マイクロカーネルLavenderにおけるUNIXサーバの構成

2F－4

外山 正勝 †　佐脇 秀登 †　芝 公仁 †　毛利 公一 †　大久保 英嗣 ††
† 立命館大学大学院理工学研究科　†† 立命館大学理工学部情報学科

1 はじめに

マイクロカーネルベースのオペレーティングシステム (OS) は, その柔軟性や拡張性が注目されている. 現在, 我々は, ユーザカスタマイズ可能なマイクロカーネル Lavender を構築している.

マイクロカーネルベースの環境では, OS として提供される機能は, 主にユーザプロセスであるシステムサーバにより実現される. Lavender のシステムサーバは, カーネルの機能を利用して OS の機能をカスタマイズし, それぞれのアプリケーションに固有の世界観を提供することができる. このような形で実現される OS 固有の特徴を, OS パーソナリティと呼ぶ. それぞれのパーソナリティは, 互いに独立した存在であり, マイクロカーネル上で複数のパーソナリティを混在させることが可能である. これにより幅広い適用性をもった OS を構築できる.

現在, いくつかのマイクロカーネル上で, UNIX パーソナリティが実現されている. これは, マイクロカーネルベース環境の研究, カーネルの性能および適用性の評価, 開発環境の整備, 実際の利用などを目的としている. Lavender では, さらに OS パーソナリティを実現する上で必要となるフレームワークの抽出, 複数のパーソナリティを混在するためのアーキテクチャの検証を目的とし, UNIX サーバの実装を進めている. 以下, 本稿では, 2 章で Lavender の特徴, 3 章で UNIX サーバの概要, 4 章でまとめを述べる.

2 Lavender の特徴

マイクロカーネルベースの OS を実現する上で, 頻発するプロセス間協調作業による性能の低下が問題視されている. Lavender では, 協調作業時のオーバヘッドを抑えるため, プロセスグループ機能とレジデントアドレス空間を用意している [1]. これらの機能により, クロスアドレススペースコールによるオーバヘッドが軽減される. さらに, プロセス間でデータや手続きを共有するモジュール機構により, システムサーバとクライアント間で処理コストの小さい協調作業が実現できる.

また, カーネル内部の機能を階層化し, 各層において粒度の異なる機能を提供するための階層化インタフェースを備えている. これにより, 処理の内容に合わせて適切な粒度のシステムインタフェースを選択することが可能となり, OS の機能の変更および拡張に柔軟に対応できる.

3 UNIX サーバの概要

3.1 構成方式

UNIX サーバの実装には, 2 つの方式がある. 単一のシステムサーバにより UNIX の機能を実現するシングルサーバ方式と, 機能単位で分割された複数のシステムサーバ群により UNIX の機能を実現するマルチサーバ方式である.

マイクロカーネル上に実装されている UNIX サーバの多くは, シングルサーバ方式を採用している [2] [3] [4]. この方式による UNIX サーバは, 従来のモノリシックカーネルをマイクロカーネル上に 1 つのプロセスとして移植したものである. このアプローチは, オリジナルのカーネルの変更が少なくて済み, 移植が比較的容易である. また, 多くの場合, 従来のアプリケーションをそのまま利用することができる.

しかし, シングルサーバ方式では, モノリシックカーネルが抱えている柔軟性の欠如といった問題点をそのまま継承してしまう. そこで, GNU Hurd[5] のようにマルチサーバ方式による UNIX の実装が試みられている. マルチサーバ方式では, OS の持つ機能を適度な粒度のシステムサーバに分割して実現する. このような機能分割により, 個々のシステムサーバの構造が単純になり, 開発時のデバッグが容易になる. また, 機能の変更および拡張が容易に実現できる.

Lavender は, ユーザカスタマイズ可能なシステムを目指しているため, ユーザの要求により必要とされる機能の変更および追加が容易に実現できるマルチサーバ方式が適していると考えた. したがって, Lavender における UNIX サーバは, マルチサーバ方式を採用している. ここで, 各種機能を備えた複数のシステムサーバ群を総称して UNIX サーバと呼ぶことにする.

しかし, この方式ではシステムサーバ間でプロセス間協調作業が頻繁に発生し, 性能が低下することが予想される. Lavender では, この問題はカーネル側で対処すべきであると考え, 効率の良いプロセス間協調作業のための機能を提供している. 例えば, 複数のシステムサーバが参照するデータおよび手続きをモジュール化して共

An Implementation of Unix Server on Lavender Micro Kernel
Masakatsu TOYAMA †, Hideto SAWAKI †, Masahito SHIBA †, Koichi MOURI †and Eiji OKUBO ††
†Graduate School of Science and Engineering, Ritsumeikan University
††Department of Computer Science,
Faculty of Science and Engineering, Ritsumeikan University

有したり, システムサーバをレジデントアドレス空間に配置することにより, 分割によるオーバヘッドを抑えることができる.

3.2 UNIXプロセス

Lavenderにおける UNIXの機能は, 図1で示すように, UNIXサーバおよびエミュレータにより提供される.

Lavenderプロセスは, 1つの仮想アドレス空間に属し, 1つ以上のスレッドを持つものである. UNIXプロセスは, 1つのスレッドを持つLavenderプロセスとして実装される. UNIXプロセスのプロセス管理は, UNIXサーバで行われる.

UNIXプロセスは, システムサービスを利用するために, UNIXサーバと協調する必要がある. この協調作業を支援するための機構として, Lavenderでは, UNIXプロセスと同じ仮想アドレス空間上にエミュレータを配置している. エミュレータは, UNIXサーバと直接やりとりを行う部分であり, プロセスに対しシステムサービスインタフェースを提供する. エミュレータを使用することにより, ユーザレベルでシステムサービスインタフェースの記述が可能になり, Lavenderカーネル自体のカスタマイズを最小限に抑えることができる. これは, 複数のOSパーソナリティが混在するような場合に都合がよい.

また, エミュレータを用いることにより, 同一プロセッサをターゲットとした既存のUNIXバイナリに変更を加えることなく実行することが可能となる. LITES[2]では, エミュレーションライブラリを用いることにより, 4.4 BSDをはじめ, いくつかの既存のシステムのバイナリ互換を実現している. Lavenderでも同様に, 従来のシステムとのバイナリ互換を目指している.

3.3 システムコール

UNIXシステムコールは, UNIXサーバへのRPCとして実装される. UNIXプロセスは, 本来のUNIXカーネル上で動作していることを想定してシステムコール要求を出す. したがって, この要求をLavenderカーネルが受信し, RPCに変換する仕組みが必要になる.

Lavenderカーネルは, 図1に示すように, UNIXプロセスから送られてきたシステムコール要求をエミュレータに転送する. エミュレータは, 要求に合わせて適切なシステムサーバを識別し, RPCにより処理を依頼する. すなわち, エミュレータにおいて, UNIXシステムコール要求はUNIXサーバへのサービス要求に変換される. 処理結果はエミュレータを通してUNIXプロセスに通知される.

4 おわりに

本稿では, Lavenderにおける UNIXサーバの構成について述べた. 我々は, 既存の UNIXシステムとの互換性のある, マルチサーバ方式の UNIXサーバの実現を目指している.

従来のシングルサーバ方式による UNIXサーバは, マイクロカーネル上に実現することによる構造的な利点が得られない. マルチサーバ方式は, 機能の変更および拡張が容易なシステムの構築を可能にする. Lavenderでは, 複数のプロセスへの分割によるオーバヘッドを抑えるため, レジデントアドレス空間やプロセスグループ機能を備えている. また, 処理コストの小さいプロセス間の協調作業を実現するためのモジュール機構を備えている. これらの機能を利用することで, Lavender上ではその構造による利点を生かしたUNIXサーバの実現が可能となる. また, エミュレータを使用した方式は, Lavenderカーネル側からUNIXプロセスに対し統一的な扱いが可能になるため, 複数のOSパーソナリティの同時走行に適した構造をしているといえる.

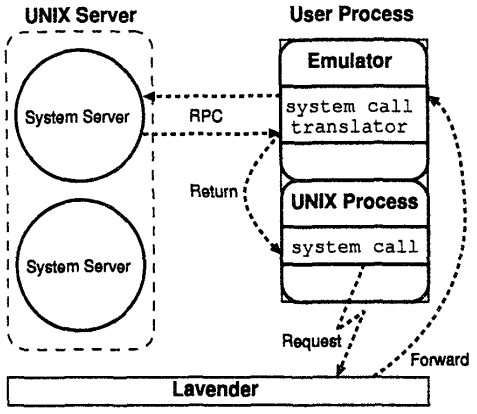

図 1: UNIXシステムコール処理の流れ

参考文献

[1] 佐脇 秀登, 芝 公仁, 豊岡 明, 毛利 公一, 大久保 英嗣: *"マイクロカーネル Lavender におけるプロセス管理方式,"* 第55回 (平成9年後期) 全国大会講演論文集 (1), pp. 222–223 (1997).

[2] Johannes Helander: *"Unix under Mach - The LITES Server -,"* Master's Thesis, Faculty of Information Technology, Helsinki University of Technology (1994).

[3] F. B. des Places, N. Stephen, and F. D. Reynolds: *"Linux on the OSF Mach3 microkernel,"* In Conference on Freely Distributable Software, OSF Research Institute (1996).

[4] H. Härtig, M. Hohmuth, J. Liedtke, S. Schönberg, and J. Wolter: *"The Performance of μ-Kernel-based Systems,"* 16th ACM Symposium on Operating Systems Principles, pp. 66–77 (1997).

[5] Free Software Foundation: *"GNU Hurd information,"* http://www.gnu.org/software/hurd/hurd.html.