システムソフトウェアと
オペレーティング・システム 76－2
（1997. 8. 21）

マイクロカーネルLavenderにおけるスケジューラの構成

毛利 公一† 佐脇 秀登† 芝 公仁† 豊岡 明† 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

マイクロカーネルLavenderは，ユーザカスタマイズ可能なカーネルとして構築されている．Lavenderは，ポリシとメカニズムの分離，階層化インタフェース，クロスアドレススペースコールのオーバヘッドの軽減などの特徴を持つ．本論文では，特にLavenderにおけるスケジューラの構成について述べる．Lavenderでは，スケジューラをマスタスケジューラとユーザスケジューラの2種類に分類している．そして，これらを多段階に構成することで，ユーザがスケジューリングアルゴリズムを容易に変更することを可能にしている．また，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ことも可能となる．

Structure of the Scheduler in Lavender Micro Kernel

Koichi Mouri† Hideto Sawaki† Hirohito Shiba† Akira Toyooka†
Eiji Okubo††

†Graduate School of Science and Engineering, Ritsumeikan University
1-1-1 Noji-Higashi, Kusatsu, Shiga 525-77, Japan

††Department of Computer Science,
Faculty of Science and Engineering, Ritsumeikan University
1-1-1 Noji-Higashi, Kusatsu, Shiga 525-77, Japan

Lavender micro kernel is designed as a user customizable kernel. Lavender has following features: separation of policy and mechanism, layered interface and decrease of overhead in cross-address-space call. In this paper, a structure of the scheduler in Lavender micro kernel is described. The scheduler consists of a master scheduler and user level schedulers. And they form a multi stage scheduler. This structure allows users to change scheduling algorithm easily, and to run several scheduling algorithms simultaneously.

1 はじめに

現在，マイクロカーネルが持つ柔軟性と拡張性が注目されている．従来のカーネルは，オペレーティングシステム (OS) がユーザに提供する機能をすべて含んでいたため，カーネルのサイズが大きくなる．また，各機能が密接に関係しているため，カーネル内の機能を容易に変更できないなどといった問題点を持つ [1]．マイクロカーネルは，これらの問題点を解決するものとして期待されている．

また，今日では広域ネットワーク，および組織内のネットワークが発展・浸透してきた．このような状況の中，動画および音声の転送を行うためのマルチメディアアプリケーションのニーズが高まっている．これらのアプリケーションを構築する場合，データの転送や再生などの処理においてリアルタイム性が必要とされ，そのリアルタイム性を OS が保証する必要がある．

さらに，現在では，高性能なコンピュータを安価に購入することができるようになり，コンピュータが様々な場面に利用されるようになった．その際に用いられるアプリケーションも多様化している．このような場合，1 つのコンピュータ上で動作するアプリケーションが OS に対して要求する機能も多様化する．

このような状況を想定し，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．Lavender では，マイクロカーネルの持つ柔軟性と拡張性の高さを活用し，個々のユーザアプリケーションが必要とする環境を OS として提供することを目的としている．本論文では，特に Lavender におけるスケジューラの構成について述べる．

我々が現在構築している Lavender におけるスケジューラは，アプリケーションに応じた柔軟なスケジューリングアルゴリズムを提供するための機構として，多段階スケジューリング方式を採用している．本スケジューラは，システム内に唯一でメカニズムを提供するマスタスケジューラと，システムに複数存在しポリシを提供するユーザスケジューラとから構成される．ユーザは，ユーザスケジューラとして様々なスケジューリングアルゴリズムを構築することができる．また，1 台のコンピュータ上で，リアルタイム・非リアルタイムを問わず，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ことが可能となる．

以下，本論文では 2 章で Lavender の構成と特徴，3 章でスケジューラの構成について述べ，4 章でスケジューラの実装と性能評価について述べる．

2 Lavender の構成と特徴

Lavender を含め，OS 全体としての構成を図 1に示す．Lavender はカーネル領域に配置され，その他のユーザプロセス，デバイスドライバ，各種サーバ類はすべてユーザ領域に配置される．

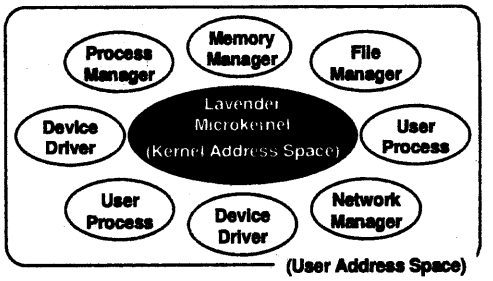

図 1 Lavender の全体構成

Lavender は，以下に挙げる特徴を持つマイクロカーネルである．

・ユーザカスタマイズ可能
・ポリシとメカニズムの分離
・階層化インタフェース
・クロスアドレススペースコールのオーバヘッド軽減

従来，カーネルの機能を変更する場合，カーネルのソースコードを変更し，さらに再構築する必要があった．Lavender ではユーザカスタマイズ可能なカーネルとして実現することで，カーネルそのものとは独立したモジュールを用いてカーネルの機能を変更することを可能にしている．Lavender では，このモジュールをサーバとして位置づけている．

Lavender は，サーバによるユーザカスタマイズ性を可能にするために，ポリシとメカニズムの分離の概念を採用している．ここでいうポリシとは，ある処理を遂行するための手法や手順を決定するものである．また，メカニズムとは，ポリシの決定に従って実際に個々の処理を行うものである．Lavender では，このポリシとメカニズムの分離を，階層化インタフェースによって実現している．

Lavender の内部の構成を図 2に示す．Lavender はメモリ管理，プロセス管理，スケジューリング，割り込み管理，プロセス間通信の機能を持つ．さらにこれらの機能はニュークリアス層，カーネル層，システム層の 3 つの層に分類されている．各層における分類方法を以下に示す．

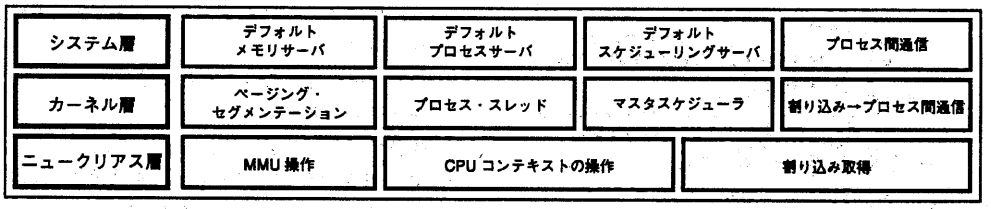

図 2　Lavender 内部の構成

(1) ニュークリアス層

この層は，ユーザに対してハードウェアを直接操作するためのインタフェースを提供する．また，ユーザがハードウェアに依存した形式でカーネル内の情報を参照，変更することを可能にする．

(2) カーネル層

ニュークリアス層で提供されるインタフェースを組み合わせて構成されている．カーネル層では，メカニズムを提供するとともに，ハードウェアの抽象化を行っている．また，カーネル内の情報や状態もハードウェアに依存しない形式に抽象化されており，ユーザはハードウェアの違いを意識することなく直接参照・変更することができる．

(3) システム層

カーネル層で提供されるインタフェースを組み合わせて構成されている．システム層では，ポリシとして，ユーザに対して従来のマイクロカーネルのシステムコールに相当する高機能なインタフェースを提供している．

マイクロカーネル方式を採用した OS では，プロセス間手続き呼び出しやプロセス間通信が頻繁に使用される．これらによって発生するオーバヘッドを小さく抑えることが，マイクロカーネルが解決すべき一つの課題となっている．Lavender では，プロセスグループ機能，レジデントアドレス空間を用いることで，プロセス間通信，プロセス間手続き呼び出しなどの際に発生する，データのコピーやアドレス空間の切り替え，コンテキストの切り替えによるオーバヘッドを軽減させている [2]．

3　スケジューラの構成

Lavender におけるスケジューラは，アプリケーションに応じた柔軟なスケジューリングアルゴリズムを提供するための機構として，多段階スケジューリング方式を採用している．本機構は，Lavender の特徴であるポリシとメカニズムの概念に基づき構成されている．以下，プロセス・スレッドモデルについて説明し，スケジューラの構成について述べる．

3.1　プロセス・スレッドモデル

マイクロカーネル Lavender は，サーバやユーザプログラムを実行するための概念としてプロセス・スレッドモデルを採用している．以下に Lavender におけるプロセスとスレッドの定義を示す．

・プロセス … プロセスは静的な概念である．プロセスは 1 つ以上の仮想アドレス空間に属し，1 つ以上のセグメントに分割された実行環境である．プロセスはスレッドに対して自分の環境を提供する．
・スレッド … スレッドは動的な概念である．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が提供する環境の中で動作する．スレッドは 1 つのプロセス内に複数あってもよい．

Lavender のスケジューラでは，スケジューリングの単位をスレッドとしている．

3.2　スケジューラの概要

Lavender のスケジューリング機構は，マスタスケジューラとユーザスケジューラの 2 種類から構成される (図 3参照)．

マスタスケジューラは，カーネル層に属し，システムに唯一である．スケジューリングアルゴリズムなどのポリシを持たない．マスタスケジューラは，ユーザスケジューラによってその内容が設定される 1 つのキューを持つ．マスタスケジューラは，そのキューを参照し，その内容に従ってスレッドに CPU を割り当てる．また，ユーザスケジューラに対してキューを操作する機能と時刻取得の機能を提供する．

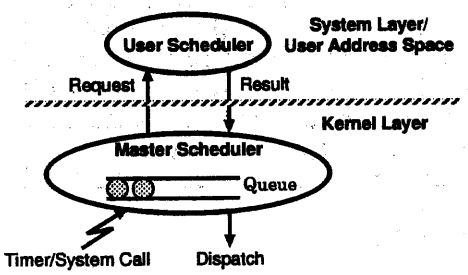

図 3　スケジューリング機構の概要

ユーザスケジューラは，デフォルトスケジューリングサーバとしてシステム層に属するか，またはユーザプロセスとして実現される．ユーザスケジューラでは，スケジューリングアルゴリズムが実装され，実際にスケジューリングを行う．スケジューリングを行う際には，必要な情報をカーネルから取得し，スケジューリングの結果はマスタスケジューラの持つキューに反映させる．

3.3　マスタスケジューラ

マスタスケジューラは，Lavender のスケジューリング機構においてメカニズムを実現する部分である．マスタスケジューラは，タイマ割り込み，システムコール，およびカーネル内からの関数呼び出しよって起動される．起動されると，どのような目的で呼び出されたかを判断し，それに対応する処理を行う．各々の処理を行うために用意されているマスタスケジューラのインタフェースを以下に示す．

・初期化
・キューの操作
・情報の取得
・スケジューラへの処理要求の受け付け
・スレッドへの CPU の割り当て

(1) 初期化

初期化ルーチンは，システムの起動時に 1 度だけ呼び出される．ここでは，以下の処理が行われる．

・マスタスケジューラの持つキューの生成
・デフォルトスケジューリングサーバへの初期化要求
・デフォルトスケジューリングサーバのキューへの登録

(2) キュー操作関数の提供

ユーザスケジューラがマスタスケジューラのキューを操作するためのインタフェースを提供している．提供されているインタフェースを以下に示す．

・キューのクリア
・キューの最後へのスレッドの追加
・キューの内容取得
・複数のスレッドを一度にキューへ追加

ユーザスケジューラがキューの操作をする場合，キューで示されるスレッドの実行順序とともに，後で述べるスレッドのスケジューリング情報も同時に登録する．

(3) 情報取得関数の提供

マスタスケジューラがユーザスケジューラに対して情報を提供するためのインタフェースとして，現在時刻を提供するインタフェースが用意されている．デバイス入出力による負荷，メモリの利用状況などの情報は，デバイスドライバ，メモリ管理部，プロセス管理部，各種サーバプロセスに問い合わせることによって取得する．

(4) スケジューラへの処理要求の受け付け

ユーザプロセスやカーネル内の各管理部がスケジューラに処理を要求するときに呼び出される．それらの要求は，ここで適切なユーザスケジューラに転送される．転送されるべきユーザスケジューラの決定は，スレッドの持つ所属スケジューラ情報を利用して行われる．

(5) スレッドへの CPU の割り当て

本インタフェースは，タイマ割込みが発生したとき，スレッドが CPU を放棄したときなどに呼び出される．その処理の流れは以下の通りである (図 4参照)．

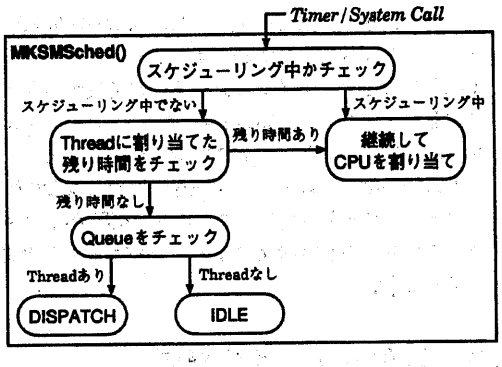

図 4　マスタスケジューラの流れ

(a) 割り込みが発生した時に実行していたスレッドがユーザスケジューラのスレッドかどうかチェックする．

(b) ユーザスケジューラのスレッドであった場合は継続して CPU を割り当てる.
(c) 上記以外のスレッドの場合, 該当するスレッドに割り当てた時間が残っている場合も継続して CPU を割り当てる.
(d) 時間が残っていない場合は, キューを参照し, 次に実行すべきスレッドがあれば CPU を割り当てる.
(e) 次に実行すべきスレッドがない場合はアイドル状態となる.

上記の処理を行うときに, 利用されるのがマスタスケジューラの持つキューである. このキューは, ユーザスケジューラによって操作される. キューではスレッドの割り当て順序を管理するとともに, スレッドのスケジューリング情報へのポインタを管理する. マスタスケジューラは, キューで示される順序とスレッドのスケジューリング情報を参照し, それに従ってスレッドに CPU を割り当てる処理を行う. マスタスケジューラの持つスケジューリング情報を以下に示す.

・スレッド ID … スレッド構造体へのポインタ.
・スレッド属性 … タイムアウト時のスレッドの扱いを決定するために用いる. 割り当て時間を使い切った時に次のスレッドに CPU 割り当てるか, 継続して CPU を割り当てるかを指定する. 通常のユーザプロセスのスレッドと, ユーザスケジューラのスレッドの区別に用いる.
・スレッド実行開始時刻 … スレッド実行開始時刻を指定する. ただし, 時刻に 0 が指定されている場合は現在の時刻に関わらず実行を開始する.
・CPU 割り当て期間 … スレッドに対して連続して CPU を割り当てる時間を指定する.

3.4 ユーザスケジューラ

ユーザスケジューラは, Lavender のスケジューリング機構においてポリシを実現する部分である. ユーザスケジューラでは, スケジューリングアルゴリズムが実装され, 実際にスケジューリングを行う機能を持つ. ユーザスケジューラの構成方法は特に限定されていないが, 外部とのインタフェースは 1 つの関数で提供される.

ユーザスケジューラは, マスタスケジューラによって呼び出されることによって実行される. このとき, マスタスケジューラから処理要求を格納した要求パケットを受け取る. ユーザスケジューラはその要求パケットに従い処理を行う. 要求パケットの内部には, パケットの先頭に処理要求コード, それ以降はそれぞれの処理要求に応じたパラメタが順に格納されている.

処理要求コードは, 最低限必要なコードと, ユーザスケジューラ毎に独自に拡張するコードとに分類できる. 最低限必要なコードとは, カーネルが必要としている機能を実現するためのコードである. 以下にそれらのコードに対応する機能を示す. これら以外の機能を追加したい場合は, それぞれのユーザスケジューラで独自に拡張することができる.

・ユーザスケジューラの初期化
・スケジューリング
・スレッド登録・削除
・スレッドの状態を実行可能状態・待ち状態に変更

3.5 複数ユーザスケジューラと多段階スケジューラ

Lavender では, 複数の, 異なるスケジューリングアルゴリズムを持つユーザスケジューラを同時に動作させることを可能とする. これを実現するためには, 複数のスケジューラによるスケジューリング結果が競合した際に, あるポリシによって競合を調整する必要がある. Lavender では, この競合を調整するポリシもユーザスケジューラとして実現する. このような場合, スケジューラを以下に示す 3 つに分類し, 多段階スケジューリング機構によって実現する (図 5参照).

・スケジューリングを行うユーザスケジューラ
・スケジューラ間の競合を調整し, マスタスケジューラのキューを直接操作するユーザスケジューラ
・マスタスケジューラ

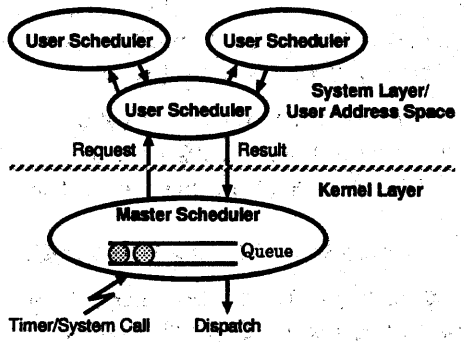

図 5 多段階スケジューラ構成

4 スケジューラの実装と性能評価

前章までに述べた機構を用い，システム層にラウンドロビンスケジューリングアルゴリズムを実現した．ラウンドロビンを実現したユーザスケジューラは以下に示す機能を持つ．

・初期化
・スケジューリング
・指定したスレッドの状態変更
・スレッド追加・削除

(1) 初期化

実行可能キューと待ちキューをそれぞれ生成する．

(2) スケジューリング

(a) 実行可能キューの先頭にあるスレッドを取り出し，マスタスケジューラのキューの最後に追加する．

(b) 取り出したスレッドを実行可能キューの最後に追加する．

(c) ユーザスケジューラ自体がスケジューリングされるよう，ユーザスケジューラをマスタスケジューラのキューの最後に追加する．

(3) スレッドの状態変更

指定されたスレッドの状態を実行可能状態または待ち状態に変更する．スレッドは，現在属しているキューからはずされ，指定された状態に応じ実行可能キュー，または待ちキューの最後に追加される．

(4) スレッドの追加・削除

追加の場合は，指定されたスレッドを実行可能キューの最後に追加する．削除の場合は，指定されたスレッドを実行可能キューと待ちキューから検索し，キューからはずす．

上記ユーザスケジューラを用い，その性能評価を行った．評価は，PC/AT 互換機 (Pentium MMX 200MHz) を使用し，以下に示す条件の下でスレッドの平均ターンアラウンドタイムを求めた．

・1 つのプロセスは 1 つのスレッドを持つ．
・プロセス数を 1 から 6 まで変化させる．
・タイムスライスを 100μs，1ms，10ms，100ms とする．
・スレッドは整数のインクリメントを 1 億回繰り返す．

実行結果を図 6に示す．1ms，10ms，100ms では結果にほぼ差がない．100μs の場合はスケジューリングオーバヘッドが顕著に現れる．また，この結果より，プロセス数が 1 の場合の 1 回のスケジューラ呼び出し・実行にかかるオーバヘッドは約 3.9μs である．

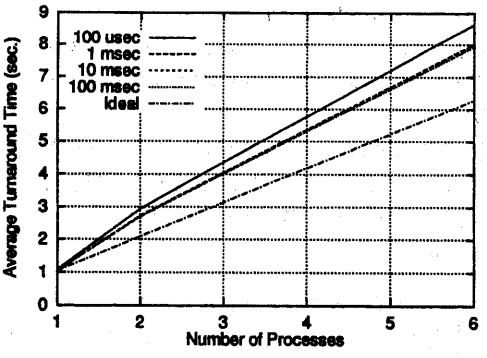

図 6 平均ターンアラウンドタイム

5 おわりに

本論文では，マイクロカーネル Lavender におけるスケジューラの構成と，その性能評価について述べた．Lavender では，スケジューラをマスタスケジューラとユーザスケジューラの 2 種類に分類して構築している．本手法を用いることで，ユーザが容易にスケジューリングアルゴリズムを変更・構築することが可能となる．また，多段階スケジューリング機構を用いることで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ことも可能となる．

今後は，実装中のレートモノトニックスケジューリングアルゴリズムについて評価する予定である．また，複数のスケジューリングアルゴリズム間で衝突が起こった際の調整手法についても実装・評価を行う予定である．さらに，タイマの割り込み間隔を可変にし，不要な割り込み処理をできる限り削減して，システムのオーバヘッドを軽減することも検討している．

参考文献

[1] B. N. Bershad et al.: *"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).

[2] 芝 公仁，佐脇 秀登，豊岡 明，毛利 公一，大久保 英嗣: *"マイクロカーネル Lavender の構成"*, 情報処理学会研究報告 97-OS-75, pp. 7–12, 情報処理学会 (1997).