早稲田大学，(株) ルネサステクノロジ，(株) 日立製作所により開発されたものです．関係者の皆様に感謝申し上げます．

参考文献

[1] 佐藤未来子，磯部泰徳，十山圭介，野尻徹，入江直彦，内山邦夫，並木美太郎．汎用ホモジニアスマルチコアプロセッサにおける OS とスレッドライブラリ．情報処理学会第 20 回コンピュータシステムシンポジウム，ポスターセッション，(2008)
[2] 早瀬清ほか，独立に周波数制御可能な 4320MIPS、SMP/AMP 対応 4 プロセッサ LSI の開発，情報処理学会研究報告,2007-ARC-55,pp.31-35(2007)
[3] 下澤拓，藤田肇，石川裕．マルチコア SH における複数カーネルの実行機構の設計と実装．情報処理学会研究報告,2008-OS-109,pp.25-32(2008)
[4] 遠藤幸典，菅井尚人，山口義一，近藤弘郁．シングルチップマルチプロセッサ上のハイブリッド OS 環境の実現 - システムアーキテクチャ -．情報処理学会第 66 回全国大会 (2004)