システムソフトウェアと
オペレーティング・システム 88−2
（２００１．　７．２６）

*Tender*における資源「演算」を利用したプロセスグループの実行速度調整法

田端 利宏†　　谷口 秀夫‡

九州大学大学院システム情報科学府†
九州大学大学院システム情報科学研究院‡

計算機を利用した様々なサービスが提供されている．計算機で提供されるサービスは，複数のプロセスから構成されることが多い．例えば，Web サーバでは，サーバ側で複数の子プロセスをあらかじめ用意し，クライアントの接続要求毎に，子プロセスが処理を行う．これまでに，我々が提案したプログラム実行速度調整方式では，プロセスを単位として，プログラム実行速度を調整できる．しかし，複数プロセスから構成されるサービスを単位として，実行速度を調整することはできない．そこで，我々は，資源「演算」を利用したプロセスグループの実行速度調整法を提案する．提案方式では，サービスを構成するプロセス群をプロセスグループという実行速度調整の単位として扱う．本論文では，資源「演算」を利用し，プロセスグループを構成し，その実行速度を調整する方法について述べる．具体的には，プロセスグループを「演算」の木構造で表すことにより，プロセスグループを実現する.「演算」を利用したことで，プロセスグループの実行速度調整と処理時間の保証が可能となる．さらに，サービスの処理内容に合わせて，プロセスグループの階層化が可能となる．

Execution Speed Control Mechanism of Process Group by Execution on ***Tender***

Toshihiro TABATA and Hideo TANIGUCHI

Graduate School of Information Science and Electrical Engineering, Kyushu University

The various services, which use a computer, are provided. There are many cases the service, which is provided with a computer, is composed of multi processes. For example, in Web server, a child process processes a request from a client. We proposed speed control mechanism of program execution. The mechanism is able to regulate the execution speed of one process. However, execution speed of a service, which is composed of the multi processes, is unable to be regulated. Therefore, we propose the execution speed control mechanism of process group by using "execution" resource. This paper describes overview of ***Tender*** operating system, speed control mechanism of program execution and the guarantee mechanism of service processing time. Furthermore, it proposes execution speed control mechanism of process group by execution on ***Tender***.

1　はじめに

計算機ハードウェアの性能向上により，ソフトウェアの処理時間は短縮される．この結果，複雑な処理が可能になっている．一方，ソフトウェアの処理速度は，ハードウェア性能に大きく依存する．このため，計算機ハードウェアの性能によって，ソフトウェアのサービス処理速度が異なる．例えば，最新でない低い計算機を想定して開発された既存ソフトウェアがあるとする．このソフトウェアを，最新の高性能な計算機ハードウェア上で走行させると，処理が

速過ぎて表示内容を人間が理解できないことがある．このため，利便性が低下し，既存の優れたソフトウェアを適度な実行速度で利用し続けることができなくなる．また，最新の高性能な計算機で，このソフトウェアを利用し続けるには，ソフトウェアの改版が必要となる．しかし，ソフトウェアの改版には，大きな工数がかかる．さらに，ハードウェア性能は，今後も向上し続けるため，ソフトウェアの改版を行い続けなければならない．したがって，将来に向け，計算機ハードウェアの性能の範囲で，利用者が求める速度，あるいはソフトウェアが提供するサービス内容に合わせた速度で，サービス処理を制御する方式を確立する必要がある．そこで，我々は，計算機ハードウェアの性能の範囲で，プログラム実行速度を自由に調整する方式を研究している．

計算機を利用した様々なサービスが提供されている．これらのサービスは，その処理が複雑であることが多く，同時に多くの要求を処理する．このため，計算機で提供されるサービスは，複数のプロセスから構成されることが多い．例えば，Web サーバでは，サーバ側で複数の子プロセスをあらかじめ用意し，クライアントの接続要求毎に，子プロセスが処理を行う．これまでに，我々は，一つのプロセスの実行速度を調整できる方式を提案した[1][2]．また，複数のプロセスの実行速度を同時に調整する方式を提案した[3]．さらに，一つのプロセスの処理時間を保証する方式を提案した[4]．しかし，提案した方式では，個々のプロセス単位で実行速度を指定する．したがって，サービスを構成するプロセス群（以降，プロセスグループと呼ぶ）を単位として，実行速度を指定することはできない．このため，サービスを単位として，プログラム実行速度を調整する場合，サービスを構成するプロセス単位で，実行速度を調整する必要があり，利便性が低下する．そこで，我々は，プロセスグループを単位として，プログラムの実行速度を調整する方式を提案する．

本論文では，資源「演算」を利用し，プロセスグループを構成し，その実行速度を調整する方法について述べる．具体的には，プロセスグループを「演算」の木構造で表すことにより，プロセスグループを実現する.「演算」を利用したことで，プロセスグループの実行速度調整と処理時間の保証が可能となる．さらに，サービスの処理内容に合わせて，プロセスグループの階層化が可能となる．

2 *Tender* オペレーティングシステム

Tenderでは，OS の操作する対象を資源として，分離し独立化している[5]．また，各資源のプログラムの呼び出しは，表プログラム構造部を介して行う．資源には，資源名と資源識別子を付与し，資源操作のインタフェースを統一している．さらに，各資源を操作するプログラム部品を資源毎に分離し，共有プログラムを排除した．また，各資源の管理情報も資源毎に分離し，各資源の管理表の間のポインタを禁止した．

このように，資源の分離と独立化を行うことで，資源の事前生成や保留により，資源の作成や削除を伴う処理を高速化している．また，OS の動作や内部状態の理解や把握が容易になり，OS の理解を支援できる．さらに，プログラムを部品化できるため，機能の追加や変更が容易になっている．

3 資源「演算」とは

3.1 基本機能

資源「演算」とは，プロセッサを利用でき，プログラムを実行できる程度（以降，演算の程度と名付ける）を持つ資源である．演算をプロセスと関連付けることにより，プロセスの実行が可能になる．つまり，利用者は，プロセスと演算を生成し，プロセスに演算を関連付けることで，演算が持つ演算の程度に応じてプロセスを走行させることができる．このため，演算の程度を変更することで，プロセッサを利用できる程度を変更できる．さらに，プロセスを生成し，演算の関連付けを行なわないことによるプロセスの作り置きや，演算の関連付けと解除によるプロセスの実行の停止と再開が容易に実現できる．

演算の程度とは，プロセッサを利用できる程度を表すもので，各演算ごとに演算の程度を持っている．現在，演算の程度には，2 種類のものがある．一つは，プログラムの実行速度を表すものである．もう一つは，できる限りの多くの演算程度を要求するもので，優先度を表すものである．

表 1に，演算管理インタフェースを示す．演算管理には，八つのインタフェースがある．

3.2 資源「演算」の種類

3.2.1 性能調整の演算

性能調整の演算は，演算の程度として，プログラムの実行速度を調整する性能（1〜100%，プロセッサそのものの性能を 100%とする，1%単位で指定）を持つ．性能調整の演算が持つ性能の総計が，100%以下

表 1 演算管理インタフェース

通番	形式	機能
1	creat_execution(mips)	mips で指定した演算を生成し，演算識別子 execid を返す．mips は，1〜100 の時，演算の程度（%，プロセッサそのものの性能を 100%とする）を要求し，0 または負の時，できる限りの多くの演算の程度を要求する（絶対値はプロセス優先度）．
2	delete_execution(execid)	演算 execid を削除する．
3	attach_execution(execid, pid)	演算 execid をプロセス pid に割り当てる．
4	detach_execution(execid, pid)	演算 execid のプロセス pid への割り当てを解除する．
5	wait_execution(pid, chan)	プロセス pid に割り当てられた演算へのプロセッサの割当を禁止し，プロセスを WAIT 状態にする．pid が 0 の時は，現在走行中のプロセスに割り当てられた演算へのプロセッサの割当を禁止し，プロセスを WAIT 状態にする．chan は，休眠識別子である．
6	wakeup_execution(pid, chan)	プロセス pid に割り当てられた演算へのプロセッサの割当を可能にし，プロセスを READY 状態にする．pid が 0 の時は，休眠識別子 chan でプロセッサの割当を禁止したプロセスに割り当てられた演算へのプロセッサの割当を可能にし，プロセスを READY 状態にする．chan が 0 の時は，すべての休眠識別子を意味する．
7	dispatch(pid)	プロセス pid を走行させる．
8	control_execution(execid, mips)	execid で指定した演算の程度を mips に変更する．

の範囲で生成できる．また，性能の総計が，100%を越えるような性能調整の演算の性能の変更はできない．

プログラムの実行速度の調整は，ある単位時間（これをタイムスロットと名付ける）で，プログラムの実行と停止を繰り返すことで実現できる．一定の連続したタイムスロットをタイムブロックと名付ける．性能調整の演算の生成したとき，または性能を変更したときに，タイムブロック中で，要求された実行速度に見合うタイムスロットを演算に割り当てる．次に，そのタイムスロットの時間分だけ，その演算に関連付けられたプロセスを走行させる．この結果，プログラムの実行速度を，要求された速度に調整できる．タイムスロットの長さは固定長であり，その長さは OS のコンパイル時に決定する．

要求された実行性能に見合うタイムスロットを，タイムブロックの中から選択する方式として，これまでに我々は四つの方式を提案し，評価した[6]．この結果，処理の均一性が最も良い改良疑似周期割当方式を採用した．処理の均一性とは，プログラムの実行速度を調整したときの処理の滑らかさを表す尺度である．改良疑似周期割当方式は，要求性能に基づき n 個のタイムスロットを演算に割り当てる場合，i 番目に割り当てるタイムスロットの位置を，（総タイムスロット数）×（$i-1$）／ n とする方式である．この方式では，一番最初に割り当てるタイムスロット位置が必ず 0（タイムブロックの先頭のタイムスロット）になる．このため，性能調整の演算を 2 個以上生成する場合に，必ず最初のタイムスロットの割り当て位置の衝突が起こる．これを回避するために，タイムブロックの先頭から最初の空きタイムスロットを検索し，その位置を起点として，タイムスロットを割り当てる．これにより，最初に割り当てるタイムスロットの位置が衝突する問題を解決できる．また，同じ個数のタイムスロットを割り当てられる演算を複数生成したときにも，タイムスロットの位置が衝突する問題を解決できる．

また，複数のプロセスの実行速度を同時に調整する場合には，プロセス間でタイムスロットの割当位置が衝突する問題がある．この問題を解決するために，衝突位置から最も近い空きタイムスロットを検

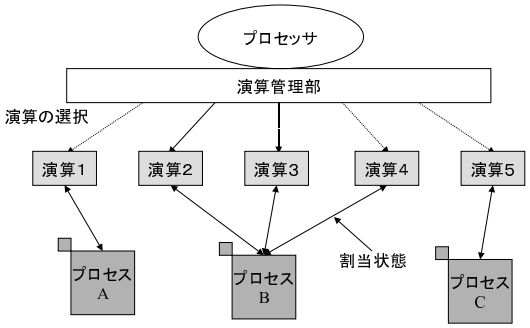

図 1 演算とプロセスの対応関係

索し，代りに割り当てる方式を採用した[3]．

3.2.2 優先度の演算

優先度の演算は，生成されたときに，スケジューリングの優先順位となる優先度の値を持つ．優先度の演算には，二つの状態がある．一つは，プロセスと関連付けられていない状態で，優先度キューにはつなげない状態で存在する．もう一つは，優先度の演算がプロセスに関連付けられている状態で，優先度キューにつなげられ，スケジューリング対象となる．

3.3 演算とプロセスの関係

演算とプロセスの関係を図 1に示す．図 1に示したように，演算とプロセスの関連付けの関係は，$1:n$（n は任意の正の整数）である．つまり，一つのプロセスに対し，複数の演算を関連付けることができる．また，一つのプロセスに対し，異なる種類の演算を関連付けることができる．各プロセスは，関連付けられた演算が持つ演算の程度の合計にしたがって，スケジューリングされる．(スケジューリング方式の詳細ついては，次の節で述べる）スケジューラ（演算管理部）は，走行させる演算を選択する．次に，選択した演算に関連付けられたプロセスを走行させる．

プロセスが複数の演算を持つ仕組みについて述べる．プロセスと演算の関連付けの対応関係を，資源「演算」の管理表に保持する．この管理表には，各資源「演算」毎に，演算識別子，演算の程度，関連付けられたプロセスの資源識別子などの情報が格納されている．したがって，演算とプロセスの関連付けと関連付けの解除は，管理表の当該エントリの更新を行うだけで実現できる．

3.4 スケジューリング方式

性能調整の演算を用いたプログラムの実行速度を調整するスケジューリング方式と，優先度によるスケジューリング方式を共存させた方式について述べる．プログラムの実行速度を高い精度で，滑らかに，調整するために，プログラムの実行速度の調整を優先する．具体的には，タイムスロットが，走行可能なプロセスが関連付けられた演算に割り当てられているとき，必ずそのプロセスを走行させる．優先度の演算を関連付けられたプロセスは，プログラムの実行速度を調整するプロセスが走行する以外の時間で，優先度に応じて走行できる．

スケジューラは，タイムスロットの境界でタイマ割り込みにより呼び出される．そして，次のタイムスロットが，走行可能なプロセスに関連付けられた演算に，割り当てられているかどうかを調べる．走行可能なプロセスに割り当てられていた場合，そのプロセスを走行させる．それ以外の場合，優先度により走行させるプロセスを選択する．実行速度を調整されているプロセスが，タイムスロットを使い切る前に WAIT 状態になった場合には，そのタイムスロットの残りの時間を利用するプロセスを優先度により選択する．

3.5 特徴

資源「演算」には，以下の特徴がある[1]．

(1) プロセス生成の高速化
(2) プロセスの処理途中での停止や再開
(3) データ部を初期化することによる再起動

さらに，プロセスに複数の演算を関連付けには，以下の特徴がある[4]．

(1) プロセスに複数の性能調整の演算を関連付けることによるユーザレベルでのスケジューリング
(2) プロセスに複数の優先度の演算を関連付けることよるプロセス走行時間の増加
(3) 性能調整の演算と優先度の演算を同時に関連付けることによる処理時間の保証

4 プロセスグループの実行速度調整法

4.1 要求条件と機能

4.1.1 要求条件

資源「演算」を利用したプロセスグループの実行速度調整機能の要求条件を以下に挙げる．

(1) プロセスグループを単位として，性能調整ができる．
(2) プロセスグループを構成するプロセス毎の性能調整ができる．
(3) プロセスグループの演算の程度を自由に設定できる．

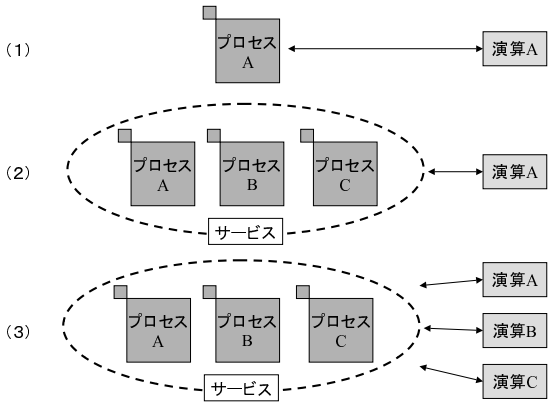

図 2 演算とプロセスグループの関係

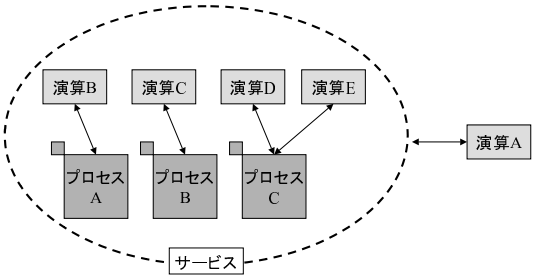

図 3 演算とプロセスグループ内のプロセスの関係

4.1.2 機能

＜プロセスグループ単位での演算の程度の指定＞

プロセスグループの実行速度調整法では，プロセスと同様にプロセスグループを扱う．つまり，プロセスグループの演算の程度を表す演算を導入する．その様子を図 2に示す．図 2(1) にあるように，プロセスには，一つ以上の演算を関連付けることができる．図 2(2) と (3) のように，プロセスと同様に，プロセスグループにも 1 つ以上の演算を関連付けできるようにする．これにより，プロセスグループ単位での実行速度の調整や，処理時間の保証が可能となる．

プロセスグループを構成するプロセスは，それぞれ役割や重要性が異なり，個別に演算の程度を指定したいという要求がある．このため，図 3のように，個々のプロセスに，演算を関連付けられる様にする．

4.2 実現方式

4.2.1 演算の多階層化

Tenderでは，資源「演算」を利用して，スケジューリングを行う．また，スケジューラは，資源「演算」の管理部に存在する．このため，スケジューリングに関する情報を，演算管理部が持つ必要がある．プロセスグループを実現するとき，プロセスグループとプロセスグループを構成するプロセスの関係は，スケジューリングに必要な情報である．このため，プロセスグループとプロセスグループを構成するプロセスの関係を，資源「演算」を用いて表現することにより，演算管理部にスケジューリングに必要な情報を持つことができる．したがって，演算の木構造で，プロセスグループを表現することとした．

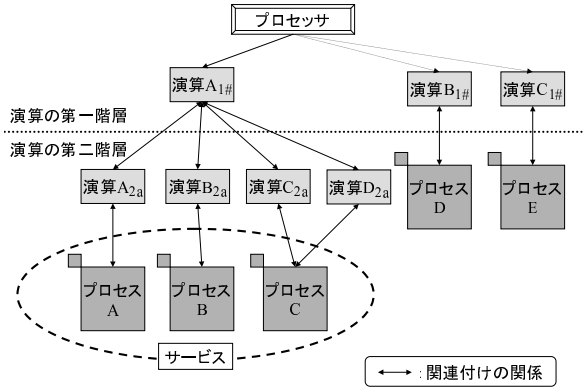

図 4 演算によるプロセスグループの実現

図 3を，演算の木構造で表現した例を図 4に示す．図 3の演算 A のように，プロセスグループの演算の程度を持つ演算をディレクトリ演算と名付ける．ディレクトリ演算は，演算の木構造で子を持つノードとなる．また，図 4の演算 A_{2a}のように，プロセスグループ内のプロセスに関連付けられた演算を，リーフ演算と名付ける．図 4のように，ディレクトリ演算とリーフ演算を関連付け，木構造を作ることで，プロセスグループを表現する．この場合は，二階層の演算で表現できる．

リーフ演算は，関連付けられているディレクトリ演算に割り当てられたプロセッサ処理時間を分割し，利用する．このとき，一つのディレクトリの演算に関連付けられた性能調整のリーフ演算の性能の合計は，100%以下とする．

図 4を用いて，プロセスグループの演算の木構造を構築する手順について述べる．最初に，プロセスグループの演算として，演算 $A_{1\#}$を生成する．次に，演算 A_{2a}，B_{2a}，C_{2a}，および D_{2a}を生成する．それから，演算 $A_{1\#}$と，演算 A_{2a}，B_{2a}，C_{2a}，および D_{2a}を関連付けることにより，プロセスグループの木構造を構築する．最後に，プロセス A，B，および C と，演算 A_{2a}，B_{2a}，C_{2a}，および D_{2a}を関連付ける．この結果，演算 $A_{1\#}$の演算の程度によって割り当てられたプロセッサの処理時間を，演算 A_{2a}，B_{2a}，C_{2a}，および D_{2a}で分割し，利用する．このとき，以下の

式が成り立つ．

$$(\text{演算 } A_{1\#}\text{の割当時間}) = \sum_{n=A}^{D} (\text{演算 } n_{2a}\text{の割当時間})$$

（ただし，プロセス A，B，および C が走行可能な状態でないとき，別の演算にプロセッサの処理時間が割り当てられる）

ここで，リーフ演算を各プロセスに演算を関連付けるのは，プロセスグループ内でのスケジューリング順位を決定するためである．したがって，リーフ演算が持つ演算の程度は，プロセスグループ内で有効な値とする．例えば，演算 $A_{1\#}$ の演算の程度が 10%で，演算 A_{2a} の演算の程度が 10%であるとする．このとき，演算 A_{2a} は，演算 $A1\#$ の 10%の処理時間が割り当てられる．したがって，プロセッサの処理時間のうち 10%× 10% = 1%が割り当てられる．

サービスの処理内容に合わせた演算の木構造を構築するために，演算の多階層化を実現する．演算の多階層化のためには，プロセスグループ内に，プロセスグループを作ることを実現する必要がある．これは，ディレクトリ演算に，ディレクトリ演算を関連付けることで可能となる．この結果，プロセスグループ内に，再帰的にプロセスグループを作成でき，n 階層（n は任意の正の整数）のプロセスグループを構築することができる．n 階層の演算の木構造の例を図 5に示す．図 5では，プロセスグループ内に，プロセスグループを作ることにより，n 階層の演算 A_{na} を実現している．

プロセッサに近い階層から順番に，演算の第一階層，第二階層，…，第 i 階層と名付ける．また，説明のために，第 $i-1$ 階層の演算 A に関連付けられた第 i 階層の演算 A を，演算 A_{ia} と表すと，プロセッサ割当時間は，以下の一般式で表される．

$$(\text{演算 } A_{i-1x}\text{の割当時間}) = \sum_{n=A}^{X} (\text{演算 } n_{ia}\text{の割当時間})$$

（x，X は任意のアルファベットを表す）

ただし，演算 A から演算 X に関連付けられたプロセスがすべて走行可能でないとき，演算 A_{i-1x} に割り当てられる割当時間は放棄され，他の演算に与えられる．

＜プロセスグループへの複数演算の割当＞

プロセスと同様に，プロセスグループの処理時間を保証するときには，プロセスグループのために，複数のプロセスグループの演算を関連付けする必要がある．その様子を図 6に示す．図 6では，ディレクトリ演算 $A_{1\#}$ と，リーフ演算 A_{2a}，B_{2a}，および C_{2a} を生成している．新たに，プロセスグループの演算として，ディレクトリ演算 $B_{1\#}$ を生成したときの処理内容を説明する．演算の関係は木構造であるため，演算 $B_{1\#}$ と，演算 A_{2a}，B_{2a}，および C_{2a} を関連付けることができない．このため，新たに演算 D_{2b}，E_{2b}，および F_{2b} を生成する．それから，プロセス A，B に関連付けする．このようにして，プロセスグループに，2 個以上の演算を確保できる．このとき，演算 F_{2a} と，プロセス D のようなサービスを構成しないプロセスを関連付けることができる．

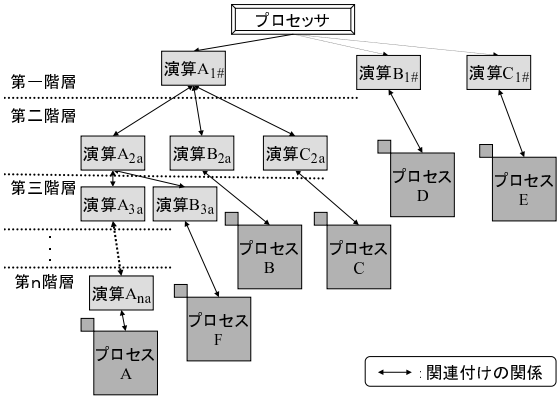

図 5 演算の多階層化（n 階層の例）

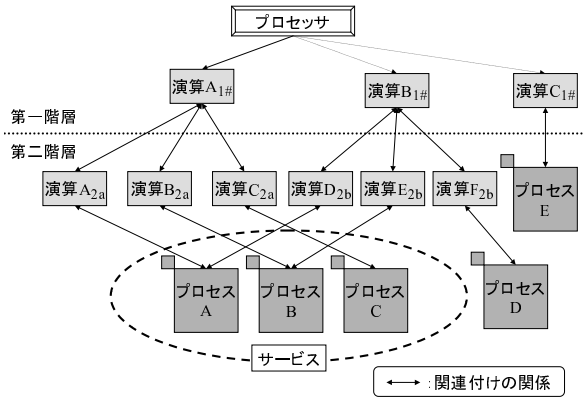

図 6 演算の階層化（処理時間の保証）

4.2.2 提供インタフェース

演算の階層化を実現するために，表 1の 3 と 4 の二つのインタフェースを拡張した．拡張したインタフェースを表 2に示す．表 1のインタフェース 3 と 4 に，演算同士の関連付けと関連付けの解除の機能を追加した．また，演算が生成されたときは，第一階層に属するものとする．演算を他の階層に移動させるときには，他の演算に関連付けする．このとき，演算の持つ演算の程度 mips の値は不変である．しかし，演算の程度 mips は，移動先の階層内でのみ有効

表 2 演算管理インタフェース（演算の階層化対応）

通番	形式	機能
3’	attach_execution(execid, rid)	rid は，プロセス識別子か，演算識別子である．(1) rid がプロセス識別子のとき，演算 execid をプロセス rid に関連付ける．ただし，演算 execid とその下の階層の演算が関連付けられているときは，エラーとする．(2) rid が演算識別子のとき，演算 rid の下の階層として，演算 execid を関連付ける．ただし，関連付けされる演算の階層で，性能 mips が確保できないときエラーとする．
4’	detach_execution(execid, rid)	rid は，プロセス識別子か，演算識別子である．(1) rid がプロセス識別子のとき，演算 execid とプロセス pid の関連付けを解除する．(2) rid が演算識別子のとき，演算 execid と演算 rid との関連付けは解除され，演算 rid は，第一階層に移動される．ただし，第一階層で，性能 mips が確保できないときエラーとする．

である．また，他の演算との関連付けを解除された演算は，第一階層に移動する．

表 2のインタフェースを用い，プロセスグループの実行速度を調整する手順を示す．プロセスの生成時には，以下の処理を行う．

(1) プロセスグループの演算が必要であれば，インタフェース 1 を用いて，プロセスグループの演算を生成する．
(2) プロセスグループ内のプロセスを生成する．
(3) インタフェース 1 を用いて，新たに演算を生成する．
(4) インタフェース 3’ を用いて，(3) で生成した演算をプロセスグループの演算に関連付ける．
(5) インタフェース 3’ を用いて，(3) で生成した演算とプロセスを関連付ける．
(6) 必要に応じて，(2) から (5) を繰り返す．

プロセスの処理終了時は，以下の処理を行う．

(1) プロセスの処理終了後，インタフェース 4’ を用いて，プロセスと演算の関連付けを解除する．
(2) インタフェース 4’ を用いて，プロセスグループの演算とプロセスに関連付けられていた演算の関連付けを解除する．
(3) インタフェース 2 を用いて，演算を削除する．
(4) プロセスを削除する．
(5) 必要に応じて，(1) から (4) を繰り返す．

4.2.3 スケジューリング方式

スケジューリング方式について，図 4を用いて述べる．スケジューラは，タイムスロット境界，またはタイムスライス境界で，次に走行させる演算を検索する．スケジューラは，第一階層から，演算を一つ選択する．リーフ演算 $B_{1\#}$ か $C_{1\#}$ を選択したときは，プロセス D か E を走行させる．ディレクトリ演算 $A_{1\#}$ を選択したとき，次の階層の演算を選択する．このように，ディレクトリ演算を選択したときは，リーフ演算を選択まで，検索する．ただし，選択したリーフ演算に関連付けられたプロセスが走行可能でない場合，次の候補の演算を選択する．例えば，図 4では，プロセス A から C が走行可能な状態でないときは，プロセス D か E を走行させることになる．

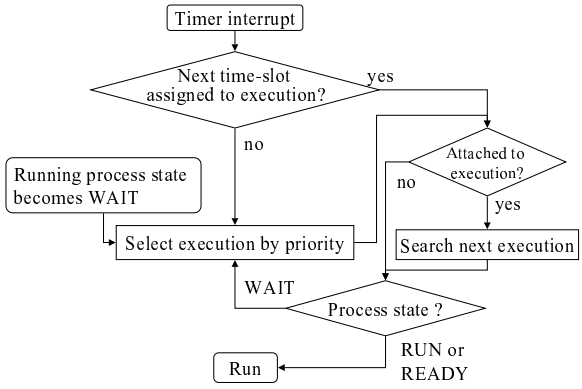

図 7 プロセスの切り替え処理（演算の階層化対応）

この処理の流れを図 7に示す．第一階層から，次のタイムスロットが演算に割り当てられているかを調べる．割り当てられてないとき，優先度に基づき演算を選択する．割り当てられているとき，その演算に他の演算が関連付けられているかを調べる．他の演算が関連付けられているとき，この演算はプロ

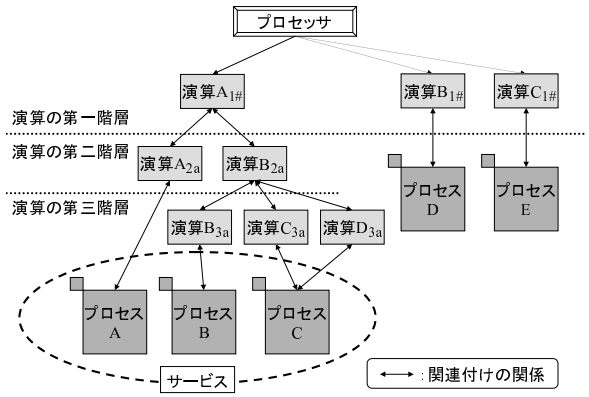

図 8 演算の多階層化の例

セスグループの演算であるので，次の階層に移動し，同様の処理をする．走行可能なプロセスに関連付けられた演算を見つけるまで，この処理を繰り返す．

4.3 期待される効果

演算の階層化を実現することで，以下の三つの効果がある．

(1) サービスの実行性能調整
特定のサービスの実行性能を調整できる．これによって，サービスを構成する複数のプロセスの処理速度を一様に調整できる．例えば，利用者は，サービスが複数のプロセスから構成されていることを意識ことなく，一つの尺度でサービス処理速度を指定できる．また，サービス毎に，任意の実行速度を割り当てられる．

(2) サービスの処理時間の保証
複数プロセスからなるサービスの処理時間を保証できる．これによって，Web サーバなどのサービス単位で，他のサービスの存在に関係なく，処理時間を保証できる．

(3) サービスの処理内容に合わせた自由なプロセッサ割当の実現
プロセスグループを二階層の演算として構築できるだけでなく，三階層以上の階層として，構築できる．これにより，利用者はサービスの処理内容に合わせて，自由にスケジューリングポリシを決定できる．例えば，図 4では，プロセス A を優先的に走行させるために，演算を三階層にしている．

5 まとめ

本論文では，プログラム実行速度調整機能を実現した ***Tender*** オペレーティングシステムの概要を紹介し，資源「演算」を利用したプログラム実行速度調整方式とサービス処理時間の保証方式について述べた．さらに，本論文で提案したプロセスグループの実行速度調整方式について述べた．具体的には，プロセスグループ単位で演算の程度を指定するために，プロセスグループの演算を導入した．プロセスグループの演算と，プロセスグループを構成するプロセスに関連付けられた演算を木構造で表現した．これにより，プロセスと同様に，プロセスグループに，複数の演算を確保できる．また，プロセスグループ内のプロセスにも，複数の演算を関連付けできる．さらに，プロセスグループ内に，プロセスグループを作れることとし，演算を多階層化した．これにより，プロセスグループ単位での性能調整，プロセスグループを構成するプロセス間の性能調整，およびプロセスグループとプロセスグループを構成するプロセスに自由な演算の程度の指定が可能となる．この結果，プロセスグループの実行速度調整と，処理時間の保証が可能となる．さらに，プロセスグループの多階層化を実現したことで，サービスの形態に合わせ，利用者が自由にスケジューリングポリシを決定できる．

残された課題として，複数プロセスから構成されるサービスの実行速度調整機能の調整精度の評価がある．

参考文献

[1] 田端利宏, 谷口秀夫: ***Tender*** オペレーティングシステムの資源「演算」によるプログラム実行速度調整機能の実現と評価, 情報処理学会論文誌, Vol.40, No.6, pp.2523–2533 (1999).

[2] 谷口秀夫: プロセススケジュールの制御によるプログラムの実行速度調整法の評価, 電子情報通信学会論文誌, Vol.J83-D-I, No.1, pp.184–193 (2000).

[3] Tabata, T., Taniguchi, H., Ushijima, K.: Implementation and Evaluation of Multiple Processes Control Mechanism for Regulating Program Execution Speed, Proceedings of International Symposium on Principles of Software Evolution (ISPSE 2000), pp.315–319 (2000).

[4] 田端利宏, 谷口秀夫: ***Tender*** オペレーティングシステムにおける資源「演算」を用いたサービス処理時間の保証, 情報処理学会論文誌, Vol.41, No.6, pp.1745–1754 (2000).

[5] 谷口秀夫, 青木義則, 後藤真孝, 村上大介, 田端利宏: 資源の独立化機構による ***Tender*** オペレーティングシステム, 情報処理学会論文誌, Vol.41, No.12, pp.3363–3374 (2000).

[6] 田端利宏, 谷口秀夫, 牛島和夫: プログラム実行速度調整機能における処理の均一性向上手法, 情報処理学会第 59 回全国大会 講演論文集（分冊 1）, pp.37–38 (1999).