マイクロカーネルLavenderにおけるシステムサーバの処理方式

2F−1

芝 公仁† 佐脇 秀登† 豊岡 明† 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

現在, 我々は, マイクロカーネル方式のオペレーティングシステムであるLavenderを開発している. Lavenderでは, システムの機能の多くがシステムサーバと呼ばれるユーザプロセスにより実現される. システムサーバを変更することでシステムの機能を変更できるため, 柔軟な拡張性を持つシステムを構築することができる.

しかし, システムの機能を複数のプロセスへ分割するため, 効率が低下するという問題がある. Lavenderでは, システムサーバの持つデータや手続きをモジュール化し, これをアプリケーションと共有することでこの問題を解決する. システムサーバとモジュールを共有したアプリケーションは, システムサーバの機能を直接使用できるため, 効率よくシステムサーバの機能を利用できる. また, システムサーバをまとめて管理し, それらが提供するサービスや資源へのインタフェースとなる機構の構築も重要な課題である. Lavenderでは, サービスツリーと呼ばれる機構でこれを実現する. サービスツリーは, システムサーバによって提供されるすべてのサービスや資源を木構造で表したものである. アプリケーションは, このサービスツリーを通してシステムサーバの機能を利用する.

以下, 本稿では, 2章でモジュール, 3章でサービスツリーについて述べ, 4章でまとめを述べる.

2 モジュール

2.1 モジュールの共有

システムを機能毎に分割し, それぞれを1つのシステムサーバで実現することで, 各機能の変更が容易になり, システムの柔軟性を向上させることができる. しかし, システムの機能を複数のプロセスに分割するため, プロセス間の協調作業が頻繁に発生し, その際に生じるオーバヘッドが問題となる. 通常, プロセス間の協調作業は, 処理要求のメッセージを相手プロセスに送る, あるいはRPC(Remote Procedure Call)を用いて相手プロセスの手続きを呼び出すなどの方法で行われる. これらの処理には, コストの大きなメッセージ通信が使用されるため, そのオーバヘッドは無視できない.

Lavenderでは, モジュールと呼ばれる機構を利用してこのオーバヘッドを軽減させている. モジュールは, 各プロセスが持つデータや手続きから構成され, プロセスにより動的に定義される. モジュールはプロセス間で共有可能であり, 各プロセスは, モジュールを共有することで, 他プロセスの持つデータの操作や手続きの呼び出しが可能となる. Lavenderでは, クライアントがサーバとモジュールを共有し, そのモジュールを使用することでサービスを利用する. 例えば, あるファイルシステムサーバは, 2次記憶にあるデータとそれに対して読み出しや書き込みなど行う手続きを持つモジュールを提供する. アプリケーションは, このファイルシステムサーバが提供するモジュールを利用してファイルの操作を行う. モジュールの共有により, クライアントはサーバの持つデータや手続きを自プロセスの資源と同様に利用することができる. そのため, システムの機能を複数のプロセスに分割した場合も, システムは効率のよいサービスを提供することができる.

2.2 モジュールの一貫性制御

Lavenderでは, 異なるマシン間でのモジュール共有も可能である. そのため, クライアントは, 同一マシン内のサーバを利用する場合と同じ方法で, 異なるマシンのサーバを利用することができる. モジュールは, データや手続きから構成されるメモリ資源である. そのため, 同一マシン内でのモジュールの共有は, MMU(Memory Management Unit)の機構を利用することで実現できる. しかし, 異なるマシン間でのメモリの共有は一貫性制御に大きなコストがかかり, 効率がよいというモジュール共有の利点が損なわれてしまう. これに関しては, メモリの利用に制約を課し, 弱い一貫性制御を行うことで, コストを軽減させることができる [1]. Lavenderでは, 次に示すようなモジュールの一貫性制御を行う.

(1) プロセスは, 獲得と解放によって, 共有の開始と終了をシステムに通知する. 一貫性は, 獲得時に保証される.

The Processing Scheme of System Servers in Lavender Micro Kernel

Masahito Shiba†, Hideto Sawaki†, Akira Toyooka†, Koichi Mouri†and Eiji Okubo††
†Graduate School of Science and Engineering, Ritsumeikan University
††Department of Computer Science, Faculty of Science and Engineering, Ritsumeikan University

(2) 獲得には, read only と read/write の 2 種類がある.
(3) 同時に 1 個の read/write による獲得が可能であり, read only による獲得は常に許可される.

サービスの要求や終了処理が, 獲得や解放の通知となるため, (1) はプログラマの負担となるものではない. システムはこれらの通知により, 必要な時にのみ一貫性制御を行うことが可能となる. また, (2) の read only であるか read/write であるかは, モジュールが持つ性質によって決まる. モジュールの利用が read only であれば, 複数のプロセスが同時に同一モジュールを獲得することができ, また一貫性制御におけるいくつかの処理を行う必要がなくなる. (3) は, read only での獲得は待たされないことを意味する. read/write での獲得時には, 獲得されるモジュールの複製を作成する. このモジュールが解放される前に read only での獲得要求があれば, そのプロセスには複製が渡される. このような一貫性制御により, 複数のサービスを並列処理することが可能となる.

現在, Lavender では, Pentium 200MHz の PC が 100Mbps のイーサネットで接続された環境で, 他のマシンが持つ 1Kbyte のモジュールの獲得を 3.5ms で行う. この時間は, モジュールの大きさにほとんど依存せず, また 1 回のメッセージ通信に要する時間とほぼ同じである. メッセージ通信を利用したサーバへのサービス要求や RPC による手続き呼び出しでは, メッセージ組み立てのオーバヘッドがかかり, またサービス要求の度にメッセージ通信が必要となる. これに対しモジュール共有の場合, メッセージ組み立ての必要がなく, 一度獲得したモジュールは以後自プロセスの資源として使用できる. そのため, クライアントは, サーバが異なるマシンにあっても, モジュール共有により効率よくサービスを利用できる. また, サーバの機能がクライアントのマシンで実行されるため, 負荷分散が可能となる.

3 サービスツリー

Lavender では, システムの多くの機能がシステムサーバによって実現される. しかし, システムサーバ毎に異なるインタフェースを持っていては, アプリケーションの作成が困難になる. この問題を解決する機構として, Lavender では, システムサーバが提供するサービスや資源への統一的なインタフェースとなるサービスツリーを提供している. サービスツリーは, システムサーバの機能を木構造で管理する. 各システムサーバは, 自身が提供する資源やサービスをサービスツリーに登録し, クライアントからの要求を待つ. 登録されるサービスや資源は, それぞれシステムサーバによって付けられた名前を持つ. 各アプリケーションは, この名前を利用することで, サービスあるいは資源を識別し, サービスの要求を行う. 例えば, あるファイルシステムサーバは, 二次記憶上のデータに名前を付け, これをサービスツリーに登録する. アプリケーションは, この名前を利用してファイルを識別し, そのファイルに対応するモジュールを要求する. 通常のシステムでは, ファイル以外の資源をファイルにマッピングする手法がよく用いられる. しかし Lavender では, ファイルはシステムサーバによって実現される資源の 1 つであり, 常にシステムに存在するものではない. Lavender では, システムサーバによって管理されるすべての資源が, サービスツリーの 1 要素として扱われる.

また, サービスツリーは, システムサーバの実体をアプリケーションから隠蔽する役割を持つ. 各アプリケーションは, サービスツリーに登録された名前を利用してサービスの要求を行う. 各サービスを実現するシステムサーバがどのプロセスであるかは, サービスツリーによって管理されるため, アプリケーションが意識する必要はない. そのため, 同じ名前で同じサービスを実現するシステムサーバであれば, 異なるシステムサーバであってもアプリケーションからは同じように利用できる. このような機能により, アプリケーションから隠蔽された形でシステムサーバを変更することができ, より柔軟な拡張性を持つシステムの構築が可能となる.

4 おわりに

本稿では, Lavender におけるシステムサーバの処理方式について述べた. Lavender では, サーバとクライアント間でのモジュール共有により, オーバヘッドの小さい効率的なサービスを実現できる. また, 異なるマシン間でのモジュールの共有も可能であるため, 異なるマシンのサーバも同じマシン上のサーバと同じ方法で利用することができる. また, サービスツリーにより, システムサーバによって提供されるサービスや資源への統一されたインタフェースが実現される. サービスツリーは, アプリケーションからシステムサーバの実体を隠蔽する役割を持つ. そのため, アプリケーションに影響を与えることなくシステムサーバの変更が可能となり, システムに柔軟な拡張性を持たせることができる.

参考文献

[1] Brian N. Bershad, Matthew J. Zekauskas and Wayne A. Sawdon: *"The Midway Distributed Shared Memory System"*, CMU Technical Report CMU-CS 93-119 (1993).