Vol.2011-OS-118 No.19
2011/7/28

Tender オペレーティングシステムにおける資源「入出力」の実現と評価

一井晴那[†1]　長尾尚[†1]
山内利宏[†1]　谷口秀夫[†1]

個々のプログラムの実行速度を調整することができれば，利用者が重要でないと判断するプログラムの実行速度を制限することで，重要なプログラムの実行速度の低下を抑制できる．***Tender*** オペレーティングシステムでは，資源「演算」を利用したプロセッサ性能の調整機能を実現している．しかし，プロセッサ性能を調整しても，入出力処理の影響を受けるため，プログラムの実行速度が低下する．そこで，***Tender*** に入出力性能性能調整機能として資源「入出力」を提案した．ここでは，***Tender*** における資源「入出力」による入出力性能調整機能の実現方式と評価について述べる．

Implementation and Evaluation of *I/O* Resource on *Tender* Operating System

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

For the mechanism of regulating the speed of program execution individually, it prevents the speed of necessary program execution by limiting the speed of unnecessary program execution. We proposed the mechanism for regulating processor performance by ***Execution*** resources on the ***Tender*** operating system. However, limiting the speed of program execution by regulating processor performance is affected by I/O processing, and the speed of program execution does slow down. Therefore, we proposed ***I/O*** resources for regurating I/O performance on ***Tender***. In this paper, we show implementaion of regurating I/O performance by ***I/O*** resources on ***Tender***, and evaluate it.

1. はじめに

計算機の普及により，1 台の計算機上で複数のサービスを実行できるようになった．しかし，個々のサービスの実行速度は，ハードウェア性能や他のサービスの影響を受ける．例えば，ウィルス対策ソフトなどのプログラムにより，高負荷な処理が実行されると，Web ブラウザやテキストエディタなどのプログラムの起動や応答性は著しく低下する．一方，プログラムの実行速度を調整できれば，資源競合時であっても，バックグラウンドで実行するプログラムの実行速度を制限することにより，フォアグラウンドで提供しているプログラムの実行速度低下を抑制できる．

我々はこれまでに，計算機ハードウェア性能の範囲でプログラムの実行速度を自由に調整する方式として，プロセッサ性能の調整法[1] と入出力性能の調整法[2] を提案した．また，分散指向永続オペレーティングシステム ***Tender*** (The ENduring operating system for Distributed EnviRonment)[3] 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[4] を提案した．しかし，プロセッサ性能だけを調整しても，入出力処理の影響を受けるため，プログラムの実行速度の調整精度は低下する．例えば，プロセッサ性能の調整により，プロセスに割り当てられるプロセッサ時間を保証しても，他プロセスによる実 I/O 処理の終了待ちが発生し，入出力時間が長大化する．このため，プログラムの実行速度の調整精度は低下する．そこで，***Tender*** の入出力性能の調整機能として，資源「入出力」[5] を提案した．ここでは，提案した資源「入出力」による入出力性能調整機能の実現方式と評価について述べる．

2. 資源「入出力」と入出力木[5]

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，各資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．

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

Vol.2011-OS-118 No.19
2011/7/28

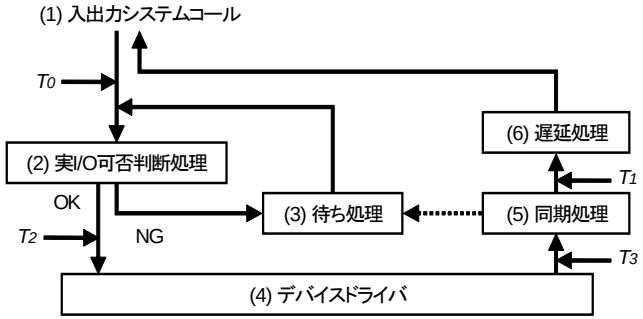


図 1 入出力要求数を調整する制御法

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「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．入出力の程度には，要求入出力性能と優先度の 2 種類がある．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の 2 種類がある．

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[6] の基本方式を**図 1** 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．許容値とは，実 I/O 要求プロセス数の最大値である．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．

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

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．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．調整しないプロセス (以降，共存プロセスと名付ける) 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．最小値を 1 としない場合，入出力性能調整を行うプロセス (以降，被調整プロセスと名付ける) の要求入出力性能が高い場合，その被調整プロセスが入出力処理を行わない間，許容値は 0 となってしまい，他のプロセスが入出力処理を行うことができなくなる．

(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) 実 I/O 処理終了後，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，被調整プロセスの入出力時間が理想の入出力時間となるまでに必要な時間である．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．

$$T_S = \frac{100}{\text{要求入出力性能}} \times \text{実 I/O 時間} - (T_1 - T_0) \quad (2)$$

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

2.3 入出力木

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．これを入出力木と名付ける．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，子を持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
(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

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 2** に示す．図 2 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．

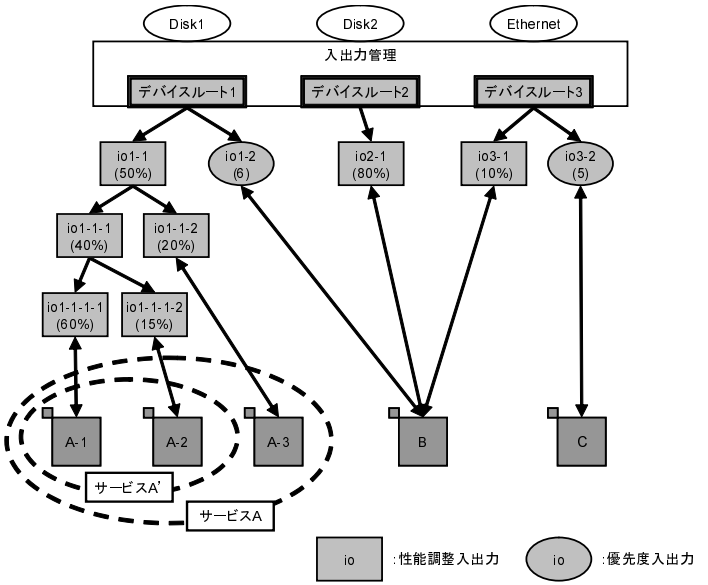


図 2 入出力木によるプロセスグループの表現

2.4 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として，プロセスグループの入出力性能の保証と他プロセスの入出力処理への影響の抑制がある．

3. 実現方式

3.1 課題

資源「入出力」の実現項目として，以下の 3 つがある．

(1) 2.2 節で述べた入出力管理の機能を実現する提供インタフェース
(2) 入出力性能調整機能の各処理
(3) 許容値の更新契機の検討と更新処理

表 1 入出力管理のインタフェース

機能	形式
入出力の生成	create_io(iodeg, dev_kind); iodeg: 入出力の程度, dev_kind: 入出力デバイスの種類
入出力の削除	delete_io(ioid); ioid: 入出力識別子
入出力に対するプロセスまたは入出力の関連付け	attach_io(ioid, pid/ioid); ioid: 入出力識別子 pid/ioid: 関連付けを行うプロセス識別子または入出力識別子
入出力に対するプロセスまたは入出力の関連付け解除	detach_io(ioid, pid/ioid); ioid: 入出力識別子 pid/ioid: 関連付け解除を行うプロセス識別子または入出力識別子
入出力の程度または許容係数の変更	ctrl_io(ioid, value, io_op); ioid: 入出力識別子 value: 変更後の入出力の程度または変更後の許容係数 io_op: 操作種別 (CHANGE_IODEG または CHANGE_K を指定)
入出力要求の受付	ctrl_io(ioid, pid, dst_node, disk_option, *buf, dst_addr, size, disk_rw, io_op); ioid: 入出力識別子, pid: プロセス識別子, dst_node: ドライブ番号 disk_option: ディスク I/O 方式, *buf: メモリアドレス dst_addr: 読み込み, または書き込み開始セクタ番号 size: 読み込み, または書き込むセクタ数 disk_rw: 読み込み, または書き込みの指定 io_op: 操作種別 (DISK_REAL_IO を指定)

以降では，これらの実現方式について述べる．

3.2 提供インタフェース

プロセスは，資源「入出力」を生成し，生成した資源「入出力」へ入出力要求を発行することで，入出力処理を行う．2.2 節で述べた各機能を実現する入出力管理のインタフェースを**表 1** に示し，以下で具体的に述べる．

create_io() は，入出力の程度 iodeg を持つ資源「入出力」を生成し，指定された入出力デバイスの種類 dev_kind に対応するルート入出力に関連付ける．ルート入出力とはデバイスルート直下に存在する要求入出力性が 100 %の入出力である．指定された入出力の程度が，1〜100 %の場合は要求入出力性能とみなし，0〜-255 の場合は優先度とみなす．優先度は値が大きいほど高い．create_io() は戻り値として生成した資源「入出力」の入出力識別子を返却する．

delete_io() は，入出力識別子 ioid を持つ資源「入出力」を削除する．削除可能な資源「入出力」は，プロセスが関連付けられていない資源「入出力」で，ルート入出力は削除対象外である．

attach_io() は，入出力識別子 ioid を持つ資源「入出力」に，プロセス識別子 pid を持つプロセスまたは資源「入出力」ioid を関連付ける．資源「入出力」とプロセスの関連付けにおいて，資源「入出力」がルート入出力の場合，関連付けを行ったプロセスのみが要求入出力性能が 100 % (入出力デバイスの性能そのもの) で入出力処理を行うことが可能となる．このため，ルート入出力の下の入出力階層に存在する資源「入出力」が存在しないものとみなすため，ルート入出力の下の入出力階層に存在する資源「入出力」は使用できなくなる．したがって，ルート入出力にプロセスを関連付けた場合，入出力デバイスの占有となる．関連付け対象となる資源「入出力」は，既に他のプロセスを関連付けられていない資源「入出力」である．また，関連付け対象となるプロセスは同じ入出力デバイスの種類に対応した資源「入出力」を関連付けられていないプロセスである．

detach_io() は，入出力識別子 ioid を持つ資源「入出力」とプロセス識別子 pid を持つプロセスまたは資源「入出力」ioid の関連付けを解除する．ルート入出力とプロセスの関連付けを解除する場合，入出力デバイスの占有解除となり，ルート入出力の下の入出力階層に存在する資源「入出力」を再び使用可能となる．

ctrl_io() は，引数の io_op で指定される値によって実行する処理が異なる．

io_op が CHANGE_IODEG の場合，引数として，入出力識別子，変更後の入出力の程度および操作種別に CHANGE_IODEG を指定する．ctrl_io() は，入出力識別子 ioid を持つ資源「入出力」の入出力の程度を指定された値 value に変更することができる．入出力の程度の変更範囲としては，要求入出力性能から優先度，または優先度から要求入出力性能に変更することは禁止であるため，エラーとする．

io_op が CHANGE_K の場合，引数として，入出力識別子，変更後の許容係数および操作種別に CHANGE_K を指定する．ctrl_io() は，入出力識別子 ioid を持つ資源「入出力」の入出力デバイスの種類に対応する許容係数を指定された値 value に変更する．許容係数は，各入出力デバイスごとに 1 つ持つ．

io_op が DISK_REAL_IO の場合，引数として，入出力識別子，プロセス識別子，ドライブ番号，ディスク I/O 方式，メモリアドレス，読み込みまたは書き込み開始セクタ，読み込みまたは書き込むセクタ数，読み込みまたは書き込みの指定，および操作種別に DISK_REAL_IO を指定する．ドライブ番号，ディスク I/O 方式，メモリアドレス，読み込みまたは書き込み開始セクタ，および読み込みまたは書き込むセクタ数はディスクへの読み込みまたは書き込みの際に指定する引数である．ctrl_io() は，入出力識別子 ioid を持つ資源「入出力」か

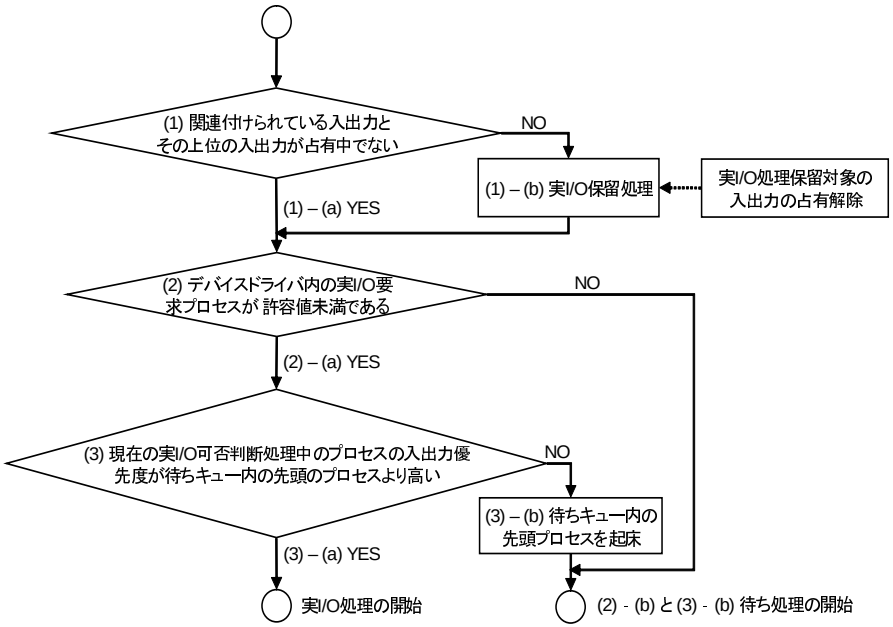


図 3 実 I/O 可否判断処理の処理流れ

ら使用する入出力デバイスを特定し，適切な入出力デバイスへ入出力要求を発行する．入出力処理終了後，入出力デバイスからの結果を受け取り，呼び出し元に返却する．また，指定された資源「入出力」と指定されたプロセスが関連付けられていない場合は，エラーである．なお，入出力処理要求発行後に，入出力識別子 ioid を持つ資源「入出力」もしくはその上位の入出力階層に属する資源「入出力」が占有された場合は占有が解除するまで，入出力処理の保留を行う．この保留処理は実 I/O 可否判断処理内で行う．

3.3 各機能の処理内容

3.3.1 実 I/O 可否判断処理

実 I/O 可否判断処理では，許容値により，デバイスドライバ内の実 I/O 要求プロセス数を制限する．**図 3** に実 I/O 可否判断の処理流れを示し，以下に実 I/O 可否判断処理の処理流れを述べる．

(1) 実 I/O 処理を行うプロセスは，そのプロセスを関連付けられている入出力またはその入出力の上位の入出力階層に属する入出力が占有されていないか調べる．
- (a) 占有されていない場合は (2) へ進む．
- (b) 占有されている場合は，占有が解除されるまで WAIT 状態とし，実 I/O 要求の保留処理を行う．

(2) デバイスドライバ内の実 I/O 要求プロセス数と許容値を比較する．
- (a) デバイスドライバ内の実 I/O 要求数が許容値未満の場合は，(3) へ進む．
- (b) デバイスドライバ内の実 I/O 要求数が許容値以上の場合は待ち処理を行う．WAIT 状態の解除後，(1) に戻る．

(3) 現在の実 I/O 判断処理中のプロセスの入出力優先度と待ちキュー内の先頭プロセスの入出力優先度を比較する．
- (a) 待ちキュー内の先頭プロセスの入出力優先度より，現在の実 I/O 判断処理中のプロセスの入出力優先度の方が高い場合，現在の実 I/O 判断処理中のプロセスが実 I/O 処理を行う．
- (b) 待ちキュー内の先頭プロセスの入出力優先度より，現在の実 I/O 判断処理中のプロセスの入出力優先度の方が低い場合，待ちキュー内の先頭のプロセスを起床させ，現在の実 I/O 判断処理中のプロセスは待ち処理を行う．WAIT 状態解除後，(1) に戻る．

プロセスに関連付けられている入出力の上位の入出力階層に占有状態の入出力が存在する場合，入出力処理を保留する．このため，待ち処理内で WAIT 状態となっている間に上位の階層の入出力が占有された場合，占有が解除されるまで，入出力処理を保留しなければならない．したがって，実 I/O 処理を許可するか否か判断する際に，(1) の処理を行う．実 I/O 可否判断処理で実 I/O 処理を許可されるプロセスは最も入出力優先度の高いプロセスであるため，実 I/O 処理を許可する場合は現在の実 I/O 判断処理中のプロセスと待ち処理内で最高入出力優先度を持つプロセスの入出力優先度を比較する必要があるため，(3) の処理を行う．

3.3.2 待 ち 処 理

待ち処理では，実 I/O 可否判断処理において，実 I/O 処理を許可されなかったプロセスを WAIT 状態にして待ちキューで管理する．待ちキュー内において，WAIT 状態のプロセスは入出力優先度が降順になるように整列している．以下に待ち処理の処理流れを述べる．

(1) 該当するプロセスを WAIT 状態にする．

(2) 待ち処理内で WAIT 状態のプロセスの起床後，実 I/O 可否判断処理に戻る．

待ちキューの実現方法として，実 I/O 処理を待つプロセスを管理する実 I/O 処理待ちリストを使用する．実 I/O 処理待ちリストのエントリは入出力要求発行時に確保し，あらかじめリスト内で整列させておく．実 I/O 処理待ちリストは要求入出力性能が降順になるように被調整プロセスを整列した後，共存プロセスの優先度が降順になるように整列させる．このため，待ち処理では並べ替えを行わず，該当プロセスを WAIT 状態とするだけである．

3.3.3 同 期 処 理

同期処理では待ち処理内の待ちキューの先頭プロセスを起床させる．以下に同期処理の処理流れを述べる．

(1) 待ちキューの先頭にプロセスが存在するか否かを調べる．
  (a) 存在する場合は次の (2) へ，
  (b) 存在しない場合は同期処理を終了する．
(2) 待ちキューの先頭のプロセスを起床させ READY 状態にする．

3.3.4 遅 延 処 理

遅延処理は被調整プロセスのみが処理を行う．このため，共存プロセスは遅延処理を行わない．以下に被調整プロセスの処理流れを述べる．

(1) 操作対象のプロセスが被調整プロセスであるか否かを調べる．
  (a) 被調整プロセスの場合は (2) へ進む．
  (b) 共存プロセスの場合は遅延処理を終了する．
(2) プロセスの要求入出力性能を基に理想の入出力時間を算出し，(3) へ進む．
(3) rdtsc 命令により，現在のハードウェアクロックを取得し，実際の経過時間を算出する．
(4) (2) で算出した理想の入出力時間と (3) で算出した実際の経過時間の差分を計算する．
(5) (4) の計算結果に前回の遅延処理時に発生した遅延時間くりこし値を加算し，遅延時間とする．
(6) 算出した遅延時間が 0 より大きいか否かを調べる．
  (a) 0 より大きい場合は (7) に進む．
  (b) 0 以下の場合は遅延時間くりこし値に加算し，遅延処理を終了する．
(7) rdtsc 命令により，停止処理開始時のハードウェアクロックを取得する．
(8) 遅延処理を実行する．
(9) 遅延処理終了後，rdtsc 命令により，遅延処理終了後のハードウェアクロックを取得する．
(10) (7) と (9) より，実際に停止した時間 (実停止時間) を算出し，実停止時間と遅延時間の差分を次回の遅延処理において，遅延時間くりこし値として使用する．

遅延処理では，理想の入出力時間と実際の経過時間との差に遅延時間くりこし値を加算することで遅延時間を算出し，算出した遅延時間を基に遅延処理を行う．遅延処理終了後，実際に遅延した時間と依頼した遅延時間との差をとり，次回の遅延処理に繰り越すことで，調整精度の低下を防ぐ．

3.4 許容値の更新契機と処理内容

許容値は，式 (1) 中の k もしくは P_i が変化した場合に更新する必要がある．入出力管理の各機能を使用する中で，k もしくは P_i が変化する．許容値は入出力性能調整の調整精度に影響を与えるため，許容値を正確に定めることは重要である．許容値の更新契機を以下に示す．

(契機 1) 入出力に対するプロセスの関連付け
(契機 2) 入出力に対するプロセスの関連付け解除
(契機 3) 実 I/O 可否判断処理前
(契機 4) 遅延処理終了後
(契機 5) 入出力の要求入出力性能の変更
(契機 6) 許容係数の変更

(契機 1) において，入出力にプロセスを関連付ける場合，許容値の算出対象となるプロセスが増加する．また，(契機 2) において，入出力とプロセスの関連付けを解除した場合，許容値の算出対象となるプロセスが減少する．つまり，式 (1) の P_i が変化するため．(契機 1) と (契機 2) において許容値を更新する必要がある．(契機 3) において，実 I/O 可否判断処理を行うプロセスは遅延処理が終了するまで，許容値の算出対象とならない．また，(契機 4) において遅延処理後のプロセスは，実 I/O 要求を発行できる．このため，許容値の算出対象となる P_i が変更される．つまり，式 (1) の P_i が変化するため，(契機 3) と (契機 4) において許容値を更新する．(契機 5) において，入出力の程度を変更した入出力にプロセスが関連付けられている場合，関連付けられているプロセスの入出力の程度も変更される．つまり，式 (1) の P_i が変化するため．(契機 5) において許容値を更新する必要がある．(契機 6) について，許容係数を変更する場合，許容値の算出に使用する被調整プロセスの持つ要求入出力性能の数が変更される．つまり，式 (1) の k が変化するため．(契機 6) において許容値を更新する必要がある．

以上より，(契機 1)〜(契機 6) において，許容値の更新を行う．このため，各契機に対応した処理を終了後，許容値の更新処理を行う．許容値の更新処理の処理流れを**図 4** に示す．

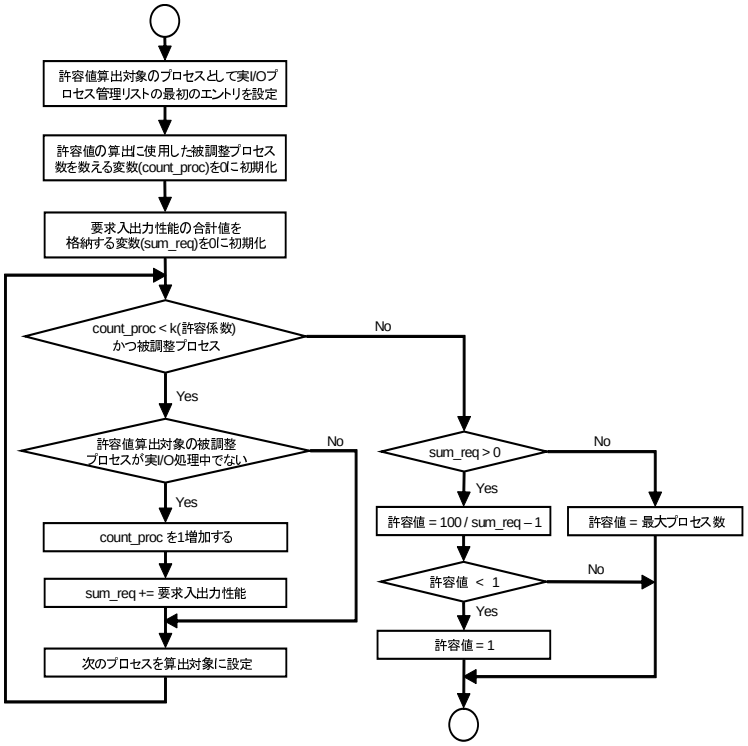


図 4 許容値の更新処理

入出力管理では，入出力に関連付けられたプロセスに関する情報を実 I/O プロセス管理リストで管理している．実 I/O プロセス管理リスト内では，プロセスは要求入出力性能が降順になるように被調整プロセスに関する情報を整列した後，優先度が降順になるように共存プロセスに関する情報を整列している．このため，実 I/O プロセス管理リストの先頭のプロセスから順に許容値の算出対象に指定する．後は，許容値の算出式にしたがって許容値の算出を行う．ここで，実 I/O プロセス管理リスト内の被調整プロセスの数が k 以下の場合，全ての被調整プロセスが許容値の算出対象となる．また，被調整プロセスが存在しない場合，許容値は最大プロセス数となる．

4. 評　　価

4.1 評 価 条 件

Celeron(2.0GHz) プロセッサと IDE HDD である ST340016A を搭載した計算機に資源「入出力」による入出力要求数を調整する制御法を実装した ***Tender*** で測定を行い，入出力性能の調整精度に着目して評価を行った．評価プログラムは，I/O 処理として ***Tender*** で read() システムコールに相当する iorequest() により磁気ディスク装置から 512 バイト読み込む入力処理を繰り返す．なお，入力位置はランダムである．また，iorequest() の前後で取得したハードウェアクロックの差を測定し，入出力時間とする．CPU 処理時間と I/O 処理時間の比率を処理比率と名付ける．調整精度の評価尺度として，調整比を用いる．調整比は，

$$\text{調整比} = \frac{\text{入出力時間の実測値}}{\text{理想の入出力時間}} \tag{3}$$

である．ここで，入出力時間の実測値とは，***Tender*** で read() システムコールに相当する iorequest() 発行から戻り値が返却されるまでの時間である．また，理想の入出力時間とは，実 I/O 時間を要求入出力性能で割った値である．以降の評価では，評価プログラムによる CPU 処理と I/O 処理を 1002 回繰り返し，入出力時間を測定し，最初と最後を除いた 1000 回の入出力時間を評価に利用する．

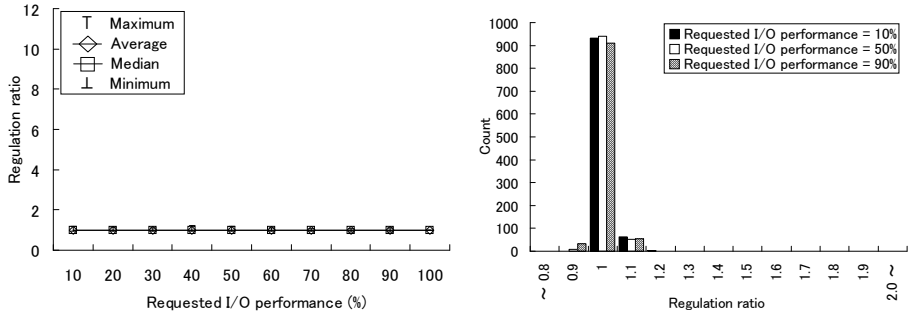
4.2 占有状態の調整精度

占有状態における被調整プロセスの調整比と頻度分布について，被調整プロセスの処理比率が 0 : 1 とし，***Tender*** で測定した場合を**図 5** に，Celeron(1.8GHz) プロセッサと SATA HDD である ST380815AS を搭載した計算機に文献 6) の制御法を実装した FreeBSD6.3-R で測定した場合を**図 6** に示す．図 5 から以下のことがいえる．

(1) 要求入出力性能にかかわらず，最大値，平均値，中央値，および最小値の場合，調整比はほぼ 1.0 付近となっている．例えば，入出力性能が 40 %のとき，調整比の最大値が最大となる．その値は約 1.21 であり，1 に近い．
(2) 要求入出力性能にかかわらず，調整比の大半は 1 付近であり，著しく大きい値は存在しない．

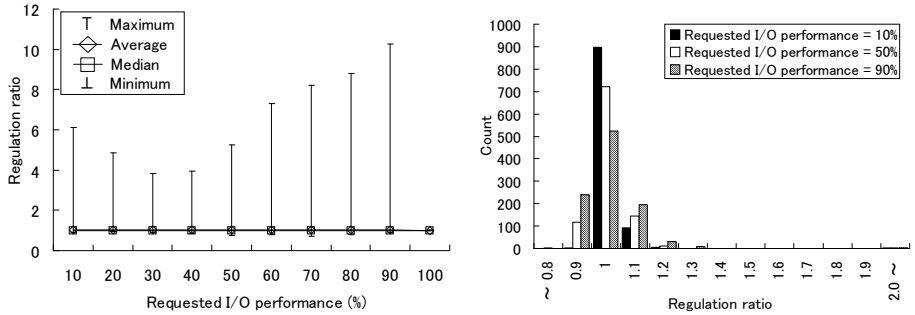
以上から，***Tender*** において，被調整プロセスが 1 つだけ走行する場合の調整精度は高いといえる．また，***Tender*** と FreeBSD6.3-R の比較から以下のことがいえる．

(1) 調整比の最大値から ***Tender*** と FreeBSD6.3-R では，***Tender*** の方が調整精度が



(A) 調整比　　(B) 調整比の頻度分布

図 5　調整比と調整比の頻度分布 (CPU:I/O=0:1, ***Tender*** で測定)



(A) 調整比　　(B) 調整比の頻度分布

図 6　調整比と調整比の頻度分布 (CPU:I/O=0:1, FreeBSD6.3-R で測定)

高い．

Tender では，調整比の最大値が要求性能にかかわらず 1 付近であったのに対し，FreeBSD6.3-R では 1 から大きく外れた値となっている．以上より，FreeBSD6.3-R より ***Tender*** で実装を行った方が調整精度が高いといえる．この理由として，***Tender*** で測定した場合の遅延可否閾値が，FreeBSD6.3-R で測定した場合の遅延可否閾値に比べて小さいことが考えられる．遅延処理において，遅延可否閾値以下の遅延時間は遅延処理を実行されず，次の遅延処理にくりこされる．FreeBSD6.3-R における遅延可否閾値は 10ms であるのに対し，***Tender*** では 1ms である．このため，遅延可否閾値が小さい ***Tender*** の方が，遅延処理が実行されやすいため，調整精度が高くなると考えられる．

5. おわりに

Tender において，資源「入出力」による入出力性能調整機能の実現方式とその評価について示した．実現方式として，入出力管理の提供インタフェースを示し，各機能の処理内容について述べた．入出力性能調整を行う方式として，入出力要求数を調整する制御法を使用する．このため，入出力性能を調整する制御法の実 I/O 可否判断処理，待ち処理，同期処理，および遅延処理の処理内容についてそれぞれ述べた．また，実 I/O 可否判断処理において，実 I/O 要求を許可するか否かの指標となる許容値の更新契機とその更新処理について述べた．評価では，被調整プロセスを 1 つだけ走行させる場合の評価を行った結果，要求入出力性能にかかわらず，調整精度が高い結果が得られた．

残された課題として，***Tender*** に資源「入出力」による複数プロセスをグループとする入出力性能調整機能の設計がある．

参 考 文 献

1) 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).
2) 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).
3) 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).
4) 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).
5) 長尾　尚，一井晴那，山内利宏，谷口秀夫：***Tender*** オペレーティングシステムにおける資源「入出力」の提案，情報処理学会研究報告， Vol.2011-OS-116, No.4, pp.1–7 (2011).
6) 長尾　尚，谷口秀夫：入出力要求数の制御によりサービス時間を調整する制御法の実現と評価，電子情報通信学会論文誌 (掲載予定)， Vol.J94-D, No.7 (2011).