Cell/B.E. への MINIX 3 の移植と ヘテロジニアスマルチコアプロセッサ上でのプロセス管理方式の実現

野尻祐亮† 並木美太郎†
†東京農工大学

1 背景と関連研究

近年のプロセッサ開発の潮流はマルチコアへと向かっているが，その流れのなかでも特にヘテロジニアス（非対称型）マルチコアという形態が注目されている．本研究で対象する Cell/B.E. は，システム全体の制御用の PPE (PowerPC Processor Element) と，ベクトル演算用の SPE (Synergistic Processor Element) の，2 種類のプロセッサコアを搭載している．

しかし，ヘテロジニアスマルチコアでは，ソフトウェアの設計・開発が複雑になるという問題点がある．プロセッサコアごとに異なる命令セットにしたがってプログラミングすることが容易でない．システムの制御，プロセス管理，リソース管理，デバイスドライバなどの OS の処理は，特に工夫しない限り 1 個のプロセッサコア（あるいは 1 種類のプロセッサコア）でしか実行されない．しかし，それらの処理は各プロセッサコアへ分散させることが望ましい．

報告 [1] では， Linux 上において，カーネルモードでの処理，特にデバイスドライバの処理をユーザモードに引き出した上で，SPE に分担させる仕組みを提案，実装している．それに対して本研究で提案する方式では， OS が直接 SPE を扱うようにすることにより，設計を単純にでき，[1] で課題であったモード遷移などのオーバヘッドを最低限にすることができる．

2 目標

ヘテロジニアスマルチコアでソフトウェアの設計・開発が複雑になるという問題を解決するため， Cell/B.E. において，すべてのプロセッサコアを統合的に管理する新たな OS を実現する．その OS において，単なる並列化とは異なる新規な点として，次の点を目標とする．

- **OS の機能の一部を， PPE だけでなく SPE などの別のプロセッサコアでも実行できるようにする．**
  特定の機能については， SPE など別のプロセッサコアで実行した方が高速であるかもしれない．また，遊休しているプロセッサコアの活用により，全体としての性能向上につながる．

Porting MINIX 3 to Cell/B.E. and Implementation of Process Management on Heterogeneous Multicore Processors
Yusuke NOJIRI†, Mitaro NAMIKI†
†Tokyo University of Agriculture and Technology

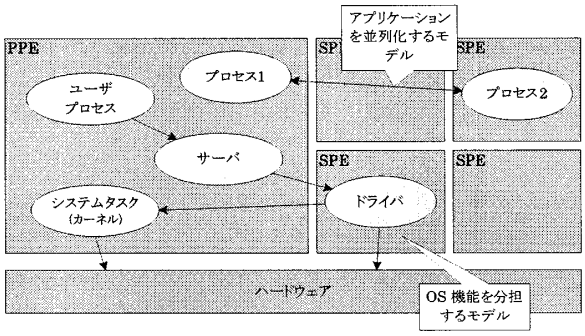

図 1: PS3 MINIX の OS アーキテクチャ

- **上位層で実行するプログラムのために，統一したプロセスモデルを提供する．**
  デバイスドライバやアプリケーションを，異なる命令セットを持つ別のプロセッサコアに移植することが容易になる．ヘテロジニアスマルチコアプロセッサを利用するにあたっての敷居を下げることができる．

並列化すべき OS の機能としては，例えばデバイスドライバが挙げられる．画面表示のためのフレームバッファドライバ，あるいはストレージやネットワーク入出力におけるデータ暗号化，圧縮のフィルタドライバは処理量が大きく，並列化することが望ましい．

3 設計

本研究で開発する OS は，マイクロカーネル構造を持つ **MINIX** をベースとする．マイクロカーネルでは， OS の機能が小さなモジュールとして独立しており，OS の機能を各プロセッサコアに割り当てて，並列に実行させるモデルが成立すると考えた．

マルチコアで OS 処理を行う OS アーキテクチャを図 1 に示す．主に二つの処理モデルを想定している．

一つは， OS 機能を各プロセッサコアで分担するモデルである．例えばデバイスドライバを SPE プロセスとして作成し実行する．OS 機能を担う一部のプロセスを SPE 用に作成すれば，プロセッサコア同士の連携は， OS のプロセス管理にのっとって行われるので，それで目標である OS 機能の各プロセッサコアへの分散が達成できる．

もう一つは，アプリケーションをマルチプロセスで作成し，並列実行するモデルである．

プロセスモデルとしては，SPE において 256KB より大きなローカルメモリを利用できない，ローカルメモリ以外のメモリ（他プロセス，I/O）に直接アクセスできないなどの制約を除けば，PPE, SPE の両プロセッサコアで同等のモデルを実現できる．異なるプロセッサコア間でも透過的にプロセス間通信を行える機構をカーネルに設け，C 標準ライブラリやシステムコールなどの機能を PPE, SPE どちらにおいても，同じように利用できるようにする．MINIX では，次に挙げる，特権が必要な操作は，カーネルが代行する設計となっている．

- 仮想アドレス/物理アドレスによるメモリコピー
- 仮想アドレスから物理アドレスへの変換
- 割込みの許可/禁止（割込みはメッセージで受け取る）
- メモリセグメントの設定（I/O などへのアクセスのため）
- プロセスの生成/実行/破棄，シグナルの処理
- 時刻，システム情報の取得/設定

SPE においても，それらのカーネルコールが利用できることで，OS 機能を果たすプログラムを SPE で動作させることが可能となる．

4 実装

目標の OS の実現を，大きく二つの段階に分けて述べる．

4.1 MINIX 3 の移植

まず，MINIX 3 を Cell/B.E. プラットフォームの PLAYSTATION 3 に移植し，PPE だけを使って動作させられるようにする．本研究で移植する MINIX は，オリジナルの x86 版ではなく，PowerPC 版の **MinixPPC**[2] である．MinixPPC に対して実装したものは次の通りである．

- ブートストラッピング部分
- MMU 制御，時計の実装
- 画面表示，USB ホスト/キーボード，内蔵ハードディスクの各種ドライバの実装
- デバイス管理機構の設計・実装

4.2 ヘテロジニアスマルチコアへの対応

そして，MINIX カーネルに手を加え，プロセスを SPE で動作させるための機構を設ける．

- カーネルによるマルチコア管理
  各プロセッサコアにおけるプロセスディスパッチ，スケジューリング，プロセッサコア動作の制御など．
- メッセージ転送機構，メモリ転送機構
  PPE–SPE 間の通信は，メモリマップ I/O で行う．

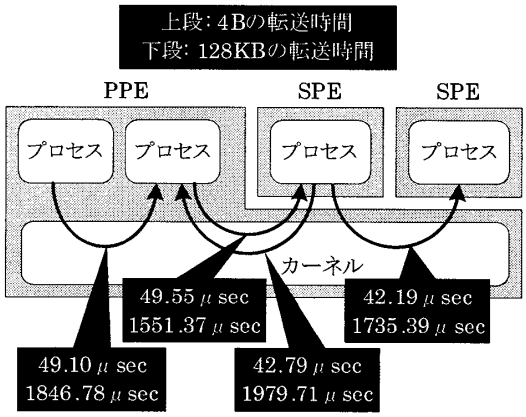

図 2: メモリ転送にかかった時間

実装の結果，SPE 上のプロセスから `printf` などのライブラリ関数や，`open`, `write` など，多くの POSIX システムコール，MINIX カーネルコールが利用できるようになった．これにより，PPE 用のプログラムとほぼ同様に SPE 用のプログラムを記述することができる．

5 評価と考察

本研究において実装した，プロセッサコアをまたぐメモリ転送機構の性能を測定する．メモリ転送性能を表す値として，メモリ転送を行うカーネルコール `sys_vircopy` のターンアラウンドタイムを計測する．この性能は，マイクロカーネルにおいて特に重要な，プロセス間の連携（システムコールも含む）に影響する．

計測した結果を図 2 に示す．大まかに見れば，いずれの転送元・転送先においても，大差は見られない．ただし，SPE から PPE へのメモリ転送は，明らかに長時間かかる傾向がある．しかし，これはメモリコピーが非効率的な実装[1]となっているためで，本研究で提案した設計に由来する問題ではない．従来の MINIX を並列化し，各プロセッサコアに処理を割り当てて性能向上を狙うという観点からは，有望な結果が得られた．

今後は，SPE で実行できる処理の種類を増やし，目標である OS 処理の分散を達成できるようにする．

参考文献

[1] 町田 浩之. Cell broadband engine, SPE assisted user space device driver. *CE Linux Forum Japan Technical Jamboree 13*, 2007.

[2] Ingmar A. Alting. MinixPPC. Master Thesis Computer Science, 2006.

[1] PPE のメモリコピーにキャッシュ書出し処理が含まれている．また，メモリコピーの実装の最適化が不十分である．最適化により改善可能である．