システムソフトウェアと
オペレーティング・システム 72－5
(1996. 5. 15)

マイクロカーネルLavenderにおけるスケジューリング方式

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

滋賀県草津市野路町 1916
0775-66-1111 (内線 8863)
†mouri@sol.cs.ritsumei.ac.jp ††okubo@cs.ritsumei.ac.jp

あらまし

マイクロカーネル Lavender は, ユーザカスクマイズ可能なカーネル, ポリシとメカニズムの分離, クロスアドレススペースコールのオーバヘッドの軽減を目標として設計されている. Lavender のプロセス管理及びプロセス間通信では, これらの目標を達成するために, それぞれ, 種々のスケジューリングアルゴリズムをユーザレベルで実現可能とする機構と, 共有メモリを使用した効率の良いプロセス間通信のための機構をユーザに提供している. これらの機構を使用することにより, アプリケーションの要求に応じた柔軟かつ高速な機能が実現可能となる. 本論文では, 以上の特徴を持つ Lavender のスケジューリング方式とプロセス間通信の方式について述べる.

キーワード オペレーティングシステム, マイクロカーネル, スケジューリング, プロセス間通信

A Scheduling Method in Lavender Micro Kernel

Koichi Mouri† Eiji Okubo††

†Graduate School of Science and Engineering, Ritsumeikan University
††Department of Computer Science, Faculty of Science and Engineering, Ritsumeikan University

1916 Noji, Kusatsu, Shiga 525, Japan
+81-775-66-1111 (Ext. 8863)
†mouri@sol.cs.ritsumei.ac.jp ††okubo@cs.ritsumei.ac.jp

Abstract

Lavender micro kernel is now being designed to achieve the following objectives: a user-customizable kernel, the separation of policy and mechanism, and the decrease of overhead in cross-address-space calls. In order to achieve these objectives, Lavender provides users with a mechanism which makes it possible to implement various user-level scheduling algorithms, and also provides an efficient inter-process communication by using a shared memory between processes. By using these mechanisms, flexible and fast functions can be implemented according to application requirements. In this paper, methods of the scheduling and inter-process communication in Lavender micro kernel are described.

key words operating system, micro kernel, scheduling, inter-process communication

1 はじめに

近年, マイクロカーネルの適用分野として, マルチメディアシステムやリアルタイムシステムが注目されている. これらのシステムでは, アプリケーションに応じた, 固有のスケジューリングが必要となる [1]. しかし, 従来のマイクロカーネルでは, スケジューラはカーネルの内部に組み込まれていおり, ユーザがスケジューリングアルゴリズムを変更したり, ロック機構や同期機構などのスケジューラに関する機能を変更することは困難である. また, マイクロカーネルでは, システムサーバをユーザプロセスで動作させる. このため, モノリシックなオペレーティングシステムに比べ, プロセス間通信が頻繁に使用される. 従って, プロセス間通信の処理によるオーバヘッドが大くなることが予想される.

我々は, これらの問題を解決し, アプリケーション固有の機能を柔軟かつ効率よく実現することを可能とするマイクロカーネル Lavender を構築している. 本論文では, Lavender におけるプロセスのスケジューリング方式とプロセス間通信機構について述べる.

Lavender では, プロセス及びスレッドのスケジューリング機構として, 内部スケジューラと外部スケジューラの 2 種類を提供している. 内部スケジューラはカーネルが提供するスケジューラである. また, 外部スケジューラは, ユーザプロセスとして実現されるスケジューラである. ユーザは, 外部スケジューラを構築することで種々のスケジューリングアルゴリズムを実現できる.

Lavender のプロセス間通信機構では, コピーによるプロセス間通信と, 共有メモリによるプロセス間通信の 2 種類の方式を提供している. 共有メモリによるプロセス間通信は, 共有メモリを用いて, カーネルとプロセス間, またはプロセス同士でメッセージを送受信することができる. この方式では, メッセージのコピーが発生せず, オーバヘッドの少ないプロセス間通信を実現できる.

以下, 2章で Lavender の概要について述べる. 3章でスケジューラの構成について, 4章でプロセス間通信機構の構成について述べる.

2 Lavender の概要

Lavender の開発に際しては, ポリシとメカニズムの分離を設計目標の中心に据えている. ポリシとは, ユーザから指示された処理を, 条件を考慮して, どのような手順と方法を用いて行うかを決定することをいう. メカニズムは, ポリシの決定に従って実際の処理を行う手続きをいう.

Lavender は, メモリ, プロセス, スレッド, プロセス間通信, 割り込みに関する機能を持つマイクロカーネルであり, 以下の特徴を有している.

(1) 階層化インタフェース

Lavender では, カーネル内の各機能を, ニュークリアス層, カーネル層, システム層の 3 つの階層によって構成している. ニュークリアス層は, ハードウェアを直接操作する機能を提供する層である. カーネル層は, ハードウェア独立の実現と, ポリシを含まない基本機能を提供する層である. システム層は, 基本機能を組み合わせて構成され, ポリシを提供する層である. これらにより, ユーザはアプリケーションに応じたインタフェースを利用したり, カーネルをカスタマイズすることができる.

(2) プロセスグループ機能

1 つのユーザアドレス空間内に複数のユーザプロセスを同時に存在させることを可能にする機能である. 同じユーザアドレス空間内に配置されるプロセス群をプロセスグループと呼ぶ. プロセスグループ内では, 従来アドレス空間の切替えの必要のあったプロセス間の手続き呼び出しを, 同一アドレス空間内における手続き呼び出しに置き換えることができる. さらに, 同一プロセスグループに属するプロセス間で, 共有メモリを確保することができる.

(3) レジデントアドレス空間

Lavender では, 仮想アドレス空間をカーネルアドレス空間, ユーザアドレス空間, レジデントアドレス空間の 3 つに分類している. レジデントアドレス空間は, ユーザアドレス空間の特殊な形態である. レジデントアドレス空間は, ユーザプロセスが配置される空間であるが, システム内に 1 つしか存在しない. さらに, レジデントアドレス空間は常に有効であり, ユーザアドレス空間のように切り替わらない. これによって, レジデントアドレス空間内にあるプロセスは, システム内に存在するすべてのプロセスに対して, 従来アドレス空間の切替えの必要のあったプロセス間の手続き呼び出しを, 同一アドレス空間内における手続き呼び出しに置き換えることができる. さらに, 共有メモリも確保することができる.

3 スケジューラの構成

Lavenderでは, プロセス及びスレッドのスケジューリング機構として, 内部スケジューラと, 外部スケジューラの2種類を提供している. 内部スケジューラは, カーネル内のシステム層で提供されているスケジューラであり, 外部スケジューラが定義されていない場合に使用される. また, 外部スケジューラは, ユーザアドレス空間に配置され, ユーザプロセスで実現されるスケジューラである. ユーザがスケジューリングアルゴリズムを変更する場合は, 外部スケジューラを構築することで実現できる.

本章では, Lavenderにおける, プロセス・スレッドモデルについて述べ, 内部スケジューラと外部スケジューラについて述べる.

3.1 プロセス・スレッドモデル

Lavenderでは, プロセス・スレッドモデルを採用している. プロセスおよびスレッドの構成を図1に示す.

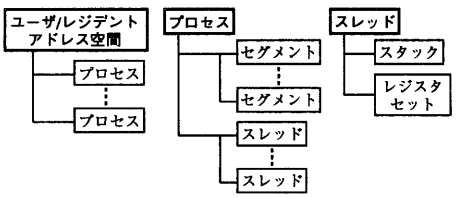

図1 プロセスとスレッド

Lavenderにおけるプロセスは, 静的な概念である. プロセスは, 1つのユーザアドレス空間またはレジデントアドレス空間に属する. プロセスに割り当てられるメモリ領域は, セグメントと呼ばれるメモリブロックの集合として構成され, それらをスレッドに対して実行環境として提供する. また, 1つのプロセスの中で複数のスレッドを動作させることができる.

スレッドは, 動的な概念であり, 1つのプロセスまたはカーネルに属す. スレッドは, レジスタセットとスタックを持った実行実体である. スレッドは, プロセスまたはカーネルが提供する環境の中で動作する.

3.2 カーネルとスケジューラの独立性

ユーザレベルで種々のスケジューリングアルゴリズムを実現させるためには, スケジューラとカーネルの独立性を向上させて構成する必要がある. それぞれを互いに独立させることで, それぞれの機能を自由に拡張・変更することができる. Lavenderでは, カーネルがスケジューラに要求する機能を以下に示すものに限定した.

・スケジューリングの要求
・スレッドの追加と削除の要求
・スレッドの状態変更の要求

また, 基本的なプロセスの状態には, 実行状態, 実行可能状態と待ち状態の3つがある. すなわち, スケジューラから見た場合, プロセスがCPU割り当ての対象になるかならないかのどちらかである. 従って, Lavenderでは, スケジューラに要求されるスレッドの状態変更に関する機能は, 以下に示す2つに限定した.

・スレッドをCPU割り当ての対象とする機能
・スレッドをCPU割り当ての対象から除外する機能

3.3 スケジューラの概要

Lavenderにおけるスケジューラは, 内部スケジューラと外部スケジューラに分けられる. 内部スケジューラは, 階層化インタフェースのシステム層に位置する(図2参照). また, 外部スケジューラは, ユーザレベルで動作するユーザプロセスとして実現される.

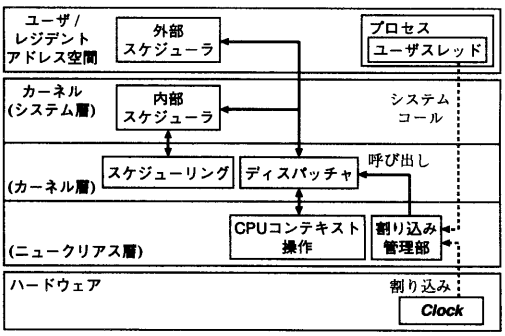

図2 スケジューラの位置と呼び出し手順

これらのスケジューラは, 図2に示す手順で呼び出され, スレッドの切替えを行う. 図中の矢印は制御の流れを表している. ハードウェアのクロックによる割り込みが発生した場合, またはユーザスレッドからシステムコールが発行された場合, 制御は割り込み管理部に移行する. このときスケジューリングが必要であ

れば, 割り込み管理部はディスパッチャを呼び出す. 外部スケジューラが定義されている場合, ディスパッチャは外部スケジューラにスケジューリングを依頼する. そうでない場合は, 内部スケジューラにスケジューリングを依頼する. 依頼を受けたスケジューラは, スケジューリングアルゴリズムに従いキューを操作し, 次にスケジューリングすべきスレッドをディスパッチャに返す. ディスパッチャは, ニュークリアス層のCPUコンテクスト操作に関する機能を用いてスレッドのコンテクストをスイッチし, 次のスレッドを動作させる.

3.4 内部スケジューラ

内部スケジューラは, カーネル内のシステム層で提供されているスケジューラである. 外部スケジューラが定義されていない場合に使用される. 内部スケジューラは, スケジューリングアルゴリズムに従ったキュー操作と, スレッドの生成・削除, および状態の変更要求の処理を行う. 現在, ラウンドロビン方式によるスケジューリングアルゴリズムが動作している. また, スレッドをCPU割当の対象とする機能と, 対象から除外する機能を提供している.

内部スケジューラの構成を図3に示す. 内部スケジューラが外部に対して提供しているインタフェースには, スケジューラの初期化を行うインタフェースと, スケジューラに対して要求を行うためのインタフェースの2つがある.

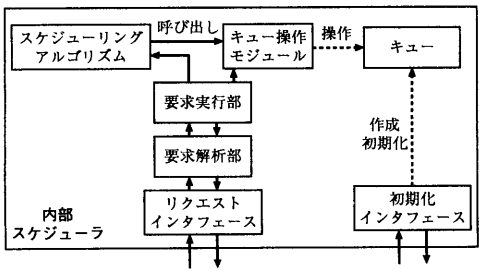

図3 内部スケジューラの構成

初期化インタフェースは, スケジューラ起動時に一度だけ呼び出され, そして, スケジューラ自体の環境を作成するための処理を行う. 具体的には, キューの作成, スレッド操作のためのポインタの初期化などが行われる.

リクエストインタフェースは, カーネルやユーザがスケジューラに対する要求を伝えるためのインタフェースである. スレッドの追加・削除, 状態変更などは, このインタフェースを介して要求することができる. リクエストインタフェースに要求が出されると, スケジューラはその要求のメッセージを解析し, 該当する内部関数を呼び出して処理を行う. 現在の実装では, このリクエストインタフェースを介して, スケジューリング, スレッドの登録・削除, スレッドの状態変更を要求することができる. スレッドの状態変更は, スレッドをスケジューリング対象とする機能, スレッドをスケジューリング対象から除外する機能を提供している.

3.5 外部スケジューラ

アプリケーションによっては, スケジューリングアルゴリズムの変更が必要とされる場合がある. Lavenderでは, このような場合に外部スケジューラを用いることができる. 外部スケジューラは, ユーザによってスケジューリングアルゴリズムを変更することのできる, ユーザレベルで動作するスケジューラである. 基本的な外部スケジューラの構成を図4に示す. 外部スケジューラは, ユーザプロセスとして起動される. そのため, 外部からの要求はすべてプロセス間通信を用いて行われる.

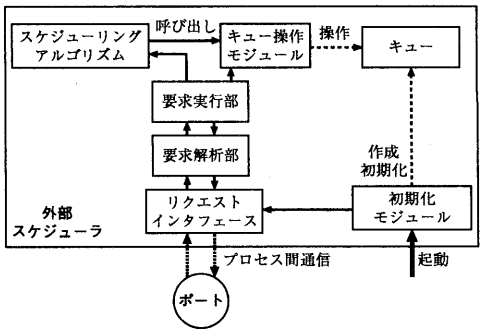

図4 外部スケジューラの構成

外部スケジューラは, 起動時に自分自身で初期化を行う. 初期化モジュールでは, キューの作成・初期化などを行う. さらに, 通信のためのポートの作成を行う必要がある. 初期化が終了した後は, リクエストインタフェースに制御を移す. リクエストインタフェースは, カーネルやユーザからプロセス間通信によって送られてきたメッセージを順次取り出し, 要求の解析を行う.

ユーザは, 外部スケジューラの要求解析部と実行部, およびスケジューリングアルゴリズムを拡張することで種々のスケジューリングに関する機能を変更・拡張することができる.

4 プロセス間通信機構の構成

Lavender におけるプロセス間通信機構は, ユーザプロセス間, およびカーネルとユーザプロセス間における, ポートを基本とした通信機構を提供している. 本章では, Lavender におけるプロセス間通信機構の概要と, ポート, メッセージ, プロセス間通信サーバの各構成要素について述べる.

4.1 プロセス間通信機構の概要

Lavender の階層化インタフェースにおけるプロセス間通信機構と, それらの間の処理の流れを図 5に示す. Lavender では, プロセス間通信は, アプリケーション同士の通信や, 各種のデバイスによる割り込みをプロセスへ通知するときに用いられる. アプリケーション同士の通信の場合, 送信するメッセージはアプリケーションが作成する. 従って, メッセージの送信を要求するシステムコールが発行された場合, その割り込みを取得した後, 直接プロセス間通信サーバが呼び出される. また, カーネルがデバイスによる割り込みを受信した場合, 送信するメッセージを作成する必要がある. 従って, デバイスによる割り込みが発生すると, それを取得した後に, 割り込みをメッセージに変換する処理を行う. そして, メッセージを送信するためにプロセス間通信サーバを呼び出す. プロセス間通信サーバは, メッセージ送信手続きを行い, 受信側のプロセスに対して, メッセージの到着通知を行う. また, 受信側プロセスからメッセージ受信の要求が到達するとメッセージを渡す.

4.2 メッセージとメッセージヘッダ

Lavender におけるメッセージは, 任意の大きさのデータのブロックである. カーネルでは, このメッセージを管理するために, メッセージヘッダ構造体を用いている. メッセージヘッダ構造体は, メッセージの識別番号, シーケンス番号, メッセージの型, メッセージのサイズ, メッセージへのポインタから構成されている. メッセージの識別番号は, メッセージを他のメッセージと識別するために用いられる. シーケンス番号は,

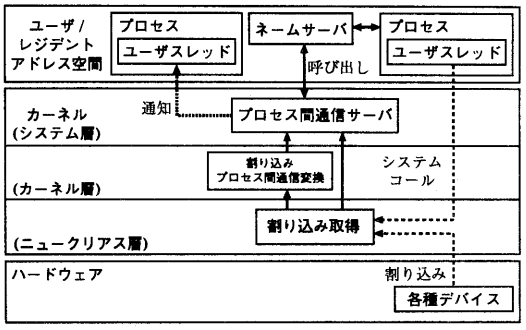

図 5 プロセス間通信機構の位置と呼び出し手順

1つのメッセージを複数に分割して送信する際のメッセージの順番を示している. また, メッセージの型は, コピーによる通信, または共有メモリによる通信のどちらを用いてメッセージが送信されたかを示す. メッセージへのポインタは, 送信すべきメッセージが格納されている場所へのポインタである.

4.3 プロセス間通信サーバ

プロセス間通信サーバはカーネル内に存在し, プロセス間通信機構において中心的な役割を果たす. プロセス間通信サーバでは, ポートの割り当て・削除, メッセージの送受信, メッセージやポートの状態取得のインタフェースを提供している. このうち, メッセージの送受信に関するプリミティブとしてコピーによる通信と, 共有メモリを用いた通信を提供している.

4.3.1 コピーによる通信

従来のプロセス間通信の方法として用いられてきたのが, コピーによるメッセージ通信である. 図 6は, プロセス B からプロセス A にメッセージを送る場合を示している. 以下にその処理の手順を示す.

(1) プロセス B は, 所有するメモリ領域内にメッセージを用意し, メッセージ送信のシステムコールを発行する
(2) プロセス間通信サーバは, プロセス B のメモリ上にあるメッセージを, カーネルヒープへコピーし, メッセージヘッダリストを更新する
(3) プロセス A は, メッセージ受信のシステムコールを発行する
(4) プロセス間通信サーバは, カーネルヒープ上にあるメッセージを, プロセス A のメモリへコピーし,

メッセージヘッダリストを更新する

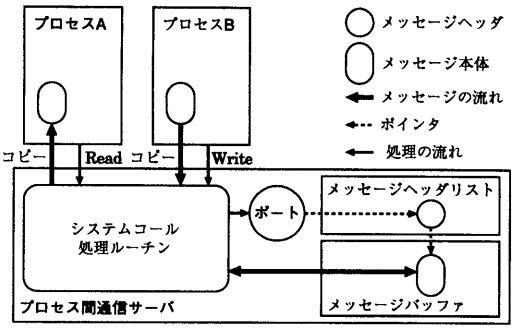

図 6　コピーによるプロセス間通信

コピーによるメッセージ通信では, メッセージのコピーが送信時と受信時の 2 回行われる. このように, コピーによるプロセス間通信は, メッセージにのコピーによるオーバヘッドが大きくなる問題がある.

4.3.2　共有メモリを用いた通信

Lavender では, プロセスグループ機能, およびレジデントアドレス空間を用いることでカーネルとプロセス間, そしてプロセス間で共有メモリを持つことができる. これをによって, 共有メモリを用いた通信を行うことができる. 図 7は, プロセス B からプロセス A にメッセージを送る場合を示している. 以下にその処理の手順を示す.

(1) プロセス B は, 所有するメモリ領域内にメッセージを用意し, メッセージ送信のシステムコールを発行する
(2) プロセス間通信サーバは, メッセージヘッダリストを更新する
(3) プロセス A は, メッセージ受信のシステムコールを発行する
(4) プロセス間通信サーバは, メッセージヘッダリストを更新する

共有メモリによるプロセス間通信は, プロセス同士が前もって共有メモリを確保することで, メッセージのコピーが発生しない, オーバヘッドの少ないプロセス間通信を実現することができる. また, 送信するメッセージの大きさを意識しなくてもよい. さらに, マイクロカーネルにおいて頻繁に使用されるプロセス

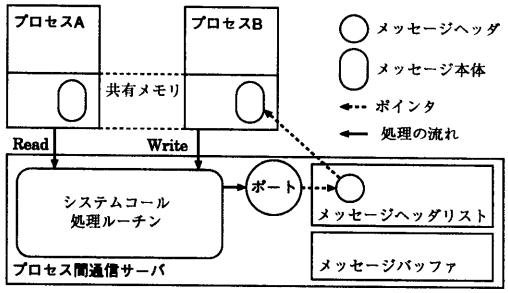

図 7　共有メモリを用いたプロセス間通信

間通信による, システム内のオーバヘッドを軽減することができる.

5　おわりに

本論文では, Lavender におけるプロセスのスケジューリング方式とプロセス間通信機構について述べた.

Lavender ではプロセス及びスレッドのスケジューリング機構として, 内部スケジューラと外部スケジューラの 2 種類を提供した. そして, スケジューリングアルゴリズムや特定の機能をカーネルと独立させることで, ユーザによるスケジューリングアルゴリズムの変更や機能の追加を可能にした.

プロセス間通信機構では, コピーによるプロセス間通信と共有メモリによるプロセス間通信の 2 種類を提供した. 共有メモリによるプロセス間通信では, メッセージのコピーが発生しない, オーバヘッドの少ないプロセス間通信を実現した.

今後は, 外部スケジューラを介して動的にアルゴリズムを変更するための機構, 複数のスケジューリングアルゴリズムを同時に動作させるための機構について検討, 実現する予定である.

参考文献

[1] Brian N. Bershad, Craig Chambers, Susan Eggers, Chris Maeda, Dylan McNamee, Przemyslaw Pardyak, Stefan Savage, Emin Gün Sirer: *"SPIN - An Extensible Microkernel for Application-specific Operating System Service"*, Technical Report UW-CSE-94-03-03, Department of Computer Science and Engineering, University of Washington (1994).