マルチコアプロセッサにおける スレッドライブラリと OS の連携方式の設計

磯部 泰徳† 佐藤 未来子‡ 並木 美太郎§
東京農工大学工学部情報コミュニケーション工学科 †/ 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，シングルコアプロセッサの性能向上の頭打ちなどの理由から，マルチコアプロセッサが注目を浴びている．マルチコアプロセッサの性能を引き出すためには，マルチコアに対応した処理内容に適したシステムソフトウェアが必要になる．このシステムソフトウェアが従来の汎用 OS と互いに通信を行い，処理を分担しつつ，一つのシステムとして動作するように連携する．この連携によってマルチコアプロセッサの特徴をより生かし，性能を引き出すことができる．

本研究では，プロセッサ上の複数のコアを管理するための OS と汎用 OS の Linux を並列に動作させるヘテロジニアスなマルチ OS 構成のシステムによって，それぞれの OS で最適な処理を行うことで，全体としての実行性能の向上を図る．具体的には，Linux などの汎用 OS で I/O や資源管理を，並列動作させる OS でプロセッサ上の複数のコアを管理とマルチスレッドプログラムの実行環境の提供を行う．並列 OS で CPU コアの管理を行うことで CPU コアを無駄なく利用できるようにし，汎用 OS と並列 OS の 2 つの OS が連携して動作することで従来のシステムと親和性の高い実行環境を提供する．

2 本研究の目標

本研究では，複数のコアの管理を主目的とした並列 OS を，汎用 OS である Linux と連携させて動作させる方式を設計することを目標とする．並列 OS 上ではマルチコアプロセッサ上での利用に特化したマルチスレッドプログラム実行環境を提供する．Linux と並列 OS を異なるコアで同時に並列に実行して，システム全体の管理と複数のコアの管理を Linux と並列 OS が分担し，互いに通信を行うことで一つのシステムとしてそれぞれの OS が協調した動作を実現する．

3 本システムの概要

本システムでは，二種類の OS(Linux と並列 OS) を動作させ，それぞれの OS が互いに連携することでマルチスレッドのプログラムを高速に処理する．この連携によってユーザーは，従来の Linux を利用したままマルチコアプロセッサの性能を十分に活用することができる．メインの OS である Linux が I/O や資源管理などのシステム全体を管理し，並列 OS がマルチスレッドプログラムの実行環境を管理する．マルチスレッドプログラムの実行環境は POSIX スレッド API を持つスレッドライブラリによって実現する．複数ある CPU コアのうち 1 つを利用して Linux を動作させ，それ以外のコアを並列 OS で管理する．Linux と並列 OS では，独自の OS 間通信機構によって情報をやり取りし，連携動作を実現する．図 1 にシステムの動作概要を示す．

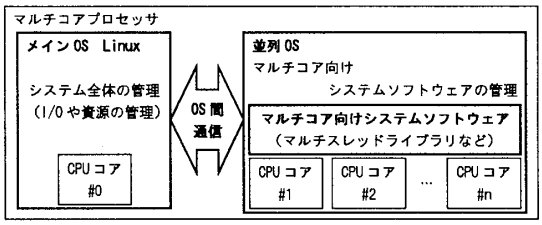

図 1: システムの動作概要

4 設計

4.1 連携方式の概要

メインメモリを Linux 用の領域と並列 OS 用の領域に分割し，それぞれの領域を Linux と並列 OS が独自に管理する．CPU コアのうちのひとつを使い Linux を動作させ，残りの CPU コアを並列 OS とスレッドライブラリで利用する．スレッドライブラリの OS との連携機能の実行モデルについて図 2 に示す．以下で，図 2 の内容を追いつつ，実行モデルについて述べる．

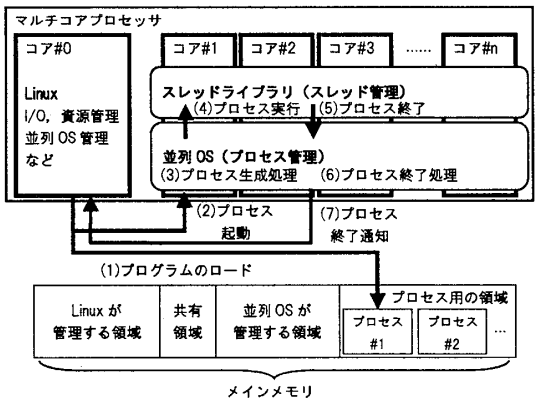

図 2: 実行モデル

(1) プログラムのロード 実行するプログラムを Linux から並列 OS のプロセス用メモリにロードする．

(2) プロセス起動 プログラムのロードが完了したら Linux は並列 OS に対し，ロードしたプログラムの実行を要求する．

(3) プロセス生成処理 並列 OS は Linux からプロセス起動通知を受け取ると，並列 OS 内部にプロセスを生成する．

(4) プロセス実行 プロセス生成終了後，並列 OS はスレッドライブラリにプロセス実行を指示する．ライブラリは指示を受け，プロセスを実行する．

(5) プロセス終了 スレッドライブラリがプロセス実行を終了し，並列 OS に対して終了通知を送信する．

Design of Execution Environment with Thread Library and Operating System for Multi-core Processor

† Hironori Isobe
Department of Computer, Information and Communication Sciences, 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

(6) プロセス終了処理 終了通知を受けた並列 OS はプロセスの終了処理を行う．実行結果は Linux と並列 OS で共有するメモリを利用してやり取りする．

(7) プロセス終了通知 並列 OS 内部でのプロセス終了処理が完了すると，並列 OS は Linux に対してプロセスが終了したことを通知する．

4.2 OS 間通信の設計

並列 OS で実行するプログラムに対応した Linux 側仮想プロセスを用意する．この Linux 側仮想プロセスによって，プログラムのロード，システムコールの依頼，シグナルの処理などを行う．Linux 側仮想プロセスに対し，並列 OS 側で実行するプログラムを並列 OS 実プロセスと呼ぶ．並列 OS 実プロセスを実行中に発生した I/O 処理などは Linux 側で動作するデーモンで処理する．並列 OS でプログラムを実行する過程を図 3，図 4 で示し，以下でその内容について述べる．

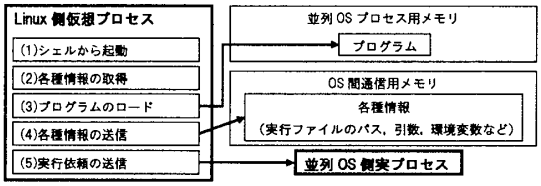

図 3: 並列 OS への実行依頼までの処理

(1)Linux 側の仮想プロセスをシェルから起動する この仮想プロセスは並列 OS 側実プロセスと 1 対 1 に対応し，プログラムのロード，実行依頼，シグナルの受信などを行う．

(2) 各種情報の取得 実行ファイルのパス，引数，環境変数などの実行に必要な情報を取得する．

(3) プログラムのロード Linux 側から並列 OS のプロセス用メモリにプログラムをロードする．

(4) 各種情報の送信 並列 OS でプログラムを実行するために必要な情報を送信する．具体的には，実行ファイルのパス，引数，環境変数，プログラムをロードしたアドレス，Linux 側仮想プロセスのプロセス ID などである．

(5) 実行依頼の送信 並列 OS へ実行を依頼するために必要な上記の処理が完了したら，Linux 側から並列 OS へ実行の依頼を行う．

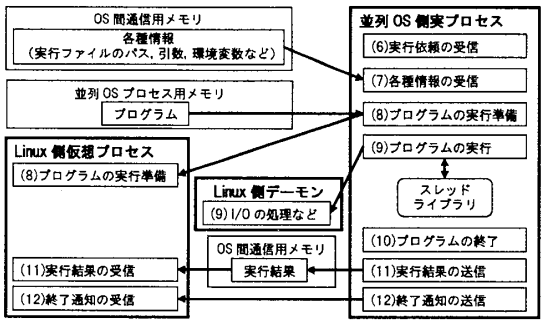

図 4: 実行依頼受信からから実行終了までの処理

(6) 実行依頼の受信 並列 OS は実行依頼を受信すると，依頼されたプログラムの実行準備を開始する．

(7) 各種情報の受信 (4) で Linux が送信してきた各種情報を受信する．

(8) プログラムの実行準備 プロセス空間の生成，ダイナミックリンクライブラリの mmap 処理などを行い，プログラムを実行できる環境を整える．必要なライブラリなどは Linux 側から受け取る．

(9) プログラムの実行 実行準備が整うと，並列 OS は管理している複数のコアを利用して，プログラムの実行処理を行う．スレッドライブラリは各コアにプログラム中のスレッドを割り当て，並列に処理する．実行中に生じた I/O 処理などは Linux 側でデーモンを動作させておき，そのデーモンが並列 OS から処理の依頼を受信して実行する．

(10) プログラムの終了 プログラムが exit システムコールを発行すると，並列 OS はプロセスの終了処理を行う．

(11) 実行結果の送信/受信 並列 OS は実行結果を Linux に対して送信する．実行結果のやり取りは OS 間共有メモリを利用する．

(12) 終了通知の送信/受信 並列 OS 側実プロセスが終了すると，並列 OS は Linux 側仮想プロセスに対して終了通知を行う．終了通知を受信すると，Linux は仮想プロセスを終了する．

Linux 側仮想プロセスと並列 OS 実プロセスは I/O やシステムコールの処理を分担している．分担の内容を以下の表 1 に示す．

表 1: Linux と並列 OS での処理の分担

Linux 仮想プロセス	並列 OS 実プロセス
I/O exit 以外のシステムコール シグナルの受信	プログラムの実行 exit システムコール シグナルの処理

並列 OS 実プロセスで発行されたシステムコールは，Linux 仮想プロセスに処理を依頼し，Linux が処理する．Linux 仮想プロセスに対するシグナルはすべて並列 OS に送信し，並列 OS の実プロセス側が処理する．

5 考察

本方式では，複数の CPU コアを管理するための専用の OS を用いることによって，柔軟かつ効率の良いコアの管理を可能にし，汎用 OS である Linux と連携させることによって，従来のシステムとの親和性を高くすることができる．これによって，汎用 OS の利点を生かしつつ，マルチコアプロセッサの性能を十分に引き出すことが可能である．

6 おわりに

本研究では，マルチコアプロセッサにおけるスレッドライブラリと OS を連携させる方式を設計した．まだ設計段階に過ぎず，実際の計算機で動作させるに至っていない．よって，この方式による Linux と並列 OS の連携をマルチコアプロセッサ上で試作し，性能を評価することが今後の課題である．

参考文献

[1] 清水正明，小笠原克久，舩生真紀子，米澤明憲．ヘテロジニアス向けリモートプロセス管理機能．コンピュータシステム・シンポジウム 2007,pp117-124 (2007)