Vol.2011-OS-116 No.4
2011/1/24

Tender オペレーティングシステムにおける資源「入出力」の提案

長尾 尚[†1] 一井晴那[†2]
山内利宏[†1] 谷口秀夫[†1]

プログラムは，プロセッサ処理と入出力処理を繰り返しながら処理を進める．***Tender*** オペレーティングシステムでは，プロセッサ処理時間の割当単位を資源「演算」で表し，資源「演算」の関係を木構造で管理することにより，プロセスやプロセスグループ単位のプロセッサ性能の調整機能を実現している．ここでは，入出力処理時間の調整単位を表す資源「入出力」を導入し，資源「演算」と同様に木構造で管理することにより，プロセスやプロセスグループ単位の入出力性能の調整機能を提案する．

Proposal of *I/O* Resource on *Tender* Operating System

Takashi Nagao ,[†1] Haruna Ichii ,[†2]
Toshihiro Yamauchi [†1] and Hideo Taniguchi [†1]

A program executes CPU processing and I/O processig. We proposed a mechanism for regulating the processor performance of process and process groups on ***Tender*** operating system. Proposed mechanism manages relations between ***Execution*** resources that encapsulate a unit of CPU usage time in hierarchy structure. In this paper, we propose a mechanism for regulating the I/O performance of process and process groups. This mechanism manages relations between ***I/O*** resources that encapsulate a unit of I/O processing time in hierarchy structure as well as previously proposed mechanism.

†1 岡山大学大学院自然科学研究科
Graduate School of Natural Science and Technology, Okayama University
†2 岡山大学工学部
Faculty of Engineering, Okayama University

1. はじめに

オペレーティングシステム（以降，OS と略す）は，計算機上で複数のプロセスを独立あるいは協調して走行させるため，プロセスの走行制御を行う．例えば，1 台の計算機上で複数のサービスを実行する場合，全てのサービスの重要性が同一であれば，各サービスを構成するプロセスを均等に実行する．また，時間制約を持つサービスを実行するのであれば，当該サービスを構成するプロセスを優先して実行し，走行保証を行う．ネットワークを介したサービス妨害攻撃には，攻撃対象になり得るサービスを構成するプロセスについて，使用資源量を制限することにより，サービス妨害攻撃の影響を抑制する．

プロセスの走行制御を行うために，OS が制御する資源には，プロセッサ量，メモリ量，ファイル量，および通信量があり，各資源について制御手法が研究されている．プロセッサ量の制御に関しては，指定されたデッドライン内に処理を実行することにより，実時間性を保証する研究[1)]，プログラム実行の周期や実行時間の比率をもとにしてプロセッサ割り当てを行う研究[2),3)] がある．メモリ量の制御に関しては，ワーキングセットにより，プロセスに対してスワップアウトされないメモリページを提供する研究[4),5)] がある．ファイル量の制御に関しては，Linux の資源制限機能である rlimit では，プロセスが作成できるファイルサイズを制限できる．通信量の制御に関しては，資源を優先して使用できる権利を QOS チケットとして表現し，QoS 制御を行う研究[6)] や無線ネットワークにおいて，帯域予約による QoS 制御を行う研究[7)] がある．

メモリや記憶装置は，価格低下が著しく，大容量化している．一方，プロセッサ性能は向上しているものの，サービスは高度化し，サービスを構成するプロセスは多くの複雑な処理を行うようになっている．また，1 台の計算機で走行するサービス数も増加している．このため，プロセッサ量の制御に関する研究が盛んに行われている．入出力性能は，プロセッサ性能に比べ向上していないため，入出力量の制御に関する手法が必要である．

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．これまでに，プロセッサ性能の調整法[8)] と入出力性能の調整法[9)] を提案した．また，***Tender*** オペレーティングシステム[10)] において，資源「演算」を利用したプロセッサ性能の調整機能[11)] を提案した．しかし，プロセッサ性能の調整のみでは，入出力時間の長大化を抑制できないため，プログラム実行速度の調整に限界がある．そこで，***Tender*** の入出力性能の調整機能として，資源「入出力」を利用した入出力性能の調整機能を提案する．

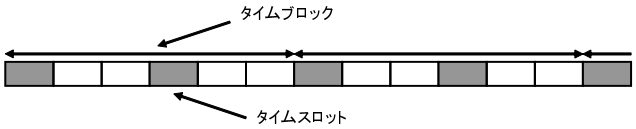

図 1 タイムスロットとタイムブロックの関係

2. 資源「演算」と演算木

2.1 *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．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（資源管理処理部と呼ぶ）を資源ごとに分離し，共有プログラムを排除している．また，各震源の管理情報も資源ごとに分離し，各資源の管理表の間の参照関係を禁止している．

以上の資源の分離と独立化により，資源の事前生成や保留による資源の作成や削除を伴う処理の高速化を実現している．また，OS の動作および内部情報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．

2.2 資源「演算」

演算[12] は，***Tender*** の資源であり，プロセッサを割り当てられる程度（以降，演算の程度と名付ける）を持つ．演算には，性能調整演算と優先度演算，および上限性能付き優先度演算がある．

性能調整演算が持つ演算の程度は，プロセスに割り当てられるプロセッサ性能 (1〜100 %，他プロセスが走行せず，ハードウェアを占有して走行したときの性能を 100 %として 1 %単位で指定) を示す．ただし，性能調整演算が持つ演算の程度の総計は 100 %を超えることはできない．プロセッサ性能の調整は，タイムスロットと名付けた単位時間で，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．タイムスロットとタイムブロックの関係を図 1 に示す．性能調整演算は，一つのタイムブロックの中から，演算の程度に見合う割合 (%) のタイムスロットを割り当てられる．割り当てられたタイムスロットの分だけ，性能調整演算に関連付けられたプロセスを走行させることで，プロセッサ処理速度を調整する．

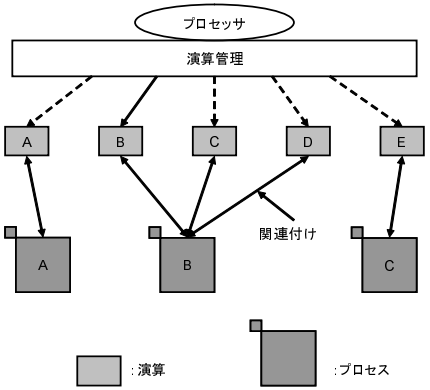

図 2 演算とプロセスの関係

優先度演算が持つ演算の程度は，スケジューリングの優先順位となる優先度を示す．ここで，優先度は固定であり，動的に変更されることはない．

上限性能付き優先度演算が持つ演算の程度は，上限性能と優先度を示す．上限性能とは，単位時間当たりに使用できるプロセッサ時間の最大割合であり，その範囲内で優先度に応じてプロセッサ時間を割り当てる．指定された最大割合に達すると，当該プロセスはその単位時間が終了するまで強制的にプロセッサ時間の割り当てを停止される．

プロセスを走行させるには，演算とプロセスを関連付ける必要がある．演算とプロセスを関連付けることにより，プロセスは，自身に関連付けられた演算に割り当てられたプロセッサ時間を利用して，走行する．演算を関連付けられていないプロセスは，プロセッサ時間を割り当てられない．

演算とプロセスの関係を図 2 に示す．図 2 に示すように，演算とプロセスの関連付けの関係は，$n:1$（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，同時に異なる種類の演算を関連付けることができる．このとき，プロセスは，自身に関連付けられた演算が持つ演算の程度の合計だけプロセッサを割り当てられる．

スケジューラは，タイムスロット境界でタイマ割り込みにより呼び出され，走行すべきプロセスを選択する．プロセッサ処理の実行速度を調整するために，性能調整演算に優先してプロセッサ時間を割り当てる．具体的には，次のタイムスロットが演算に割り当てられており，その演算が走行可能なプロセスに関連付けられている場合，必ずそのプロセスを走行さ

せる．そうでない場合，優先度演算または上限性能付き優先度演算を関連付けられたプロセスを優先度と上限性能に応じて走行させる．

2.3 演 算 木

演算を木構造（以降，演算木と名付ける）で管理することにより，複数のプロセスから構成されるサービスの実行性能を調整する．演算木の根はプロセッサそのものの性能を表し，演算木の各頂点は演算を表している．演算木の根をルート演算，子を持つ演算をディレクトリ演算と呼び，子を持たない演算をリーフ演算と呼ぶ．

演算木でのプロセッサ時間の割り当ては，直下の演算が持つ演算の程度に応じて，ルート演算およびディレクトリ演算のプロセッサ時間を直下の演算に分割して割り当てる．したがって，ルート演算の場合，直下の演算が持つ演算の程度は，プロセッサそのものの性能に対する演算の程度となる．ディレクトリ演算の場合，直下の演算が持つ演算の程度は，当該ディレクトリ演算に割り当てられたプロセッサ時間を 100 %とみなしたときの割合になる．サービスを構成するプロセス群をプロセスグループとしたとき，ディレクトリ演算はプロセスグループの演算を表し，リーフ演算は各プロセスの演算を表す．

演算木の例を図 3 に示す．図 3 では，サービス A は 4 つのプロセスで構成され，サービス A のサブプロセスグループとなるサービス A ’を持つ．また，サービス B とサービス C はそれぞれ一つのプロセスで構成される．サービス A のプロセスグループに対する演算は性能調整演算 exec1 であり，演算の程度は 50 %である．これより，サービス A はプロセッサそのものの性能の 50 %を割り当てられることとなる．また，サービス A ’のプロセスグループに対する演算は性能調整演算 exec1-1 であり，演算の程度は 40 %である．これにより，サービス A ’は性能調整演算 exec1 の 40 %の性能を割り当てられることとなり，これはプロセッサそのものの性能の 20 %（= 50 %× 40 %）である．

2.4 特徴と問題点

資源「演算」により，以下のことが可能となる．

(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) データ部の初期化によるプロセス再起動

また，一つのプロセスに複数の演算を関連付ける機能による特徴を以下に示す．

(1) 複数の優先度演算を関連付けることにより，優先度を上げることなくプロセスの走行時間を増やすことできる．
(2) 性能調整演算と優先度演算を同時に関連付けることにより，性能調整演算でプロセッサ時間の割当を保証でき，プロセッサに空きがあれば，優先度演算によりプロセッサの空き時間を利用して処理を行える．

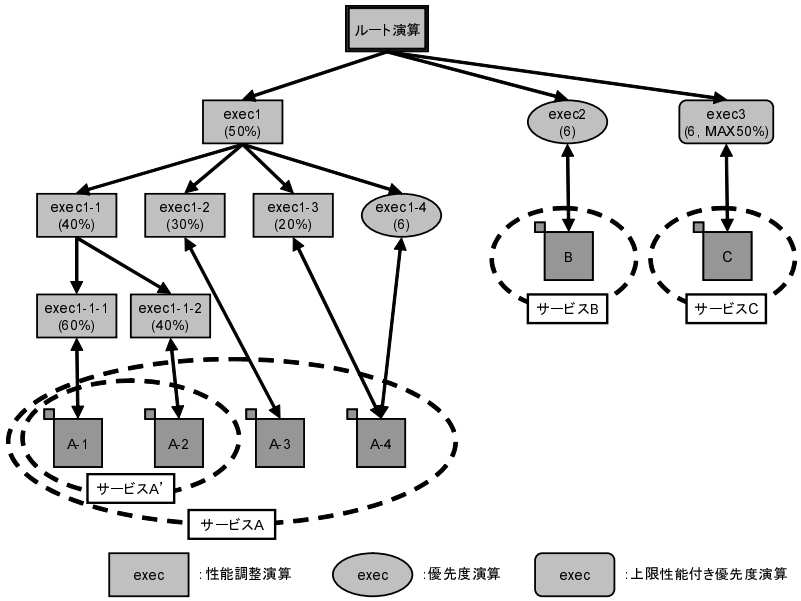

図 3 演算木によるプロセスグループの表現

さらに，演算木による特徴を以下に示す．

(1) プロセスグループのプロセッサ性能の調整

プロセスグループに対応するディレクトリ演算に性能調整演算または上限性能付き優先度演算を関連付け，演算の程度または上限性能を変更することにより，プロセスグループを構成するすべてのプロセスのプロセッサ性能を一度に調整できる．例えば，利用者はサービスを構成するプロセス数を意識することなく，ディレクトリ演算が持つ演算の程度を変更することで，サービス全体のプロセッサ性能を調整できる．

(2) サービスの実行速度制限

プロセスグループに対するディレクトリ演算に性能調整演算または上限性能付き優先度演算を関連付けることにより，複数のプロセスにより構成されるサービスの実行速度を制限できる．

しかし，プロセッサ性能の調整のみでは，入出力処理の影響を受けてしまうため，サービス実行速度の調整に限界がある．プロセッサ性能の調整の問題を以下に示す．

（問題 1）入出力時間の長大化によるサービス実行速度の低下

プロセッサ性能の調整により，プロセスのプロセッサ時間を保証しても，他プロセスの入出力要求により実行される入出力デバイスの I/O 処理（以降，実 I/O 処理と名付ける）の終了待ちが発生し，入出力時間が長大化する．また，当該プロセスが終了待ちの間，割り当てられたプロセッサ時間を利用して処理を行えない．したがって，プロセッサ時間を保証したプロセスであっても，実 I/O 処理の終了待ちにより，サービス実行速度が低下する．

（問題 2）入出力性能を制限できないことによる調整精度の低下

プロセッサ性能の調整により，プロセスのプロセッサ時間を制限しても，プロセスの入出力時間を制限できない．このため，他プロセスが入出力要求を発行していないとき，当該プロセスが入出力要求を発行すると，入出力デバイスの性能そのもので入出力処理が実行されるため，サービス実行速度の調整精度が低下する．

3. 資源「入出力」と入出力木

3.1 プロセッサ処理と入出力処理の違い

プロセッサ処理は，スケジューラにより，割り当てられるプロセッサ時間を利用してプロセスが走行することにより実行される．このため，スケジューラは，プロセスへのプロセッサ割当量を制御することにより，プロセスのプロセッサ性能を制御できる．一方，入出力処理は，プロセスが発行した入出力要求を入出力デバイスが実行することにより実行される．プロセッサ処理と異なり，実 I/O 処理は中断できないため，入出力デバイスの処理を時間で分割し，プロセスへ割り当てることはできない．そこで，入出力要求数を調整する制御法[13] を利用し，プロセスの入出力時間を調整する．具体的には，利用者の要求する入出力性能（以降，要求入出力性能と名付ける）から理想の入出力時間を算出し，プロセスの入出力時間を理想の入出力時間に調整する．理想の入出力時間の算出式を以下に示す．

$$T_S = \frac{100}{\text{要求入出力性能}} \times \text{実 I/O 時間} \quad (1)$$

入出力要求を調整する制御法は，デバイスドライバへの実 I/O 要求数（実 I/O 処理を行うプロセス数）を制御することにより，要求入出力性能から算出する理想の入出力時間内にプロセスの実 I/O 処理を終了する．また，実 I/O 処理終了後に，遅延処理を行うことにより，プロセスの入出力時間を理想の入出力時間に調整する．

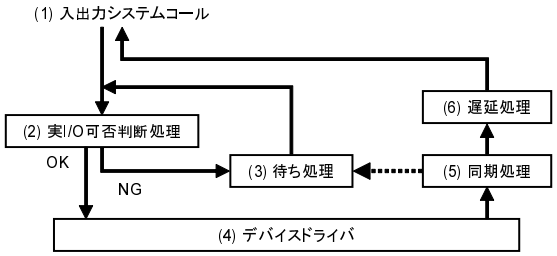

図 4 入出力要求数を調整する制御法

3.2 資源「入出力」

Tender の資源として入出力を定義し，***Tender*** における入出力性能の調整機能を実現する．入出力は，入出力処理に要する時間の程度（以降，入出力の程度と名付ける）を持つ．入出力要求を調整する制御法は，入出力の程度に基づき，プロセスの入出力時間または入出力要求の実行順序を決定する．入出力には，性能調整入出力と優先度入出力がある．ただし，優先度入出力は，入出力要求の実行順序を決定するため，上限性能を設定できない．したがって，上限性能付き優先度入出力は存在しない．

性能調整入出力が持つ入出力の程度は，要求入出力性能 (1〜100 %，他プロセスが走行せず，ハードウェアを占有して走行したときの性能を 100 %として 1 %単位で指定) を示す．

優先度入出力が持つ入出力の程度は，入出力要求の実行の優先順位となる優先度を示す．

入出力要求数を調整する制御法の基本方式を図 4 に示し，以下に説明する．

（1）プロセスは入出力システムコールを発行する．

（2）許容値に基づき，実 I/O 処理を許可するか否か判断する．現在のデバイスドライバへの実 I/O 要求数が許容値以上の場合，実 I/O 処理を許可せず，(3）の処理を行う．そうでない場合，(4）の処理を行う．許容値の算出式を以下に示す．

$$\text{許容値} = MAX(1, \frac{100}{\sum_{i=1}^{k} P_i} - 1) \quad (2)$$

ここで，P_i とは，実 I/O 要求を行っていないプロセスの中で上位 i 番目の要求入出力性能，k とは，許容係数である．調整しないプロセスが入出力処理を実行できなくなることを防ぐため，許容値は少なくとも 1 である．また，許容係数 k は，許容値の算出において，考慮するプロセス数であり，利用者が求められる調整精度の高さにあわせて決定する．

（3）入出力優先度に基づき，プロセスを待ちキューにつないで待ち状態にする．性能調整

入出力，優先度入出力の順に高入出力優先度とし，性能調整入出力間では入出力処理を実行する性能が高いもの，優先度入出力間では優先度が高いものを高入出力優先度とする．また，同入出力優先度は LRU で管理する．同期処理からの同期により，入出力優先度が最も高いプロセスを起床する．

（4）実 I/O 処理を行う．

（5）待ち処理に同期を送信する．

（6）要求入出力性能に基づき，遅延処理を実施する．遅延時間は，理想の入出力時間から入出力処理に要した時間 $(T_1 - T_0)$ を減算した値である．遅延時間 T_S の算出式を以下に示す．

$$T_S = \frac{100}{\text{要求入出力性能}} \times \text{実 I/O 時間} - (T_1 - T_0) \quad (3)$$

ここで，実 I/O 時間とは，実 I/O 処理に要する時間である．

プロセスが入出力処理を行うには，入出力とプロセスを関連付ける必要がある．入出力とプロセスを関連付けることにより，入出力から算出する入出力優先度を利用して，当該入出力に関連付けられたプロセスが発行する入出力要求の実行順序を決定する．入出力を関連付けられていないプロセスは，入出力要求の実行順序を決定できないため，入出力処理を実行できない．入出力管理の提供する機能を以下に示す．

（機能 1）入出力の生成
（機能 2）入出力の削除
（機能 3）入出力に対する入出力の関連付け
（機能 4）入出力に対する入出力の関連付けの解除
（機能 5）入出力に対するプロセスの関連付け
（機能 6）入出力に対するプロセスの関連付けの解除
（機能 7）入出力の程度の変更
（機能 8）入出力要求の受付

利用者にとって，入出力は，UNIX 系 OS におけるファイルディスクリプタに相当する．このため，入出力処理を行うプロセスは，入出力を生成し，生成した入出力へ入出力要求を発行する．ただし，ファイルディスクリプタはアクセス先のファイルを表現するのに対し，入出力はアクセス先の入出力デバイスと入出力性能を表現する．具体的な入出力の利用手順を以下に示す．利用者は，最初に入出力を生成し，利用したい入出力の程度を設定する．次に生成した入出力を使用したい入出力デバイスに対応する入出力木（詳細は 3.3 節で述べる）に関連付け，入出力処理を行うプロセスを生成した入出力に関連付ける．プロセスは，自身に関連付けられた入出力へ入出力要求を発行し，入出力処理を行う．このとき，プロセスの入出力時間は，当該プロセスに関連付けられた入出力が持つ入出力の程度にあわせて調整される．入出力デバイスの使用を停止する場合，利用者は，入出力とプロセスの関連付けを解除する．

3.3 入出力木

2.3 節で述べた演算木と同様，入出力木により，木構造で入出力を管理し，複数のプロセスから構成されるサービスの入出力時間を調整する．また，入出力デバイスごとに入出力木の根（以降，デバイスルートと名付ける）を用意することにより，各入出力デバイスで独立に入出力時間を調整する．入出力木の各頂点は，入出力を表しており，子を持つ入出力をディレクトリ入出力，子をもたない入出力をリーフ入出力と呼ぶ．ただし，入出力木の構成には，以下の制約がある．

（制約 1）優先度入出力は，デバイスルート直下のリーフ入出力でなくてはならない

性能調整入出力と優先度入出力は，いずれも入出力要求の実行順序を決定する．しかし，性能調整入出力と優先度入出力における入出力要求の実行順序の制御目的が異なる．具体的には，性能調整入出力は，理想の入出力時間内に実 I/O 処理を終了することを目的とする．一方，優先度入出力は，優先度の降順で入出力要求を実行することを目的とする．したがって，一つの入出力要求に対して両制御を適用できない．このため，優先度入出力は，性能調整入出力の親または子になれない．また，優先度入出力は，入出力要求の実行順序を決定する相対的な指標であるため，優先度入出力の持つ優先度を下位へ分配できない．このため，優先度入出力はディレクトリ入出力になれない．以上により，優先度入出力は，デバイスルート直下のリーフ入出力でなくてはならない．

（制約 2）プロセスは同一入出力木に属する複数のリーフ入出力に関連付けできない

入出力は，プロセスの入出力時間または入出力要求の実行順序を決定する．このため，同一入出力木に属する複数の入出力に関連付けられたプロセスが入出力要求を発行した場合，どの理想の入出力時間または入出力要求の実行順序に基づき入出力処理を実行してよいか判断できない．このため，プロセスは，同一入出力木に属する複数の入出力と関連付けできない．

プロセスの要求入出力性能は，プロセスに関連付けられた性能調整入出力からルートデバイスまでの性能調整入出力が持つ要求入出力性能の積算で求める．入出力木の例を図 5 に示す．図 5 では，サービス A は 3 つのプロセスで構成され，サービス A のサブプロセスグループとなるサービス A ’を持つ．Disk1 において，サービス A のプロセスグループに対す

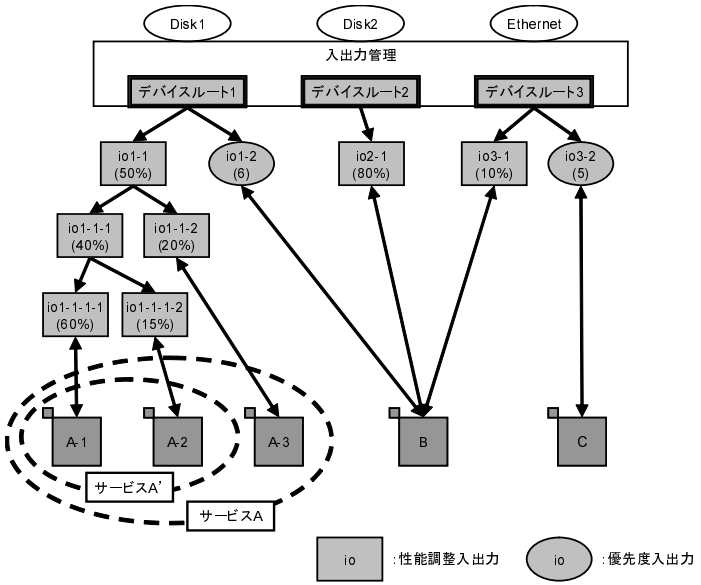

図 5 入出力木によるプロセスグループの表現

る入出力は性能調整入出力 io1-1 であり，入出力の程度は 50 %である．これより，サービス A は Disk1 そのものの性能の 50 %で入出力処理を実行できる．また，サービス A ’のプロセスグループに対する入出力は性能調整入出力 io1-1-1 であり，入出力の程度は 40 %である．これにより，サービス A ’は性能調整入出力 1-1 の 40 %の性能で入出力処理を実行することとなり，これは Disk1 そのものの性能の 20 %（= 50 %× 40 %）である．また，プロセス B は，複数の入出力と関連付けられることにより，複数の入出力デバイスを利用できる．例えば，Disk1 において，プロセス B に対する入出力は，優先度入出力 io1-2 であるため，Disk1 に対するプロセス B の入出力要求は，優先度 6 で実行される．一方，Disk2 においては，性能調整入出力 io2-1 であるため，プロセス B は，Disk2 そのものの性能の 80 %で入出力処理を実行できる．同様に，Ethernet に対しては，Ethernet そのものの性能の 10 %で入出力処理を実行できる．

3.4 期待される効果

資源「入出力」を導入することにより，プロセスが使用できる入出力デバイスを制限できる．具体的には，使用禁止にしたい入出力デバイスに対応する入出力木に属す入出力に当該プロセスを関連付けないことにより，プロセスが使用できる入出力デバイスを制限できる．

また，入出力木による特徴を以下に示す．

(1) プロセスグループの入出力性能の保証

プロセスグループに対応するディレクトリ入出力に性能調整入出力を関連付け，要求入出力性能を変更することにより，プロセスグループを構成するすべてのプロセスの入出力性能を一度に調整できる．これにより，プロセスの入出力時間を保証できるため，2.4 節で述べた（問題 1）を解決できる．

(2) 他プロセスの入出力処理への影響の抑制

プロセスグループに対応するディレクトリ入出力に性能調整入出力を関連付け，要求入出力性能を低く設定することにより，他プロセスの入出力時間の長大化を抑制できる．例えば，バックグラウンドで走行するプロセスに関連付けられた性能調整入出力の要求入出力性能を低く設定することにより，単位時間当たりの実 I/O 処理の回数を減少させる．このため，フォアグラウンドで走行するプロセスの実 I/O 処理をただちに実行でき，フォアグラウンドで走行するプロセスの入出力時間の長大化を抑制できる．また，性能調整入出力は，他プロセスの入出力要求の数に関わらず，入出力時間を調整するため，2.4 節で述べた（問題 2）を解決できる．

4. おわりに

Tender において，入出力処理時間の調整単位を資源化した資源「入出力」を利用したプロセスグループの入出力性能の調整機能を提案した．

Tender における資源「演算」を利用したプロセスグループのプロセッサ性能の調整機能を述べ，その問題点を述べた．具体的には，プロセスは，プロセッサ処理と入出力処理を繰り返すため，プロセッサ性能のみ調整しても，入出力時間の長大化の影響により，サービスの実行速度を上手く調整できない．したがって，サービスの実行速度を調整するためには，入出力性能の調整機能が必要である．提案機能は，資源「演算」と同様に資源「入出力」を木構造で管理した入出力木を導入し，プロセスグループの入出力性能を調整する．また，入出力デバイスごとに入出力木を構成することにより，入出力デバイスごとに入出力性能を調整できる．

残された課題として，実装と評価がある．

Vol.2011-OS-116 No.4
2011/1/24

参 考 文 献

1) Liu, C.L. and Layland, J.W.: Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment, *J. ACM*, Vol.20, pp.46–61 (1973).
2) Jones, M.B., Roşu, D. and Roşu, M.-C.: CPU reservations and time constraints: efficient, predictable scheduling of independent activities, *SIGOPS Oper. Syst. Rev.*, Vol.31, pp.198–211 (1997).
3) Nieh, J. and Lam, M.S.: The design, implementation and evaluation of SMART: a scheduler for multimedia applications, *SIGOPS Oper. Syst. Rev.*, Vol. 31, pp. 184–197 (1997).
4) Denning, P.J. and Schwartz, S.C.: Properties of the working-set model, *Commun. ACM*, Vol.15, pp.191–198 (1972).
5) Ghanem, M.Z.: Dynamic partitioning of the main memory using the working set concept, *IBM J. Res. Dev.*, Vol.19, pp.445–450 (1975).
6) Kawachiya, K. and Tokuda, H.: Dynamic QOS control based on the QOS-Ticket model, *Proc. IEEE ICMCS '96*, pp.78–85 (1996).
7) Oliviera, C., Kim, J. and Suda, T.: An Adaptive Bandwidth Reservation Scheme for High Speed Multimedia Wireless Networks, *IEEE J. Selected Areas in Comm.*, Vol.16, pp.858–874 (1998).
8) 谷口秀夫：サービス処理時間を調整するプロセスのスケジュール法，電子情報通信学会論文誌 (D-I)， Vol.81, No.4, pp.386–392 (1998).
9) 谷口秀夫：入出力時間の制御によりサービス時間を調整する制御法，電子情報通信学会論文誌 (D-I)， Vol.83, No.5, pp.469–477 (2000).
10) 谷口秀夫：分散指向永続オペレーティングシステム ***Tender***，情報処理学会コンピュータシステムシンポジウム，シンポジウム論文集，Vol.95, No.7, pp.47–54 (1995).
11) 田端利宏，乃村能成，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」を利用したプロセスグループの実行性能調整法，電子情報通信学会論文誌 (D-I), Vol.87, No.11, pp.961–974 (2004).
12) Yamauchi, T., Hara, T. and Taniguchi, H.: A Mechanism that Bounds Execution Performance for Process Group for Mitigating CPU Abuse, *Communications in Computer and Information Science (CCIS)*, Vol.122, pp.83–93 (2010).
13) 長尾　尚，谷口秀夫：入出力性能の制御によりプログラム実行速度を調整する制御法，電子情報通信学会技術研究報告. CPSY, コンピュータシステム， Vol.109, No.237, pp. 33–38 (2009).