情報処理学会研究報告
IPSJ SIG Technical Report

Vol.2009-ARC-183 No.25
Vol.2009-OS-111 No.25
2009/4/23

入出力性能を調整制御する機能の提案

長尾　尚[†1]　谷口秀夫[†2]

計算機ハードウェアの性能に左右されないでソフトウェアの実行速度を調整できれば，サービスの利便性は向上する．このため，我々は，プログラム実行速度を調整する方法として，プロセッサの性能を調整する方法と入出力性能を調整する方法を提案した．しかし，提案した入出力性能を調整する方法は，入出力要求の頻度の影響を受ける．そこで，本論文では，入出力要求の頻度の影響を受けない入出力性能の調整法を提案する．提案する制御法は，実入出力処理のひとつひとつをプロセスに割り当てる．具体的には，ひとつの実 I/O 時間を入出力スロットと名付け，プロセスに割り当てる入出力スロットの割合を調整する．

Proposal of Function for Controlling Regulating I/O Performance

Takashi Nagao [†1] and Hideo Taniguchi [†2]

If execution speed of software is regulated without concerning by performance of the computer hardware, Convenience of the service will become better. Before now, We proposed a mechanism of regulating processor performance and a mechanism of regulating I/O performance. However, a mechanism of regulating I/O performance is affected by outbreak frequency of I/O request. Therefore, We propose a mechanism unaffected by outbread frequency of I/O request. Proposed mechanism assigns each of read I/O process to a process. Specifically, we call one of read I/O time I/O slot. And proposed mechanism control percentage of assigning it to a process.

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

1. はじめに

計算機ハードウェアの性能向上が著しいため，低性能から高性能まで様々な性能をもつ計算機が同時に存在している．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，例えば，同じプログラムであっても表示速度が計算機ごとに大きく異なってしまい，利便性が低下することがある．また，ソフトウェア教材は，学習者が内容を理解するためにアニメーションで視覚的に説明を行う．このため，利用者は，見やすい速度でゆっくり動かしたい場合もあり，また慣れれば高速に動かしたい場合もある．つまり，利用者は，サービス処理速度を自分の目的に合わせて自由に変更したい．しかし，サービス処理速度を調整する機能を各ソフトウェアに組み込むことは，大きな工数を必要とする．

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，利用者の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時間を調整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．現在までに，プロセッサ性能を調整するスケジュール法[1] や入出力性能を調整する制御法[2),3)] を提案した．しかし，文献 1) で提案した制御法は，入出力処理の割合が大きいプログラムの実行速度を調整できない．また，文献 2),3) で提案した制御法は，調整の精度が入出力要求の頻度の影響を受ける．

そこで，入出力要求の頻度の影響を受けない入出力性能の調整制御法を提案する．提案する制御法は，実入出力処理のひとつひとつをプロセスに割り当てる．具体的には，ひとつの実 I/O 時間を入出力スロットと名付け，プロセスに割り当てる入出力スロットの割合を調整する．

I/O スケジューリングについては，入出力要求の順番を入れ替えることでスループットを調整しディスク性能を向上させる研究[5]，実時間性を保証する I/O スケジューリング法の研究[6),7)] がある．また，プロセッサスケジューリングについては，CPU 使用の予約と使用時間の制御によりスケジューリングを予想可能にする研究[8]，多重化 I/O の実行間隔制御においてスケジュール操作を行うことにより CPU 資源を効率的に使用する研究[9] がある．これらの研究は，ハードウェア性能を最大限に引き出す．これらに対し，本制御法は，要求された入出力性能をプロセスに提供し，プログラムの実行速度を調整する．

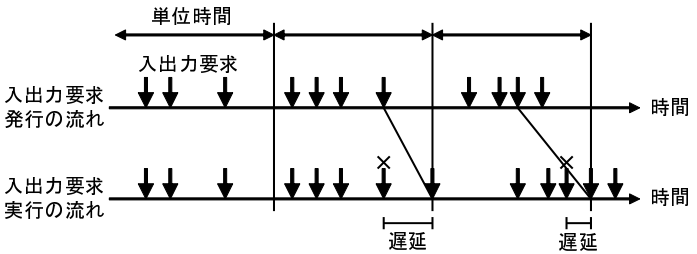


図 1 入出力回数の調整（入出力処理 3 回/単位時間）

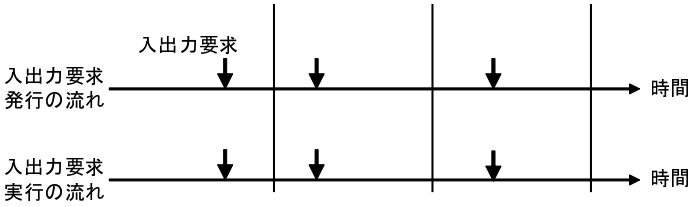


図 2 入出力回数の調整における問題

2. 既存手法の問題

2.1 入出力回数を調整する方法

入出力回数を調整する方法[2] を図 1 に示す．入出力回数を調整する方法は，単位時間における入出力の回数を調整する．具体的には，調整のために指定された入出力回数を上限回数とし，単位時間内の入出力回数を上限回数以下に合わせるため，入出力要求を遅延して実行することにより，入出力の開始時期を調整する．しかし，この方法は，上限回数を超える入出力要求のみ遅延するため，以下の問題がある．

(問題 1) 入出力要求が少ない場合は調整できない．

具体的には，被調整プロセスが単位時間内で上限回数以下の入出力要求を発行すると性能調整用の遅延処理が実行されない．この様子を図 2 に示す．図 2 において，単位時間内の要求された入出力回数を 3 回とする．ここでは，単位時間内で 3 回を超える入出力要求は発生しないため，遅延は発生しない．このため，プロセスに提供される入出力性能は入出力デバイスのハードウェア性能そのものになってしまう．

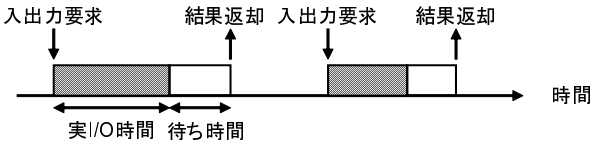


図 3 入出力時間の調整

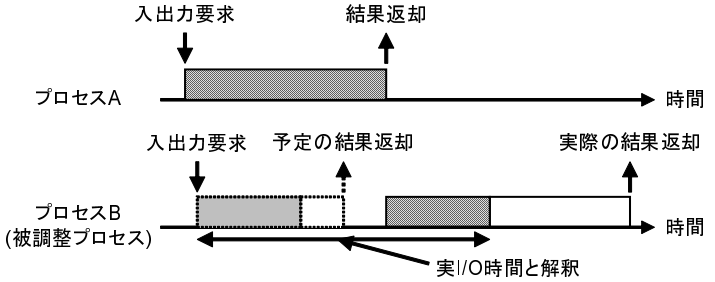


図 4 入出力時間の調整における問題

2.2 入出力時間を調整する方法

入出力時間を調整する方法[3] を図 3 に示す．入出力時間を調整する方法は，一つの入出力に要する時間を調整する．具体的には，実 I/O 処理に遅延処理を加えることで，入出力の終了時期を調整する．したがって，この方法は，各入出力要求ごとにその時間を調整できるため，きめ細かな入出力性能の調整が可能である．しかし，この方法は，実行した実 I/O 処理時間は 100 %の性能で実行と仮定して調整を行うため，以下の問題がある．

(問題 2) 他プロセスの入出力要求の影響を受ける．

具体的には，他プロセスの入出力要求い基づく実 I/O 処理により被調整プロセスの実 I/O 処理が待たされ，遅延処理が長くなる．この様子を図 4 に示す．図 4 において，プロセス A の実 I/O 処理により，被調整プロセス B の実 I/O 処理は待たされる．このため，被調整プロセス B の実 I/O 時間は，入出力要求からプロセス A の実 I/O 処理終了，さらに被調整プロセス B の実 I/O 処理終了までと解釈される．この結果，遅延処理を長く決定，実行してしまう．

図 4 においてプロセス B の入出力要求の実行開始は，プロセス A の入出力要求による実 I/O 処理終了まで待たされる．この待ち時間が調整により加えられる待ち時間を超える場

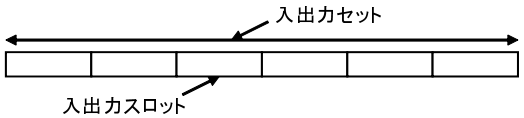


図 5 入出力スロットと入出力セット

合，プロセス B の入出力要求による実 I/O 処理が終了した時点で，調整予定の結果返却の時刻を超過する．

3. 入出力性能の調整方法

3.1 基 本 方 式

2 章で述べたように入出力回数を調整する方法においては，調整の精度が自プロセスの入出力要求の頻度に影響される．また，入出力時間を調整する方法においては，調整の精度が他プロセスの入出力要求の頻度に影響される．つまり，両方法とも，調整の精度が入出力要求の頻度の影響を受ける．

そこで，実入出力処理のひとつひとつをプロセスに割り当てる制御法を提案する．具体的には，図 5 に示す入出力スロットと入出力セットの概念を導入する．入出力スロットは，ひとつの実 I/O 時間である．入出力セットは，複数の連続する入出力スロットをまとめたものである．入出力セットを単位として，プロセスに入出力スロットを割り当てる割合を調整する．プロセスは，割り当てられた入出力スロットにおいて実 I/O を実行する．

本制御法における調整制御された入出力処理の様子を図 6 に示す．ここで，プロセス A は要求性能 33 %の被調整プロセス，プロセス B は調整されない共存プロセスとする．プロセス A の入出力要求は，割り当てられた入出力スロットまで遅延して実行する．一方，プロセス B の入出力要求は，被調整プロセスに割り当てていない入出力スロット（以降，空入出力スロットと呼ぶ）で実行する．このため，被調整プロセスが頻繁に入出力要求を発行しない場合でも，要求性能に合わせた遅延を行うことが可能である．また，他プロセスが頻繁に入出力要求を発行する場合でも，被調整プロセスの入出力要求は割り当てられた入出力スロットで実行されるため，調整できる．

3.2 制 御 方 式

本制御法を実現する制御処理の流れを図 7 に示し，以降に説明する．

(1) 入出力スロット位置とその開始時刻，および現在時刻から現在の入出力スロットを特定する．

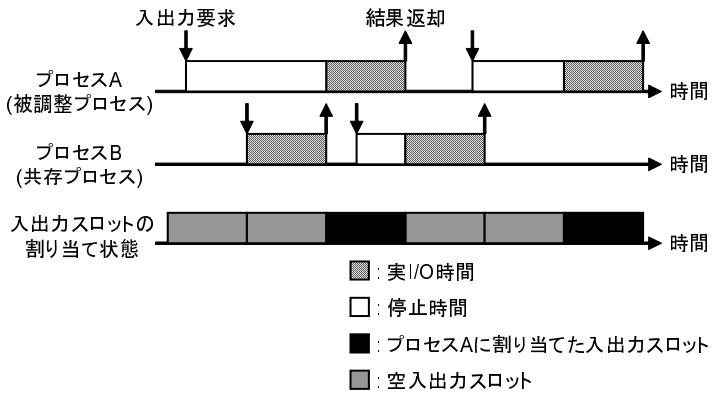


図 6 入出力スロットを用いた入出力性能調整

(2) 自プロセスが被調整プロセスか共存プロセスか判別し，以下の処理を行う．

(A) 被調整プロセスであれば，現在の入出力スロットが自プロセスに割り当てられているか判別し，以下の処理を行う．

(a) 現在の入出力スロットが自プロセスに割り当てられているならば，実 I/O 処理中のプロセス数を参照し，以下の処理を行う．

(i) 実 I/O 処理中のプロセス数が 0 でないなら，被調整プロセス用の待ちキューに自プロセスを追加し，待ち状態となる．

(b) 現在の入出力スロットが自プロセスに割り当てられていないならば，次に割り当てられている入出力スロットまでの時間だけ待ち状態となる．

(B) 共存プロセスであれば，現在の入出力スロットが空入出力スロットであるか判別し，以下の処理を行う．

(a) 現在の入出力スロットが空入出力スロットであれば，実 I/O 処理中のプロセス数を参照し，以下の処理を行う．

(i) 実 I/O 処理中のプロセス数が 0 でないなら，共存プロセス用の待ちキューに自プロセスを追加し，待ち状態となる．

(b) 現在の入出力スロットが空入出力スロットでなければ，次に割り当てられている入出力スロットまでの時間だけ待ち状態となる．

(2) 実 I/O 処理中プロセス数を 1 増やす.
(3) 実 I/O 処理を開始する.
(4) 実 I/O 処理中プロセス数を 1 減らす.
(5) 実 I/O 時間から入出力スロット時間を更新する.
(6) 入出力スロット位置を更新する.
(7) 実 I/O 処理中のプロセス数を参照し，以下の処理を行う.
(A) 実 I/O 処理中のプロセス数が 0 ならば，被調整プロセス用の待ちキューを参照し，以下の処理を行う.
(a) 被調整プロセス用の待ちキューが空でないなら，キューの先頭のプロセスの処理を再開させる.
(b) 被調整プロセス用の待ちキューが空なら，共存プロセス用の待ちキューを参照し，以下の処理を行う.
(i) 共存プロセス用の待ちキューが空でないなら，キューの先頭のプロセスの処理を再開させる.

3.3 課題と対処

3.3.1 課　　題

本制御法には，大きく 2 つの課題がある.
(課題 1) 入出力スロット時間の決定法
(課題 2) 入出力スロットの割り当て方式
各課題について対処法を以降に述べる.

3.3.2 入出力スロット時間の決定法

本制御法は，現在の入出力スロットが利用可能でない場合，次に利用可能な入出力スロットまで時間待ち状態となる．この時間は，入出力スロット時間から算出する．このため，実際の実 I/O 時間と入出力スロット時間の差が大きくなると，調整の精度は低下する．したがって，この差を小さくすることが重要である.

実 I/O 時間は変動するため，入出力スロット時間を随時更新する必要がある．そこで，過去の実 I/O 時間から入出力スロット時間を更新し決定する．更新し決定する方法として，以下の方法がある．ここで，i 回目の実 I/O 時間を I_i，このときに決定する入出力スロット時間を T_i とする.

(1) 直前法：直前の実 I/O 時間を利用

$$T_n = I_n$$

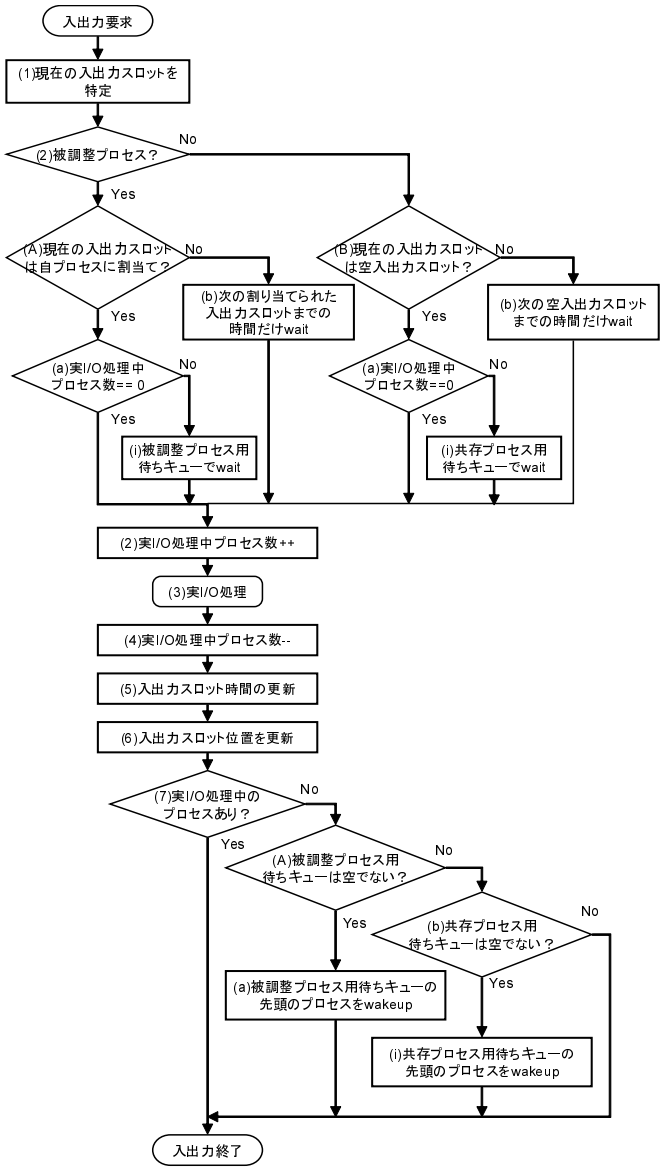


図 7 制御処理の流れ

(2) 平均法：最近 m 回の実 I/O 時間の算術平均を利用

$$T_n = \frac{1}{m}\sum_{k=1}^{m} I_{n-k+1}$$

(3) 中央値法：以前の全ての実 I/O 時間の中央値を利用

$T_n = Median\{I_1 ... I_n\}$

(4) フィードバック法 (FB 法)：直前の入出力スロット時間に実 I/O 時間と直前の入出力スロット時間の差に重み W を乗じた値を加算した値を利用

$T_n = T_{n-1} + (I_n - T_{n-1}) * W$

各方法の比較を表 1 に示す．直前法は，直前の実 I/O 時間をそのまま入出力スロット時間とするため，計算量は各方法の中で最も少ない．一方，キャッシュヒットにより極端に短くなった実 I/O 時間をそのまま入出力スロット時間に反映するため，外れ値の影響を受けやすい．平均法は，外れ値の影響を抑えることができる．また，入出力デバイスの性能変化による実 I/O 時間の変化を入出力スロット時間に反映できる．一方，過去 n 回の実 I/O 時間の算術平均をとる必要があるため，計算量は多い．中央値法は，中央値をとることから，各方法の中で最も外れ値の影響を受けにくい．一方，過去の実 I/O 時間を大小関係に基づき整列する必要があるため，計算量は多い．また，実 I/O 時間が変化しても，変化後の実 I/O 時間が中央値になるまでは入出力スロット時間に反映されない．FB 法は，実 I/O 時間と入出力スロット時間との差を次の入出力スロット時間に反映させることから，実 I/O 時間の変化に対応できる．また，重みを制御することで外れ値の影響を抑えることができる．さらに，過去の実 I/O 時間を保存する必要はないため，計算量は少ない．一方，実 I/O 時間の変化に対応でき，かつ，外れ値の影響を抑えることができる重みを事前に決定する必要がある．

各方法を定量的に比較するため，磁気ディスク入出力を取り上げ，実測評価を行う．具体的には，1 セクタ（512B）単位のランダムアクセスを 1001 回行った際の実 I/O 時間で評価した．このときのアクセスパターンは，各アクセス間にランダム長の cpu 処理（具体的には，0〜20ms のループ処理）の有無と読み書きを組み合わせた 4 パターンである．なお，平均法は $m = 100$，FB 法は重み W =0.1〜0.9 まで 0.1 刻みで変化とした．

各方法について精度と安定性を評価する．このため，精度と安定性は，それぞれ算出した入出力スロット時間と実 I/O 時間の差について，平均と分散で評価する．平均が小さいものは，精度が高く，分散が小さいものは安定性が良いと判断できる．

表 1 入出力スロット時間の決定法の比較

方法	長所	短所
直前法	(1) 計算量が少ない	(1) 外れ値の影響を受けやすい
平均法	(1) 実 I/O 時間の変化に対応できる	(1) 計算量が多い
中央値法	(1) 外れ値の影響を受けにくい	(1) 計算量が多い (2) 実 I/O 時間の変化に対応しにくい
FB 法	(1) 実 I/O 時間の変化に対応できる (2) 外れ値の影響を抑制できる (3) 計算量が少ない	(1) 重みを事前決定する必要がある

測定結果を図 8 と図 9 に示す．図 8 と図 9 より，各方法の特徴を以下に述べる．

(1) 直前法は，読み込み処理の場合，精度が最も良いものの，安定性が最も低い．なお，書き込み処理の場合，精度と安定性は最も悪い．

(2) 平均法は，読み込み処理の場合，精度が最も悪い．なお，書き込み処理の場合，精度が最も良い．

(3) 中央値法は，読み込み処理の場合，平均法に次いで精度が悪い．

(4) FB 法は，読み込み処理の場合，重み W が小さいほど安定性は高く，重み W が大きいほど精度が良い．なお，書き込み処理の場合，重み W が小さいほど精度と安定性が良くなる．特に重み W を 0.1 とした場合，最も安定性が高く，平均法に次いで精度が良い．

これらの結果から，読み込み処理と書き込み処理のどちらでも精度と安定性が良い FB 法（$W = 0.1$）を採用する．

3.3.3 入出力スロットの割り当て方式

本制御法では，要求性能に合わせて被調整プロセスに入出力スロットを割り当てる．複数の被調整プロセスが存在する場合，割り当てる入出力スロット位置が衝突することがある．この場合，入出力スロットの割り当て方式が問題となる．ここでは，処理の均一性が高い改良擬似周期割当て方式[4] を採用する．なお，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ものである．

改良擬似周期割当て方式は，要求性能に基づき n 個の入出力スロットをプロセスに割り当てる場合，入出力セットの先頭から最初の空き入出力スロットを起点として，i 番目に割り当てる入出力スロットの位置を (総入出力スロット数)$*(i-1)/n$ とする方式である．

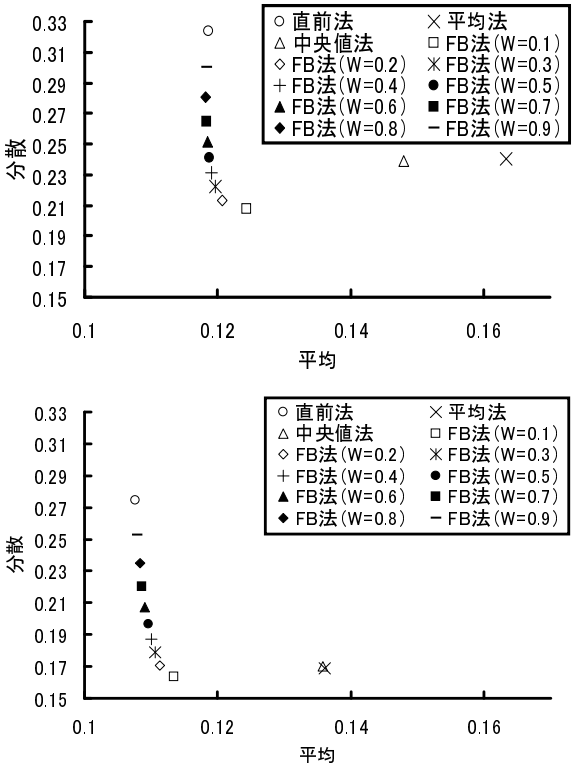


図 8 (算出した入出力スロット時間－実 I/O 時間) の平均と分散（読み込み）

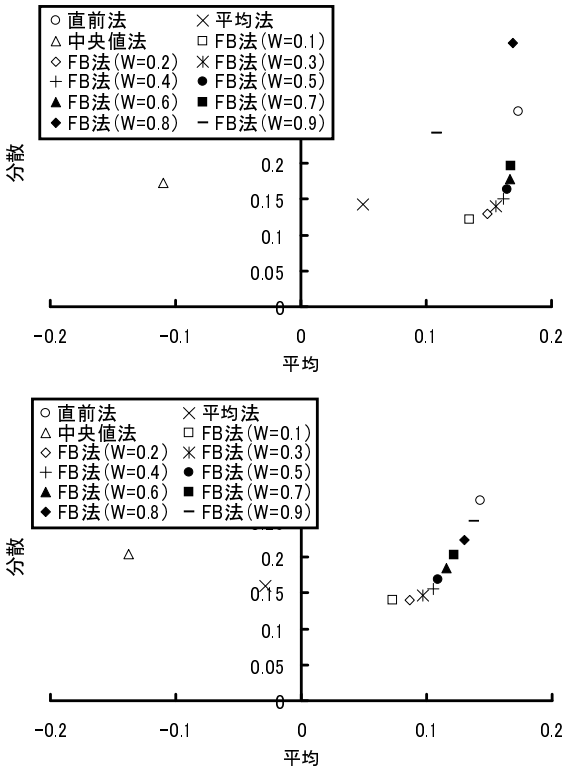


図 9 (算出した入出力スロット時間－実 I/O 時間) の平均と分散（書き込み）

4. 実装と評価

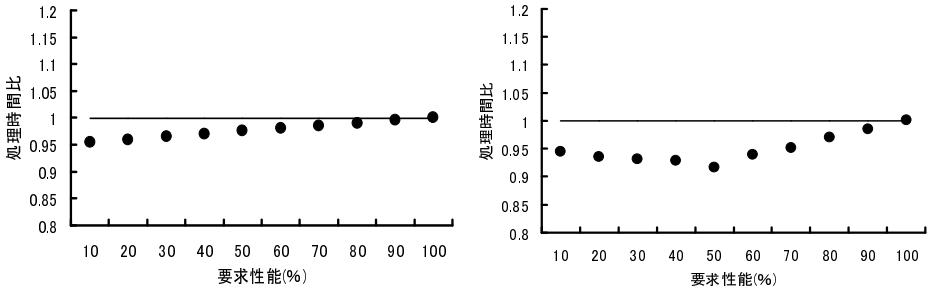
4.1 測定環境と評価プログラム

本制御法を FreeBSD 6.3-RELEASE 上に実装し，Celeron(1.8GHz) プロセッサを搭載した計算機で評価した．測定は，シングルユーザモード環境である．評価プログラムは，入力処理と CPU 処理をくり返す．入力処理は，磁気ディスク装置から raw デバイス形式で 512 バイト読み込む．読み込み位置はランダムである．CPU 処理は，特定のメモリ領域の値をインクリメントする処理である．CPU 処理の時間を変化させて入力処理と CPU 処理の比率を設定する．なお，入力処理の際に実 I/O 時間を測定し，調整の精度の評価に利用する．

4.2 評価と考察

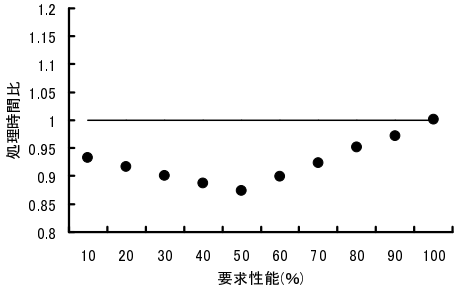
入出力要求の頻度が本制御法に与える影響を明らかにするため，入力処理と CPU 処理の割合を変化させ，評価を行った．

入力処理と CPU 処理の割合が，1:0，4:1，2:1，1:1，1:2，および 1:4 の場合について，要求性能を 10 %から 100 %まで，10 %刻みで変化させて実測した．実測結果を図 10 に示す．なお，処理時間比は，

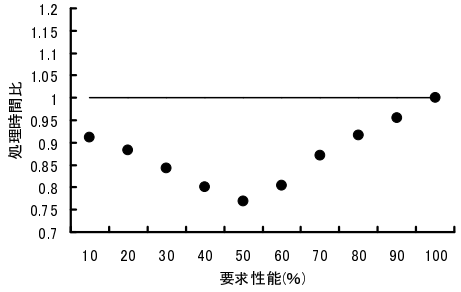


(A)I/O:CPU = 1:0

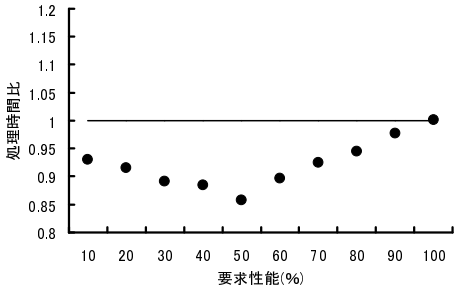
(B)I/O:CPU = 4:1



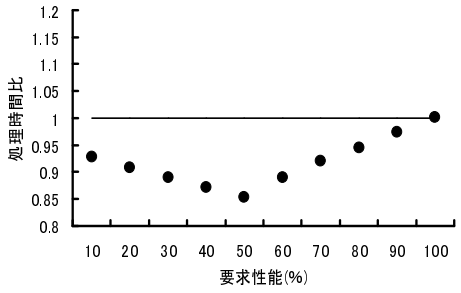
(C)I/O:CPU = 2:1



(D)I/O:CPU = 1:1



(E)I/O:CPU = 1:2



(F)I/O:CPU = 1:4

図 10 入出力の頻度による影響

$$\text{処理時間比} = \frac{\text{実測値}}{\text{理論値}} \tag{1}$$

である．実測値は測定した入出力時間であり，理論値は当該の要求性能で理想的に入出力性能の調整を行ったと仮定して算出した入出力時間である．また，処理時間比が 1 より大きい場合，入出力要求が理論値よりも長時間で実行された場合である．一方，1 より小さい場合，入出力要求が理論値よりも短時間で実行された場合である．被調整プロセスに対し各要求性能で入力処理を行った場合の理論値を 1 とした場合の処理時間である．

図 10 より以下のことがわかる．

(1) 大半が入出力処理の場合 (図 10(A))，処理時間比の最大誤差は 5 %未満であり，要求性能に関わらず調整の精度が良い．また，要求性能が低いほど調整の精度は悪くなる．これは，次の理由による．要求性能が低い場合，割り当て入出力スロットの間隔は大きくなる．つまり，割り込て入出力スロット間に多くの空入出力スロットが存在する．このとき，割り当て入出力スロットまでの待ち時間は,（入出力スロット時間）×（空入出力スロット数）で算出する．一方，入出力スロット時間は，実 I/O 時間と同一ではない．このため，入出力スロット時間と実 I/O 時間の差が空入出力スロット数分だけ積算されて誤差となるためである．

(2) 入力処理と CPU 処理の割合が同じ場合 (図 10(D)) では，要求性能が 50 %のとき，調整の精度が最も悪い．これは，次の理由による．要求性能が 50 %のとき，割り当て入出力スロットと空入出力スロットは交互になる．また，CPU 処理時間と入出力スロット時間は同じである．このため，CPU 処理が終了する時刻には，現在の入出力スロット位置が次の入出力スロットに更新される．このとき，入力処理を行うと，現在の入出力スロットが割り当て入出力スロットとなり，遅延処理が行われないためである．

(3) 図 10(B)〜(F) において，要求性能が 50 %以下の場合，調整の精度は要求性能が高くなるほど悪くなる．一方，要求性能が 50 %以上の場合，調整の精度は要求性能が低くなるほど悪くなる．これは，次の理由による．要求性能が 50 %以下の場合，割り当て入出力スロット間には空入出力スロットが存在する．また，(2) で述べたとおり，CPU 処理により次の割り当て入出力スロットまでの時間が短くなる．このため，CPU 処理時間に対する割り当て入出力スロット間の時間の比が小さいほど，CPU 処理の影響を大きく受けるためである．一方，要求性能が 50 %以上の場合，要求性能が低いほど，割り当て入出力スロット間に空入出力スロットが存在する場合が多くなる．このため，入力処理において割り当て入出力スロットまで待つ場合が増えるため，CPU 処理時間

による影響を受けやすくなるためである．

(4) CPU 処理の割合が低い場合 (図 10(B)〜(C)) は CPU 処理の割合が高い場合 (図 10(E)〜(F)) に比べ，調整の精度は良い．これは CPU 処理時間が短くなるため，次の割り当て入出力スロットまでの時間に与える影響が低いためである．

以上のことから，本制御法は入力処理の割合が CPU 処理の割合と比べて高いとき，入出力性能を調整できているといえる．

5. おわりに

入出力性能を調整する既存手法の問題点を明らかにし，入出力要求の頻度の影響を受けない入出力性能の調整制御法を提案した．

入出力回数を調整する方法は，入出力要求が少ない場合は調整できない．また，入出力時間を調整する方法は，他プロセスの入出力要求の影響を受ける．これらに対し，提案する制御法は，ひとつの実 I/O 時間を入出力スロットとする概念を導入し，プロセスに入出力スロットを割り当てる割合を調整する．プロセスの入出力要求の実行は，そのプロセスが利用可能な入出力スロットで実行される．これにより，上記の問題を解決している．

実現時の課題として，入出力スロット時間の決定法，および入出力スロットの割り当て方式を示し，その対処法について述べた．入出力スロット時間の決定法は実測評価により，直前の入出力スロット時間に実 I/O 時間と直前の入出力スロット時間の差に重み W を乗じた値を加算した値を利用する FB 法 ($W = 0.1$) を採用する．また，入出力スロットの割り当て方式は，処理の均一性の高い改良擬似周期割当て方式[4] を採用する．

さらに，本制御法を実装し評価した．調整の精度は，処理全体に占める入出力処理の割合が高い場合に良く，入出力処理の割合が低くなると低下する．また，多くの場合において，要求性能が 50 %のときに調整の精度が低下する．

残された課題として，調整の精度の向上がある．

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 ”，信学論 (D-I)，Vol.J81-D-I，No.4，pp.386–392，1998.
2) 谷口 秀夫，坂口 修：“ 入出力回数の制御によりサービス時間を調整する制御法 ”，信学論 (D-I)，Vol.J81-D-I，No.11，pp.1211–1218，1998.
3) 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 ”，信学論 (D-I)，Vol.J83-D-I，No.5，pp.469–477，2000.
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 ”，情処学論，Vol.41，No.6，pp.1745–1754，2000.
5) Anna Povzner，Tim Kaldewey，Scott Brandt，Richard Golding，Theodore M. Wong，Carlos Maltzahn，“ Efficient Guaranteed Disk Request Scheduling with Fahrrad，”EuroSys ’08，pp.13-25，(2008.04)
6) H.P.Chang，R.I.Chang，W.K.Shih，R.C.Chang “ GSR: A global seek-optimizing real-time disk-scheduling algorithm，”The Journal of Systems and Software，pp.198-215，(2007)
7) S.A.Brandt，S.Banachowski，C.Lin，and T.Bisson.“ Dynamic integrated scheduling of hard real-time,soft real-time and non-real-time processes，”Real-Time Systems Systems2003，pp.396-407，(2003.12)
8) M.B.Jones，D.Rosu，and M.-C.Rosu，“ CPU reservations and time constraints: Efficient，predictable scheduling of independent activities，”Symposium on Operating Systems Principles，pp.198-211，(1997.10)
9) 河合栄治，門林雄基，山口　英，“ 多重化 I/O の実行間隔制御におけるスケジュール操作による確定的なプロセッサ利用の実現，”情報処理学会研究報告, 2003-OS-93，vol.2003，No.42, pp.33-40 (2003).