Tender における資源「入出力」の評価

佐野 弘尚[1]　山内 利宏[1]　谷口 秀夫[1]

概要：1 台の計算機上で複数のプログラムを実行した場合，個々のプログラムの実行は，資源の競合により，他のプログラムの影響を受ける．これにより，利用者が優先したいプログラムの実行速度が低下することがある．ここで，重要でないプログラムの実行速度を調整し資源利用を抑制することにより，優先したいプログラムの実行速度の低下を抑制できる．また，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．このため，プロセスが利用できるプロセッサ処理の量や入出力処理の量を調整することにより，プログラムの実行速度を調整できる．そこで，プロセッサ処理の量を調整する制御法を提案した．しかし，プロセッサ処理の量のみの調整では，プログラムの入出力処理の割合が大きい場合，プログラムの実行速度の調整精度は低下する．これにより，優先したいプログラムの実行速度の低下を抑制できなくなる．このため，入出力処理の量を調整する制御法として，入出力要求数を調整する制御法を提案した．本稿では，***Tender*** オペレーティングシステムに資源「入出力」として実現した入出力量を調整する制御法の実現における課題と対処について述べ，共存プロセスの有無による調整精度への影響を評価した結果を報告する．

1. はじめに

計算機の性能向上により，1 台の計算機上で複数のプログラムを実行できる．1 台の計算機上で複数のプログラムを実行した場合，個々のプログラムの実行は，他のプログラムの影響を受ける．たとえば，ウイルス対策ソフトが実行を開始すると，Web ブラウザなどのフォアグラウンドのプログラムの応答時間が長くなることがある [1]．ここで，優先して実行する必要のないプログラムの実行を抑制することにより，他のプログラムへの影響を抑制できる．

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．このため，単位時間に利用できるプロセッサ処理の量と入出力処理の量を調整することにより，プログラムの実行速度を調整できる．文献 [2] において，プロセッサ処理の量を調整する制御法を提案した．しかし，この制御法は，プロセッサ処理の量の調整のみによりプログラムの実行速度を調整している．このため，プログラムの入出力処理の割合が大きい場合，プログラムの実行速度の調整精度は低下する．これにより，優先したいプログラムの実行速度の低下を抑制できなくなる．そこで，文献 [1] において，入出力処理の量を調整する制御法として，入出力要求数を調整する制御法を提案した．この制御法により，プログラムの入出力処理の割合が大きい場合であっても，プログラムの実行速度の調整精度は低下しない．また，本制御法に優先度の概念を導入し，分散指向永続オペレーティングシステム ***Tender***[3]（以降，***Tender*** と略す）に資源「入出力」として実現した [4]．

本稿では，まず，資源「入出力」として実現した入出力処理の量を調整する制御法（以降，入出力性能調整機能と略す）への優先度の概念の実現における課題と対処について述べる．次に，入出力性能調整機能を評価した結果を報告する．具体的には，優先度を持つ資源「入出力」を関連付けているプロセスの有無による入出力性能調整機能の調整精度への影響を評価した結果について報告する．

2. 入出力性能調整機能

本章では，文献 [4] で報告した入出力性能調整機能を資源「入出力」として実現した ***Tender*** と資源「入出力」について説明する．

2.1 *Tender* オペレーティングシステム

Tender では，オペレーティングシステム（以降，OS と略す）が制御し，管理する対象を資源と呼び，資源の分離と独立化をしている．たとえば，***Tender*** では，既存の OS のプロセスを複数の資源に分離し，各資源に識別子を与え，独立した資源として管理している．

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

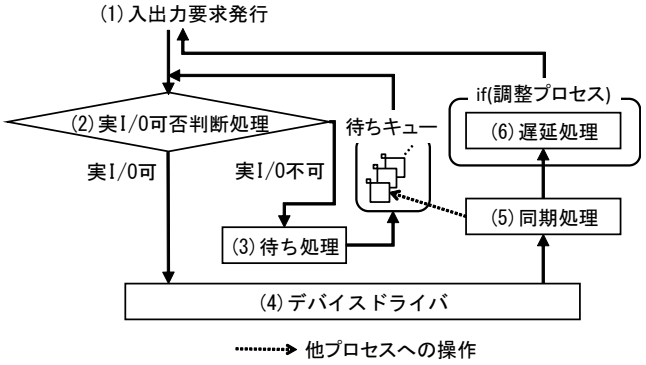

図 1 入出力性能調整機能の基本方式

2.2 資源「入出力」

2.2.1 資源「入出力」の役割

資源「入出力」は，***Tender*** におけるプロセスの入出力処理を管理するための資源である．入出力処理を行うプロセスは，資源「入出力」を生成し，生成した資源「入出力」と自身を関連付け，自身と関連付けている資源「入出力」へ入出力要求を発行する．資源「入出力」は，アクセス先の入出力デバイスの種類と入出力性能を調整する程度（以降，入出力の程度と略す）を持つ．プロセスは，入出力要求を発行した資源「入出力」が持つ入出力の程度に基づき，入出力要求を制御され，利用できる入出力性能を調整される．入出力の程度として，要求入出力性能と優先度がある．資源「入出力」の利用手順を 2.6 節に示す．

2.2.2 要求入出力性能を持つ資源「入出力」

要求入出力性能は，プロセスに与える入出力デバイスの性能を示す．入出力デバイスを占有して実行した際の入出力デバイスの性能を 100%とし，1%単位でプロセスに与える入出力デバイスの性能を指定できる．たとえば，要求入出力性能の値が 10%の場合，入出力デバイス占有時の 10%の入出力性能で入出力処理を実行するため，入出力処理に要する時間（以降，入出力時間と略す）は，入出力デバイスを占有して実行した際の入出力時間の 10 倍となる．以降，要求入出力性能を持つ資源「入出力」を性能調整入出力と呼ぶ．また，性能調整入出力を関連付けているプロセスを調整プロセスと呼ぶ．

2.2.3 優先度を持つ資源「入出力」

優先度は，プロセスが入出力処理を実行する優先順位を示す．最高優先度を 0，最低優先度を-255 として整数で指定する．優先度の値が大きいほど，優先して入出力処理を実行する．以降，優先度を持つ資源「入出力」を優先度入出力と呼ぶ．

2.3 基本方式

入出力性能調整機能の基本方式を**図 1** に示し，処理内容を以降で説明する．

（1）プロセスは，入出力要求を発行する．

（2）実 I/O 可否判断規則（詳細は，3.2 節にて述べる）に基づき，実 I/O 要求を許可するか否かを判断する．実 I/O 要求を許可する場合，（4）デバイスドライバへ実 I/O 要求を発行する．許可しない場合，（3）待ち処理を行う．

（3）実 I/O 要求を許可しなかったプロセスを待ち状態にする．また，待ち状態にしたプロセスに関連付けている資源「入出力」を入出力優先度（詳細は，2.4 節にて述べる）に基づき，待ちキューにつなぐ．

（4）実 I/O 処理を行う．

（5）待ちキューの先頭の資源「入出力」を関連付けているプロセス（以降，先頭プロセスと略す）を待ち状態から起床する．また，プロセッサを割り当てられるまでの待ち時間の短縮のために，起床したプロセスのプロセス優先度を高く設定する．

（6）実 I/O 処理を完了したプロセスが調整プロセスである場合，プロセスが利用できる入出力性能を調整するために，遅延時間だけ遅延する．調整プロセスでない場合，遅延処理を行わない．また，実 I/O 処理を完了したプロセスが遅延から復帰する際にプロセス優先度を高く設定する．遅延処理後，実 I/O 処理を完了したプロセスが，同期処理や遅延処理においてプロセス優先度を高く設定されたプロセスである場合，プロセス優先度を高く設定する前の優先度に戻す．

デバイスドライバは，到着順に実 I/O 要求を処理する．このため，入出力優先度の高い資源「入出力」を関連付けているプロセスほど，先に実 I/O 要求を発行する必要がある．そこで，実 I/O 可否判断処理（図 1（2））により，入出力優先度の低い資源「入出力」を関連付けているプロセスの実 I/O 要求を抑制し，入出力優先度の高い資源「入出力」を関連付けているプロセスの実 I/O 要求を先に発行する．

遅延処理（図 1（6））について，遅延時間の算出方法は，文献 [1] と同じである．具体的には，デバイスドライバでの入出力時間と調整プロセスの持つ要求入出力性能から理想の入出力時間を算出し，理想の入出力時間と入出力要求発行から遅延処理前までの入出力時間との差から遅延時間を算出する．遅延処理により，入出力要求を発行してから実 I/O 処理結果を返却するまでの時間（以降，実際の入出力時間と略す）を理想の入出力時間に近づける．

2.4 入出力優先度

入出力優先度とは，実 I/O 要求を許可しなかったプロセスに関連付けている資源「入出力」を待ちキューで管理する際の基準である．待ちキューでは，資源「入出力」を入出力優先度の降順で管理する．性能調整入出力，優先度入出力の順に入出力優先度は高い．性能調整入出力間では要求入出力性能の値が大きい資源「入出力」ほど，優先度入出力間では優先度の値が大きい資源「入出力」ほど入出力

表 1　入出力管理のインタフェース

通番	形式	機能
(1)	create_io(iodeg, dev_kind);	入出力の程度 iodeg を持つ資源「入出力」ioid を生成する．生成した資源「入出力」ioid は，dev_kind ごとに管理する．iodeg が 1 から 100 の場合は，要求入出力性能を意味し，-255 から 0 の場合は，優先度を意味する．
(2)	delete_io(ioid);	資源「入出力」ioid を削除する．
(3)	attach_io(ioid, pid);	資源「入出力」ioid とプロセス識別子 pid を持つプロセスを関連付ける．
(4)	detach_io(ioid, pid);	資源「入出力」ioid とプロセス識別子 pid を持つプロセスの関連付けを解除する．
(5)	change_iodeg(ioid, iodeg);	資源「入出力」ioid の入出力の程度を iodeg に変更する．
(6)	request_io(ioid, pid, dst_node, disk_option, *buf, dst_addr, size, disk_rw);	資源「入出力」ioid から利用する入出力デバイスを特定し，入出力デバイスへ入出力要求を発行する．引数として，入出力識別子 ioid，プロセス識別子 pid，ドライブ番号 dst_node，ディスク I/O 方式 disk_option，メモリアドレス*buf，読み込みまたは書き込み開始セクタ dst_addr，読み込みまたは書き込みセクタ数 size，および読み込みまたは書き込み指定子 disk_rw を指定する．

優先度は高い．同一の値の性能調整入出力や優先度入出力間では先に入出力要求を発行した資源「入出力」ほど入出力優先度は高い．

2.5 基本方針

入出力性能調整機能では，遅延処理により，調整プロセスが利用できる入出力性能を調整する．また，実 I/O 可否判断処理により，実 I/O 要求を発行しているプロセス数（以降，実 I/O 要求中プロセス数と略す）を抑制する．実 I/O 要求中プロセス数を抑制することにより，入出力優先度の高いプロセスの調整精度を高める．

2.6 提供機能とインタフェース

Tender の入出力管理の 6 つの機能のインタフェースを**表 1** に示す．資源「入出力」を管理する入出力管理が提供する機能は，資源「入出力」の生成，資源「入出力」の削除，資源「入出力」に対するプロセスの関連付け，資源「入出力」に対するプロセスの関連付けの解除，入出力の程度の変更，および入出力要求の受付である．

また，具体的な資源「入出力」の利用手順を述べる．入出力処理を行うプロセスは，利用したい入出力デバイスの種類と入出力の程度を指定し，資源「入出力」を生成する．次に，生成した資源「入出力」に自身を関連付ける．入出力要求の受付機能により，プロセスは，自身に関連付けている資源「入出力」へ入出力要求を発行し，入出力処理を行う．プロセスの入出力処理は，関連付けている資源「入出力」の持つ入出力の程度に基づき，制御される．入出力処理を停止する場合，資源「入出力」と自身の関連付けを解除する．次に，利用しない資源「入出力」を削除する．

3. 実現における課題

3.1 課題

性能調整入出力の調整精度を優先し，優先度入出力を共存させるための課題を以下に示す．

（課題 1）優先度入出力が存在する影響を抑制する実 I/O 可否判断規則の検討

実 I/O 可否判断規則とは，実 I/O 可否判断処理において，実 I/O 要求の可否を判断するための規則である [1]．優先度入出力を関連付けているプロセスによる実 I/O 要求が性能調整入出力の調整精度に与える影響を抑制するように実 I/O 可否判断規則を検討する．

（課題 2）許容値算出の際に考慮するプロセスの検討

許容値とは，実 I/O 可否判断処理において，実 I/O 要求を許可するプロセス数の最大値であり，調整プロセスが持つ要求入出力性能から算出する [1]．許容値に基づき，実 I/O 要求の可否を判断する．このため，適切な実 I/O 要求の可否判断ができるよう許容値算出の際に考慮する調整プロセスを検討する．

（課題 3）優先度変更処理の対象プロセスの検討

プロセス優先度を高く設定する処理（以降，優先度変更処理と略す）を同期処理や遅延処理に導入することにより，待ち状態や遅延から復帰する際のプロセッサ割り当てを待つ時間が短縮できる [1]．優先度入出力を関連付けているプロセスを優先度変更処理の対象とするか否か検討する．

（課題 1）と（課題 2）への対処により，性能調整入出力の調整精度への優先度入出力が存在する影響が小さくなると考えられる．また，（課題 3）への対処により，プロセッサの利用効率の低下を抑制できると考えられる．各課題について，以降の節で説明する．

3.2 優先度入出力が存在する影響を抑制する実 I/O 可否判断規則

文献 [1] と異なり，***Tender*** における資源「入出力」では優先度入出力を導入している [4]．そこで，実 I/O 可否判断規則を再検討した．検討後の実 I/O 可否判断規則を**図 2** に示し，以降で説明する．

(a) 入出力要求を発行したプロセスに関連付けている資源「入出力」が持つ入出力の程度から調整プロセスであるか否かを確認する．調整プロセスである場合，(b)

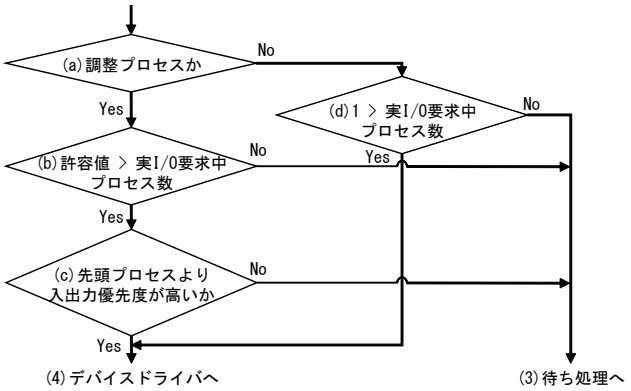

図 2 実 I/O 可否判断規則

に移行する．そうでない場合，(d) に移行する．

(b) 許容値と実 I/O 要求中プロセス数を比較する．許容値の方が大きい場合，(c) に移行する．そうでない場合，(3) 待ち処理を行う．

(c) 自身の入出力優先度と先頭プロセスの入出力優先度を比較する．自身の入出力優先度の方が高い場合，(4) デバイスドライバへ実 I/O 要求を発行する．そうでない場合，先頭プロセスを起床し，(3) 待ち処理を行う．

(d) 1 より実 I/O 要求中プロセス数が小さいか否か確認する．実 I/O 要求中プロセス数が 1 より小さい場合，(4) デバイスドライバへ実 I/O 要求を発行する．そうでない場合，(3) 待ち処理を行う．

(b) と (c) の制御により，実 I/O 要求中プロセス数が抑制され，入出力優先度の高いプロセスは，先に実 I/O 要求を発行する．

文献 [1] と比較し，追加した制御は (a) と (d) の制御である．(a) の制御により，プロセスに関連付けている資源「入出力」によって，異なる処理流れとする．(d) の制御では，性能調整入出力の調整精度への優先度入出力が存在する影響を小さくするため，実 I/O 要求中プロセス数が 0 の場合のみ，実 I/O 要求を許可する．

3.3 許容値算出の際に考慮するプロセス

文献 [1] では，許容値算出の際にデバイスドライバへ実 I/O 要求を発行している調整プロセスを考慮しない．これは，UNIX 系の OS に代表される多くの既存の OS において，同期型の入出力処理が採用されているためである．同期型の入出力処理の場合，プロセスが入出力処理中に新たな入出力要求を発行することはないため，許容値算出の際にデバイスドライバへ実 I/O 要求を発行している調整プロセスを考慮しない．

しかし，許容値算出の際にデバイスドライバへ実 I/O 要求を発行している調整プロセスを考慮しない場合，短期間に集中的な入出力処理を行うプロセスが高い入出力性能を要求した際に，調整精度が低下することがある．これは，実 I/O 可否判断処理において，高い入出力性能を要求する調整プロセスを考慮せず，多数の実 I/O 要求を許可してしまい，実 I/O 要求中プロセス数が増加するためである．これにより，高い入出力性能を要求するプロセスが実 I/O 処理を完了した後，入出力要求を短期間に再び発行すると，増加した実 I/O 要求中プロセス数の影響を受ける．

アプリケーションの I/O の特徴の調査 [5] により，短期間に集中的な入出力処理を行うプロセスが存在することがわかっている．プロセスが短期間に集中的な入出力処理を行うことによる調整精度の低下を抑制するため，文献 [1] と異なり，許容値算出の際にデバイスドライバへ実 I/O 要求を発行している調整プロセスを考慮する．

なお，許容値算出の方法は，文献 [1] と同じである．算出式を以下に示す．$\boldsymbol{P_i}$ は，すべての調整プロセスのうち上位 $\boldsymbol{i}$ 番目の要求入出力性能である．また，$\boldsymbol{k}$ は許容係数と呼ばれる値であり，許容値算出において考慮する調整プロセスの数を示す [1].

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

3.4 優先度変更処理の対象プロセス

優先度変更処理の対象を調整プロセスのみとした場合，プロセッサの利用効率は低下する．これは，調整プロセスのみ優先する処理があることにより，優先度入出力を関連付けているプロセスの資源利用が過剰に抑制されるためである．プロセッサ処理と入出力処理を交互に行う調整プロセス 1 つと優先度入出力を関連付けているプロセス 1 つを生成した場合を考える．優先度入出力を関連付けているプロセスのプロセス優先度の値は，調整プロセスのプロセス優先度と同じ値か小さい値とする．優先度入出力を関連付けているプロセスを優先度変更処理の対象としない場合，調整プロセスの調整精度のみ優先されるため，優先度入出力を関連付けているプロセスは資源利用を抑制され実 I/O 処理許可待ちとなる．この場合，調整プロセスが実 I/O 要求を発行している間にプロセッサを利用するプロセスが存在しなくなり，プロセッサの利用効率は低下する．

そこで，優先度入出力を関連付けているプロセスを優先度変更処理の対象とする．この場合，調整プロセスの調整精度と優先度入出力を関連付けているプロセスの実 I/O 要求が交互に優先される．これにより，調整プロセスが実 I/O 要求を発行している間に，優先度入出力を関連付けているプロセスが実 I/O 要求許可待ちではなくプロセッサ処理を行うため，プロセッサの利用効率の低下を抑制できる．

4. 評価

4.1 評価内容と評価環境

入出力性能調整機能により，プロセスが利用する入出力性能をどの程度調整できるかを示すため，以下の評価を

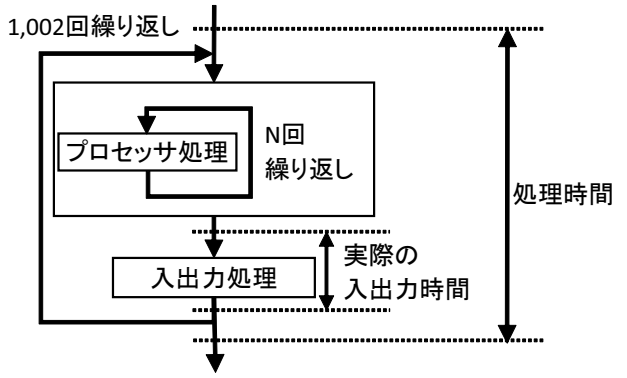

図 3 評価プログラムの処理流れ

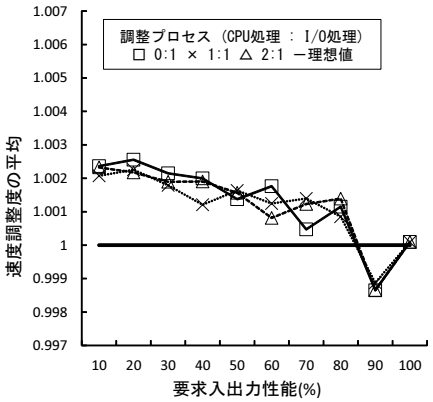

図 4 共存プロセスが存在しない場合の調整プロセスの速度調整度

行った．また，（評価 2）では，優先度入出力を関連付けているプロセス（以降，共存プロセスと略す）が存在する影響を明らかにする．

（評価 1）共存プロセスが存在しない場合の調整精度
入出力デバイスを利用するプロセスが調整プロセス 1 つのみの場合に調整プロセスに与える入出力性能をどの程度調整できるかを評価した．本評価により，共存プロセスが存在しない場合の入出力性能調整機能の調整精度を評価する．

（評価 2）共存プロセスが存在する場合の調整精度
入出力デバイスを利用するプロセスが調整プロセス 1 つと共存プロセス 1 つの場合に調整プロセスに与える入出力性能をどの程度調整できるかを評価した．本評価により，共存プロセスが存在する場合の入出力性能調整機能の調整精度を評価する．

評価に利用した評価プログラムの処理流れを**図 3** に示す．評価プログラムは，プロセッサ処理と入出力処理を交互に 1,002 回繰り返す．プロセッサ処理は，特定のメモリ領域の値のインクリメント処理とし，入出力処理は，磁気ディスク装置から 512 バイトを読み込む処理とする．この評価プログラムを走行させ，デバイスドライバでの入出力時間と実際の入出力時間を測定した．デバイスドライバでの入出力時間と要求入出力性能から理想の入出力時間を算出し，理想の入出力時間と実際の入出力時間から速度調整度を算出する．速度調整度により，プロセスが利用できる入出力性能をどの程度調整できたかを評価する．

速度調整度とは，文献 [2] において定義した評価尺度であり，デバイスドライバでの入出力時間を $\boldsymbol{T_d}$，実際の入出力時間を $\boldsymbol{T_r}$ とすると，以下の式で表される．なお，分母に示した式により，理想の入出力時間は算出される．

$$\text{速度調整度} = \frac{\boldsymbol{T_r}}{\boldsymbol{T_d} \times \frac{100}{\text{要求入出力性能}}} \qquad (2)$$

速度調整度は，1 に近いほど与えた入出力性能に処理速度を調整できていることを示し，1 を超えると与えた入出力性能以下，1 未満では与えた入出力性能以上に調整していることを示す．

また，評価では，評価プログラムの処理時間に含まれるプロセッサ処理に要する時間の割合と指定する要求入出力性能を変更して測定を行った．具体的には，プロセッサ処理のインクリメント処理の回数（N 回）を変更することにより，評価プログラムの処理時間に含まれるプロセッサ処理に要する時間と入出力時間の割合（以降，処理割合と略す）を CPU:I/O = 0:1, 1:1, および 2:1 と変化させた．なお，算出した 1,002 回分の速度調整度のうち，最初の 1 回と最後の 1 回を除いた 1,000 回分の速度調整度の平均を評価結果として利用した．

また，評価環境として，Core i7-4790S（3.20 GHz）プロセッサと磁気ディスク装置（東芝 DT01ACA050，容量 500 G バイト，7,200 RPM，32 M バイトキャッシュ）を搭載した計算機を利用した．また，***Tender*** は，4 コアのうち 1 コアのみ利用した．

4.2 共存プロセスが存在しない場合の調整精度

図 4 に共存プロセスが存在しない場合の調整プロセスの速度調整度を示す．図 4 から，いずれの要求入出力性能と処理割合においても，速度調整度の平均と理想値である 1 との差（以降，実測値と理想値の差と略す）が最大約 0.0026 であり，速度調整度の平均は，理想値である 1 に近い．このことから，共存プロセスが存在しない場合，入出力性能調整機能の調整精度は高いといえる．

4.3 共存プロセスが存在する場合の調整精度

4.3.1 共存プロセスからの影響

まず，入出力性能調整機能において，調整プロセスが利用する入出力性能を調整する際に共存プロセスの存在により受ける影響を以下に示す．

（影響 1）共存プロセスの実 I/O 処理完了待ち時間
調整プロセスが入出力要求を発行した際に共存プロセスが実 I/O 要求発行済みであった場合，デバイスドライバが到着順に実 I/O 要求を処理するため，調整プロ

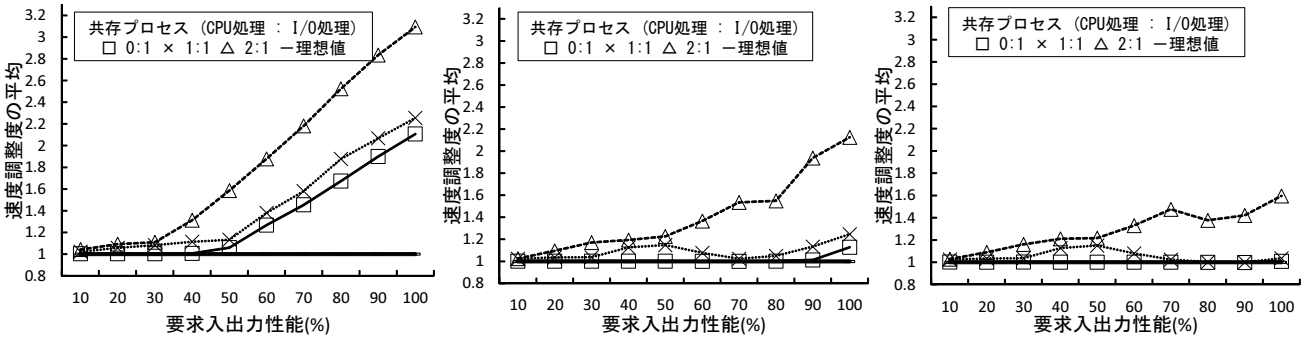

図 5 共存プロセスが存在する場合の調整プロセスの速度調整度（(影響 3）含む）

セスの実 I/O 要求発行は，共存プロセスの実 I/O 処理完了までの待ち時間の影響を受ける．

（影響 2）同期処理や遅延処理において起床した際のプロセッサ割当待ち時間
同期処理や遅延処理での遅延からの復帰によりプロセスが起床した際，起床してからプロセッサを割り当てられるまでの待ち時間が生じる．調整プロセスの実際の入出力時間は，調整プロセスが起床した際のプロセッサ割当待ち時間の影響を受ける．

（影響 3）入出力結果返却時のプロセッサ割当待ち時間
Tender において，実 I/O 処理を完了したプロセスとプロセス優先度が同じ値のプロセスが存在する場合，かつ優先度復元処理を行った場合，実 I/O 処理を完了したプロセスは，入出力結果返却が完了する前に，プロセッサの利用を中断される．このため，プロセッサを再び割り当てられるまでの待ち時間が生じる．調整プロセスの実際の入出力時間は，入出力結果返却時のプロセッサ割当待ち時間の影響を受ける．

入出力性能調整機能では，実 I/O 可否判断処理によって，実 I/O 要求中プロセス数を抑制し，（影響 1）を小さくしている．また，優先度変更処理によって，起床した際のプロセッサ割り当てを待つ時間を短縮し，（影響 2）を小さくしている．しかし，（影響 3）について，入出力性能調整機能では対処していない．この影響を明らかにするため，（影響 3）を含む場合と含まない場合で測定を行った．（影響 3）を含む場合では，実際の入出力時間として，入出力要求発行から入出力結果返却完了までの入出力時間を測定し，（影響 3）を含まない場合では，入出力要求発行から入出力結果返却前までの入出力時間を測定した．

4.3.2 入出力結果返却時のプロセッサ割当待ち時間を含む場合

図 5 に共存プロセスが存在する場合の調整プロセスの速度調整度（(影響 3）含む）を示す．なお，各プロセスのプロセス優先度は同じ値とし，共存プロセスの入出力の程度は優先度入出力の 0 とする．

図 5 の調整プロセスの処理割合が（A）0:1，（B）1:1，および（C）2:1 の場合を比較すると，調整プロセスのプロセッサ処理の割合が大きいほど実測値と理想値の差が小さく，速度調整度の平均は，理想値である 1 に近い．このことから，調整プロセスのプロセッサ処理の割合が大きいほど，入出力性能調整機能の調整精度が高くなる傾向があるといえる．これは，***Tender*** において，本評価の条件では，実 I/O 処理を完了したプロセスが入出力結果返却時に続けてプロセッサを利用できないためである．これにより，実 I/O 処理を完了したプロセスが続けて実 I/O 要求を発行することもない．つまり，調整プロセスのプロセッサ処理の割合が大きいほど，調整プロセスがプロセッサ処理をしている間に共存プロセスの入出力処理が完了し，調整プロセスが入出力処理を開始した際には実 I/O 要求を発行しているプロセスが存在しなくなる．このため，（影響 1）が小さくなり，調整精度は高くなる．

また，要求入出力性能が 30%以下の場合，実測値と理想値の差が小さく，速度調整度の平均は，理想値である 1 に近い．このことから，要求入出力性能が低いほど，入出力性能調整機能の調整精度は高くなる傾向があるといえる．

さらに，図 5 の共存プロセスの処理割合が□ 0:1，× 1:1，および△ 2:1 の場合を比較すると，共存プロセスのプロセッサ処理の割合が大きいほど実測値と理想値の差が大きく，速度調整度の平均は，理想値である 1 より大きくなる傾向がある．これは，実 I/O 処理を完了した調整プロセスが（影響 3）を受けているためである．調整プロセスがプロセッサを再び割り当てられ入出力結果返却が完了するのは，共存プロセスによるプロセッサ処理が完了した後である．このため，共存プロセスのプロセッサ処理の割合が大きいほど，調整精度が受ける影響は大きくなる．

以上から，調整プロセスのプロセッサ処理の割合が大きい場合や要求入出力性能が低い場合，入出力性能調整機能の調整精度は高くなる傾向がある．共存プロセスのプロセッサ処理の割合が大きいほど，（影響 3）を大きく受ける．次項では，（影響 3）を含まない場合の測定を行い，（影

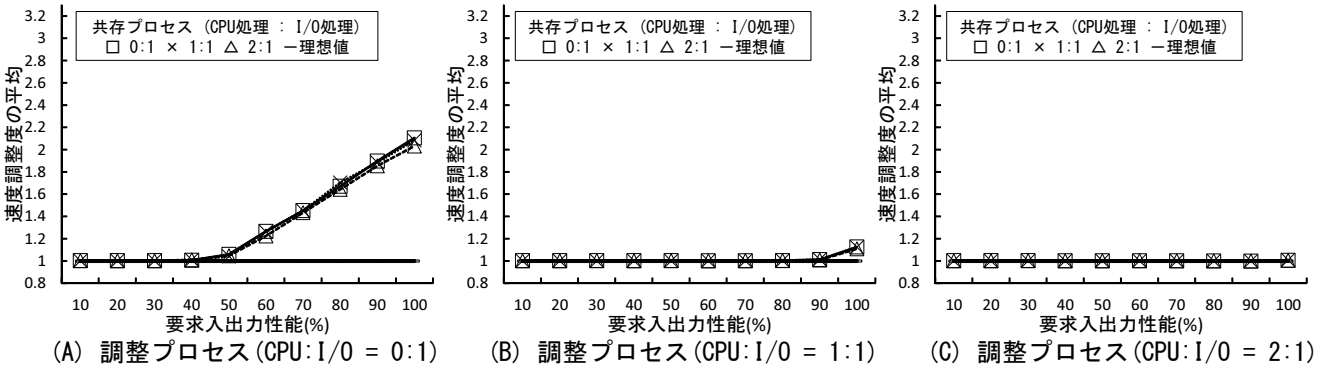

図 6 共存プロセスが存在する場合の調整プロセスの速度調整度（(影響 3）含まない）

響 3）を除いた入出力性能調整機能の調整精度を明らかにする．

4.3.3 入出力結果返却時のプロセッサ割当待ち時間を含まない場合

図 6 に共存プロセスが存在する場合の調整プロセスの速度調整度（(影響 3）含まない）を示す．なお，(影響 3）を含めないこと以外の条件は，4.3.2 項の条件と同じである．

図 5 と図 6 を比較すると，図 6 の方が実測値と理想値の差が小さく，速度調整度の平均は，理想値である 1 に近い．このことから，共存プロセスが存在する影響のうち，(影響 3）は大きいといえる．

また，図 6 の調整プロセスの処理割合が（A）0:1 の場合，要求入出力性能が高いほど実測値と理想値の差が大きく，速度調整度の平均は，理想値である 1 より大きくなる傾向がある．調整プロセスの入出力処理の割合が大きいほど，共存プロセスとの資源の競合が起きることが多くなり，(影響 1）が生じることが多くなる．(影響 1）により，式 (2）の T_r の値が実 I/O 処理 1 回分増加する．ここで，要求入出力性能が高いほど分母である理想の入出力時間が小さくなり，速度調整度は大きくなる．このため，(影響 1）を受ける場合，要求入出力性能が高いほど入出力性能調整機能の調整精度は低下する．

さらに，図 6 の共存プロセスの処理割合が□ 0:1，× 1:1，および△ 2:1 の場合を比較すると，いずれの要求入出力性能と調整プロセスの処理割合においても，共存プロセスのプロセッサ処理の割合にかかわらず，速度調整度の平均は，同程度の値を示している．これは，優先度変更処理によって，(影響 2）を小さくしているためである．これにより，(影響 3）を除くと，共存プロセスのプロセッサ処理による入出力性能調整機能への影響は小さいといえる．

5. 関連研究

既存の入出力スケジューリングに関する研究は，スループットの向上を対象とした研究と，入出力性能の保証を対象とした研究に分類できる．

スループットの向上を対象とした研究 [6]，[7] では，効率のよいディスクアクセスによりスループットの向上を実現している．一方，***Tender*** における資源「入出力」では，指定した入出力性能に調整することを目的としており，目的が異なる．

入出力性能の保証を対象とした研究は，デッドラインまでに入出力処理を完了することにより入出力性能を保証する研究と，プロセスが利用できる入出力性能を調停する研究に分類できる．

デッドラインまでに入出力処理を完了することにより入出力性能を保証する研究として，SCAN-EDF [8]，Fahrrad [9]，および Horizon [10] がある．これらの研究では，デッドラインが短いプロセスを優先することにより，プロセスの入出力性能を保証する．また，入出力性能の保証に加えて，スループットの向上も実現している．SCAN-EDF では，デッドラインを超過しない範囲で，シーク時間が短くなるように入出力要求の順番を入れ替える．Fahrrad では，デッドラインと入出力要求の発行頻度を考慮して，入出力要求の順番を入れ替える．Horizon では，入出力要求を保留する要求キューを 2 段にわけ，入出力要求の再整列と低遅延を要求するワークロードへの入出力性能の保証を行う．しかし，***Tender*** における資源「入出力」とは異なり，入出力優先度の低いプロセスによる入出力処理を抑制しない．このため，入出力優先度の低いプロセスの入出力処理による入出力優先度の高いプロセスの入出力時間の長大化を抑制できない．

プロセスが利用できる入出力性能を調停する研究として，RateWindows [11]，Arbitrator [12]，Maestro [13] がある．RateWindows では，単位時間当たりのファイル I/O 量を観測し，指定した入出力性能の閾値を越えないようにプロセスを遅延させる．閾値として，毎秒何 K バイトまでのファイル I/O を許可するかを指定する．このため，計算機性能とプロセスに含まれる入出力処理の割合から利用者が閾値を算出する必要がある．しかし，***Tender*** における資源「入出力」では，1 から 100%の間の数値を指定するの

みでよい．Arbitrator では，アプリケーションが入出力性能を保証されることにどの程度敏感であるかということを入出力性能の配分に導入している．敏感さに基づき，入出力性能を配分する．Maestro では，プロセス優先度に基づき，入出力性能を配分する．これらの研究では，入出力優先度の高いプロセスの入出力性能の保証を優先しない．一方，***Tender*** における資源「入出力」では，入出力優先度の高いプロセスの入出力性能の保証を優先する．重要なプロセスの入出力性能の保証を優先する研究として，Idletime Scheduling [14] がある．この研究では，バックグラウンドで実行するプロセスの資源利用を抑制することによりフォアグラウンドで実行するプロセスの実行速度を保証する．しかし，フォアグラウンドで実行するプロセスの実行速度を調整しない．一方，***Tender*** における資源「入出力」では，フォアグラウンドで実行するプロセスの入出力性能も調整できる．

また，***Tender*** における資源「入出力」に似た機能として，Cgroups [15] がある．Cgroups のサブシステムである blkio [16] は，重み付き比例配分と I/O スロットリングの制御ポリシーを提供する．重み付き比例配分では，入出力性能を配分するプロセス数が増減した際，個々のプロセスに配分する入出力性能が変化する．***Tender*** における資源「入出力」では，プロセス数が増減した場合にも，個々のプロセスに目的の入出力性能を与えることができる．I/O スロットリングでは，プロセスの入出力性能を目的の入出力性能に制限できる．しかし，入出力優先度の高いプロセスの入出力性能の調整を優先しない．

6. おわりに

Tender に資源「入出力」として実現している入出力処理の量を調整する制御法を評価した．評価により，***Tender*** に実現している入出力処理の量を調整する制御法の調整精度を示した．具体的には，共存プロセスが存在しない場合，調整精度の評価尺度である速度調整度の実測値と理想値の差が最大約 0.0026 であり，入出力処理の量を調整する制御法の調整精度は高いことを示した．また，共存プロセスが存在する場合の入出力処理の量を調整する制御法の調整精度を示し，調整精度が共存プロセスの存在から受ける影響を明らかにした．

残された課題として，既存の OS との比較評価やベンチマークによる評価といった詳細な評価がある．

参考文献

[1] 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 D，vol.J94-D，no.7，pp.1047–1057，2011.

[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（D-I），vol.J83-D-I，no.1，pp.184–193，2000.

[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] 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入出力性能調整機能の評価，”情報処理学会研究報告，vol.2012-OS-121，no.6，pp.1–8，2012.

[5] Yang Liu and Raghul Gunasekaran and Xiaosong Ma and Sudharshan S. Vazhkudai，“Automatic Identification of Application I/O Signatures from Noisy Server-side Traces，”Proceedings of the 12th USENIX Conference on File and Storage Technologies(FAST'14)，2014.

[6] Alexander Thomasian，“Survey and Analysis of Disk Scheduling Methods，”ACM SIGARCH Computer Architecture News，vol.39，no.2，pp.8–25，2011.

[7] Sitaram Iyer and Peter Druschel，“Anticipatory scheduling: A disk scheduling framework to overcome deceptive idleness in synchronous I/O，”Proceedings of the 18th ACM Symposium on Operating Systems Principles(SOSP'01)，vol.35，no.5，pp.117–130，2001.

[8] A.L.N. Reddy and Jim Wyllie and K.B.R. Wijayaratne，“Disk Scheduling in a Multimedia I/O System，”ACM Transactions on Multimedia Computing, Communications and Applications(TOMCCAP)，vol.1，no.1，pp.37–59， 2005.

[9] Anna Povzner and Tim Kaldewey and Scott Brandt and Richard Golding and Theodore M. Wong and Carlos Maltzahn，“Efficient Guaranteed Disk Request Scheduling with Fahrrad，”Proceedings of the 3rd ACM SIGOPS/EuroSys European Conference on Computer Systems 2008(EuroSys'08)，vol.42，no.4，pp.13–25，2008.

[10] Anna Povzner and Darren Sawyer and Scott Brandt，“Horizon: Efficient Deadline-Driven Disk I/O Management for Distributed Storage Systems，”Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing(HPDC'10)，pp.1–12，2010.

[11] Kyung D. Ryu and Jeffrey K. Hollingsworth and Peter J. Keleher，“Efficient Network and I/O Throttling for Fine-grain Cycle Stealing，”Proceedings of the 2001 ACM/IEEE conference on Supercomputing(SC'01)，2001.

[12] Quan Zhang and Dan Feng and Fang Wang and Yanwen Xie，“An Interposed I/O Scheduling Framework for Latency and Throughput Guarantees，”Journal of Applied Science and Engineering(JASE)，vol.17，no.2，pp.193–202，2014.

[13] Arif Merchant and Mustafa Uysal and Pradeep Padala and Xiaoyun Zhu and Sharad Singhal and Kang Shin，“Maestro: Quality-of-Service in Large Disk Arrays，”Proceedings of the 8th ACM International Conference on Autonomic Computing(ICAC'11)，pp.245–254，2011.

[14] Quan Zhang and Dan Feng and Fang Wang and Yanwen Xie，“Idletime Scheduling with Preemption Intervals，”Proceedings of 20th ACM Symposium on Operating Systems Principles(SOSP'05)，vol.39，no.5，pp.249–262，2005.

[15] Cgroups，https://www.kernel.org/doc/Documentation/cgroup-v1/cgroups.txt, accessed, Nov.1, 2016

[16] block I/O controller，https://www.kernel.org/doc/Documentation/cgroup-v1/blkio-controller.txt, accessed, Nov.1, 2016