Vol.2019-OS-145 No.2
2019/2/28

入出力スループットの低下を抑制する入出力性能の調整法の実現と評価

長尾尚[†1]　田辺雅則[†1]　横山和俊[†2]　谷口秀夫[†1]

概要：計算機のハードウェア性能やプロセスの動作に関わらず，利用者がプロセスの実行速度を制御できれば，計算機の利便性が向上する．これまで，著者らはプロセッサ性能や入出力性能の調整法を提案した．ここで，入出力性能の調整法では，調整対象プロセスが要求する入出力性能を保証するために，入出力デバイスに発行できる他プロセスの入出力要求数を制限する．しかし，従来方式では各調整対象プロセスが要求する入出力性能の合計性能を保証するために，入出力デバイスに発行する入出力要求数を過少に抑えてしまう．この結果，入出力デバイスが入出力要求の受け取り待ちとなり，入出力スループットが低下する．本稿では，他の調整対象プロセスの入出力要求による待ち時間の増加分だけ入出力デバイスに発行できる入出力要求数を制限する方式を提案する．評価により，提案方式が調整対象プロセスの入出力性能を精度良く調整し，かつ入出力スループットを向上できることを示す．

キーワード：入出力スケジューラ，性能調整，オペレーティングシステム

Implementation and Evaluation of I/O Performance Regulating Method to Suppress Decrease in I/O Throughput

TAKASHI NAGAO[†1]　MASANORI TANABE[†1]
KAZUTOSHI YOKOYAMA[†2]　HIDEO TANIGUCHI[†1]

Abstract: If user can control execution speed of the process independent of hardware performance and other process, we can improve convenience of the computer. We proposed performance regulating method for processor or I/O device. The proposed method for I/O device limits the number of I/O request for a I/O device in order to guarantee regulation process's I/O response time. However, the proposed method tries to reserve total requested I/O performance of all regulation processes. And, the number of I/O request for the I/O device may be too small. As a result, the I/O device wait I/O request from OS, the I/O throughput decreases. In this paper, we propose calculation method of the limit number by considering the waiting time which increases according to requests from other regulation processes. We show that proposed method can increase I/O throughput and guarantee regulation process's I/O response time.

Keywords: I/O Scheduling, Regulating Performance, OS

1. はじめに

計算機の性能向上に伴い，利用者は一台の計算機上で複数のソフトウェアを同時に動作させる．例えば，「ウィルス対策ソフトウェアにファイルを検査させつつ，動画を視聴しながら文章を作成する」といった利用が挙げられる．このとき，他のソフトウェアによって，利用者が直接操作するソフトウェアの入力受付や表示が滞ると，利用者がストレスを感じる場合がある．利用者がソフトウェアを起動するとき，計算機内ではオペレーティングシステム（以降，OS と呼ぶ）が当該ソフトウェアに対応するプロセスを生成し，ソフトウェアの動作を管理する．具体的には，OS はどのプロセスにどの時間だけプロセッサを割当てるのか，プロセスの入出力要求をどの順番で入出力デバイスに発行するのか，を制御する．この制御によって他プロセスが優先されると，利用者が直接操作するソフトウェアの動作が遅くなることがある．

他のプロセスに関わらず特定のプロセスを一定の速度で処理できれば，ソフトウェアの動作が安定し，計算機の利便性が向上する．これまで，著者らは，プロセッサ性能の調整法[1]や入出力性能の調整法[2]を提案した．この入出力性能の調整法は，利用者が指定する性能（以降，要求入出力性能と呼ぶ）に合わせて，プロセスの入出力の処理時間（以降，入出力時間と呼ぶ）を調整する．具体的には，要求入出力性能が示す時間まで，プロセスへの入出力処理の完了通知を待たせることで，入出力時間を一定に保つ．さらに，OS が入出力デバイスに発行する入出力要求数を許容値と呼ぶ値以下に制限することで，調整対象のプロセスの調整精度を保っている．つまり，要求入出力性能を保証している．

文献[2]では，調整対象プロセスが複数ある場合，これらの合計の要求入出力性能を保証するように許容値を算出する．しかし，この方法で算出する許容値は，各調整対象プロセスの入出力時間の保証に必要な値よりも小さい場合が

†1 岡山大学大学院自然科学研究科
†2 高知工科大学情報学群

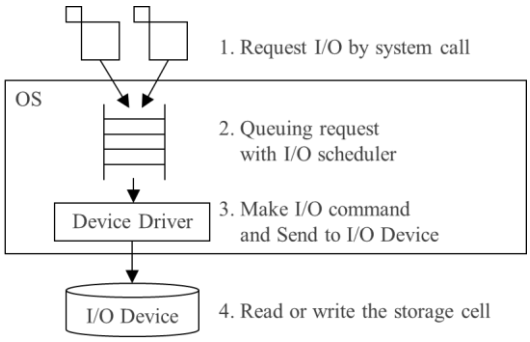


図 1　入出力処理の流れ
Figure 1　Overview of I/O.

ある．つまり，入出力デバイスが同時に受け取る入出力要求数が過少となる場合がある．一方，入出力デバイスは入出力要求を絶え間なく処理することで，入出力スループットを向上させている．このため，入出力デバイスが同時に受け取る入出力要求数が少ないと，新たな入出力要求を受け取る前に全ての入出力要求を処理してしまい，入出力スループットの低下を招く．

本稿では，調整対象プロセスの入出力時間を精度良く調整し，かつ入出力スループットを向上できる許容値の算出方法を提案する．提案方式では，調整対象プロセスごとに入出力時間の保証に必要な許容値を算出することで，許容値を大きくする．具体的には，それぞれの要求入出力性能から調整対象でないプロセスの入出力要求による待ち時間の許容量を算出する．そして，この時間から他の調整対象プロセスの入出力要求が先に処理されることで生じる待ち時間だけ差し引き，許容値を算出する．評価にて，提案方式は，調整対象プロセスの入出力時間を精度良く調整し，かつ入出力スループットを向上できることを示す．

2. 入出力処理の流れ

プロセスは入出力デバイス内のデータを読み書きする場合，OS が提供する入出力用のシステムコールを呼び出す．このシステムコールの処理流れを図 1 に示す．OS は当該入出力デバイス用の待ちキューにプロセスの入出力要求を格納し，プロセスを待ち状態にする．待ちキュー内の入出力要求は，OS の入出力スケジューラによって整列される．例えば，Linux ではシーク時間の最小化や公平性の担保といった目的に応じて，noop, deadline, acticipatory, cfq の 4 種類の入出力スケジューラ[3]を実現している．

OS は入出力デバイスの状態を監視しており，入出力要求を受け取り可能であれば，待ちキューから入出力要求を取り出し，デバイスドライバに渡す．デバイスドライバは入出力要求を入出力デバイスが解釈可能な形式に変換して発行する．入出力デバイスは受け取った入出力要求を解釈し，記憶媒体を読み書きする（実 I/O 処理と呼ぶ）．この後，入出力デバイスは OS に入出力要求の処理完了を通知する．

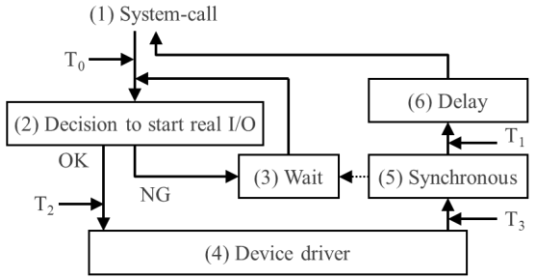


図 2　基本方式
Figure 2　Basic Method.

OS は通知を受け取ると，対応するプロセスを起床する．そして，プロセスは結果を確認し，自身の次の処理に進む．以上がシステムコールの流れである．

ここで，入出力デバイスが入出力要求を受け取り可能となっても，OS がこの状態を検出し，デバイスドライバが入出力要求を変換するために時間を要する．このため，入出力デバイスはただちに入出力要求を受け取れるとは限らない．この結果，OS には入出力要求があるにも関わらず，入出力デバイスは入出力要求の受け取り待ちとなり，入出力スループットが低下してしまう．

このような背景により，Intel 社は同時に 32 つの入出力要求を受け取り可能な入出力デバイスのインタフェース Advanced Host Controller Interface (AHCI)[4]を実現しており，個人向け計算機に普及している．これにより，入出力デバイスは入出力要求を絶え間なく処理できる．したがって，入出力デバイスの処理能力を引き出すには，OS ができる限り多くの入出力要求を発行することが重要となる．

3. 入出力性能を調整する従来方式[2]の問題

3.1 基本方式

入出力性能の調整法の目的は，利用者が指定する要求入出力性能の入出力デバイスが存在し，この入出力デバイスを調整対象プロセスが占有するかのようにみせることである．ここで，要求入出力性能とは，入出力デバイスの処理能力そのものを 100%とした百分率の値である．この目的のために，実 I/O 処理終了後のプロセスの起床前に，要求入出力性能に合わせて遅延処理を実施し，調整対象プロセスの入出力時間を調整する．さらに，入出力デバイスに同時に発行する入出力要求数を制御する．これにより，他プロセスによる入出力時間の長大化を抑制する．基本方式を図 2 に示し，以下に説明する．

（1）プロセスはシステムコールを発行する．

（2）実 I/O 可否判断規則（詳細は 3.1.1 項で述べる）に基づき入出力デバイスへの入出力要求の発行を許可するか否か判断する．許可しない場合，（3）の処理を行う．許可する場合，（4）の処理を行う．

（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．要求入出力性能が高いものを高入出力優先度とし，同入出力優先度は到着順で管理する．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）デバイスドライバを介して入出力要求を発行し，入出力デバイスに実 I/O 処理を実行させる．

（5）待ち処理に同期を送信する．

（6）要求入出力性能に相当する入出力時間（以降，理想の入出力時間と呼ぶ）となるまで，プロセスの起床を遅延する．詳細は 3.1.2 項で述べる．

3.1.1 実 I/O 可否判断規則

調整対象プロセスの実 I/O 処理が理想の入出力時間内に終了するよう，入出力デバイス内の入出力要求数を許容値と呼ぶ値以下に制限する．

最初に，調整対象プロセスが 1 つだけ存在する場合について，許容値の決定式を述べる．理想の入出力時間には，入出力デバイスが当該プロセスの実 I/O 処理を実行する時間（実 I/O 時間と呼ぶ）が含まれる．このため，理想の入出力時間から 1 回分の実 I/O 時間を差し引いた時間だけ，他プロセスの実 I/O 処理を待つことができる．したがって，この時間内に入出力デバイスが処理できる入出力要求数を許容値とする．例えば，要求入出力性能が 20%の場合，理想の入出力時間は実 I/O 時間の 5 倍である．自身の実 I/O 時間を除いた時間内に，4 つの入出力要求を処理できるため，許容値は 4 となる．なお，他プロセスの入出力要求を発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．以上より，要求入出力性能 P の調整対象プロセスが 1 つだけ存在するときの許容値は以下である．

$$max\left(1,\frac{100}{P}-1\right) \tag{1}$$

次に，調整対象プロセスが複数存在する場合を述べる．全ての調整対象プロセスに対して，理想の入出力時間内に実 I/O 処理が完了することを保証するには，許容値の決定において全ての調整対象プロセスを考慮する必要がある．しかし，要求入出力性能が低い調整対象プロセスを考慮しなくても良い場合がある．例えば，要求入出力性能が 20%と 10%の調整対象プロセスが 1 つずつ存在する場合を考える．要求入出力性能 10%の調整対象プロセスを考慮しない場合，許容値は max(1,100/20-1)=4 となる．ここで，要求入出力性能 20%の調整対象プロセスの入出力要求は，要求入出力性能 10%の調整対象プロセスよりも優先される．この結果，要求入出力性能 20%の調整対象プロセスの待ち時間は最長で実 I/O 時間の 4 倍となり，理想の入出力時間（実 I/O 時間の 5 倍）までに実 I/O 処理を完了できる．一方，要求入出力性能 10%の調整対象プロセスの待ち時間は最長で実 I/O 時間の 5 倍となり，こちらも理想の入出力時間（実 I/O 時間の 10 倍）までに実 I/O 処理を完了できる．

上記の留意点をもとに，許容値の決定において，要求入出力性能が高い上位 k 番目までの要求入出力性能の和を利用する．以降，この k を許容係数と呼ぶ．システムコールを発行していない調整対象プロセスの上位 i 番目の要求入出力性能を P_i とするとき，以下の式で許容値を決定する．

$$max\left(1,\frac{100}{\sum_{i=1}^{k}P_i}-1\right) \tag{2}$$

なお，許容係数は求められる調整精度に応じて設定される．例えば，求められる調整精度があまり高くない場合，許容係数を小さくすることが望ましい．一方，多くの調整対象プロセスについて入出力性能の調整精度を高めるには，許容係数は大きい方が良い．

3.1.2 遅延処理方式

実 I/O 処理終了後のプロセスの起床を遅延させることで，調整対象プロセスのシステムコールの終了時刻を調整する．

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）を減算して求める．したがって，以下の式により遅延時間 T_S を決定する．

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

ここで，実 I/O 時間は入出力デバイスごとに異なるため，次の方法で計測する．入出力デバイス内の入出力要求が 1 つの場合，図 2 の T_2 と T_3 の差分が実 I/O 時間である．一方，入出力デバイス内で実 I/O 開始待ちが発生する場合，つまり複数の入出力要求がある場合には，入出力要求を連続して処理するため，ある実 I/O 処理の終了時刻と次の実 I/O 処理の開始時刻は同じである．このため，T_3 の間隔（入出力要求の処理完了通知，つまり実 I/O 処理終了の時刻の間隔）が実 I/O 時間である．したがって，以下の式により実 I/O 時間を決定する．

$$\text{実 I/O 時間}=min\left(T_3-T_2,T_3\text{の間隔}\right) \tag{4}$$

また，遅延ではインターバルタイマ割込により経過時間を監視するため，算出した遅延時間 T_S と実際にプロセスを停止した時間に差が生じ，理想の入出力時間から乖離し得る．そこで，遅延時間 T_S と実際の停止時間の差分を次回の遅延処理へ繰り越す．これにより，個々の入出力時間のばらつきを平準化する．

3.2 従来方式の問題

入出力デバイスの処理能力を引き出し，入出力スループットを向上させるには，許容値を大きくする必要がある．

従来方式では，複数の調整対象プロセスがある場合，入出力時間の保証に必要な値よりも小さい許容値を算出し得る．例えば，要求入出力性能 10%の調整対象プロセスが 2 つ，許容係数 2 の場合を考える．従来方式では，要求入出力要求の合計値 20%を用いるため，許容値は 4 となる．ここで，2 つの調整対象プロセスがたて続けに入出力要求を発行すると，後から発行した調整対象プロセスは，先に発

行された入出力要求を待つ必要がある．この結果，各調整対象プロセスの待ち時間は，実 I/O 時間の 4 倍または 5 倍となる．ところで，どちらの調整対象プロセスも他プロセスの入出力要求を 9 つまで待つことができる．つまり，許容値 8 であっても，入出力時間を保証できる．

許容値が必要よりも小さいと，入出力デバイスが同時に受け取る入出力要求数が過少となる．したがって，従来方式では，デバイスの処理能力を十分に引き出せないため，入出力スループットが低下する．

4. 提案方式

4.1 許容値の算出方法

他の調整対象プロセスによる待ち時間の増加は，次の要因で発生する．

（要因１）自プロセスよりも高い要求入出力性能の調整対象プロセスが入出力要求を発行する．

（要因２）同じ要求入出力性能の調整対象プロセスが自プロセスより先に入出力要求を発行する．

入出力要求の発行を事前に予測することはできないため，（要因１）と（要因２）を踏まえた最長の待ち時間を見込む必要がある．つまり，自プロセスより高い（要因１）または同じ（要因２）要求入出力性能を持つ調整対象プロセスの入出力要求だけ待ち時間が増加すると考える．そこで，この待ち時間と自身の実 I/O 時間を理想の入出力時間から差し引いたものを当該プロセスの許容値A_iとする．したがって，待ち時間の増加を考慮した許容値A_iは以下である．

$$A_i = \frac{100}{P_i} - P_i\text{以上の要求入出力性能のプロセス数} - 1 \quad (5)$$

この許容値A_iは自プロセス以外の全ての調整対象プロセスを考慮しているため，全ての調整対象プロセスについて，許容値A_iを満足すれば入出力時間を保証できる．ただし，入出力処理が停止しないように許容値は 1 以上とする．つまり，以下の式で全体の許容値Aを決定する．

$$A = max\left(\min_i(A_i), 1\right) \quad (6)$$

4.2 期待される効果

提案方式では，それぞれの調整対象プロセスが必要とする許容値から全体の許容値を求めるため，許容値を過少とすることがない．つまり，従来方式に比べて入出力デバイス内の入出力要求数を増やすことができ，入出力スループットの向上を期待できる．

例えば，3.2 節で示した事例では，それぞれの調整対象プロセスの許容値を 100/10-1-1=8 と算出し，これらの最小値 8 を全体の許容値とする．提案方式で算出する許容値は，従来方式の許容値 4 よりも大きくなる．

なお，調整対象プロセスの入出力時間は保証できるため，従来方式と同じく入出力時間を精度良く調整できる．

表 1　計算機の能力

Table 1　Specification of Computer.

Processer	2.5GHz 4core
Main Memory	8GB
I/O Device	AHCI v1.31 5400rpm SATA/600 HDD

表 2　測定パタン

Table 2　Measurement Pattern.

Number of regulation processes	2, 3, 4, 5
I/O size [kB]	0.5, 4, 8, 16, 32

5. 評価

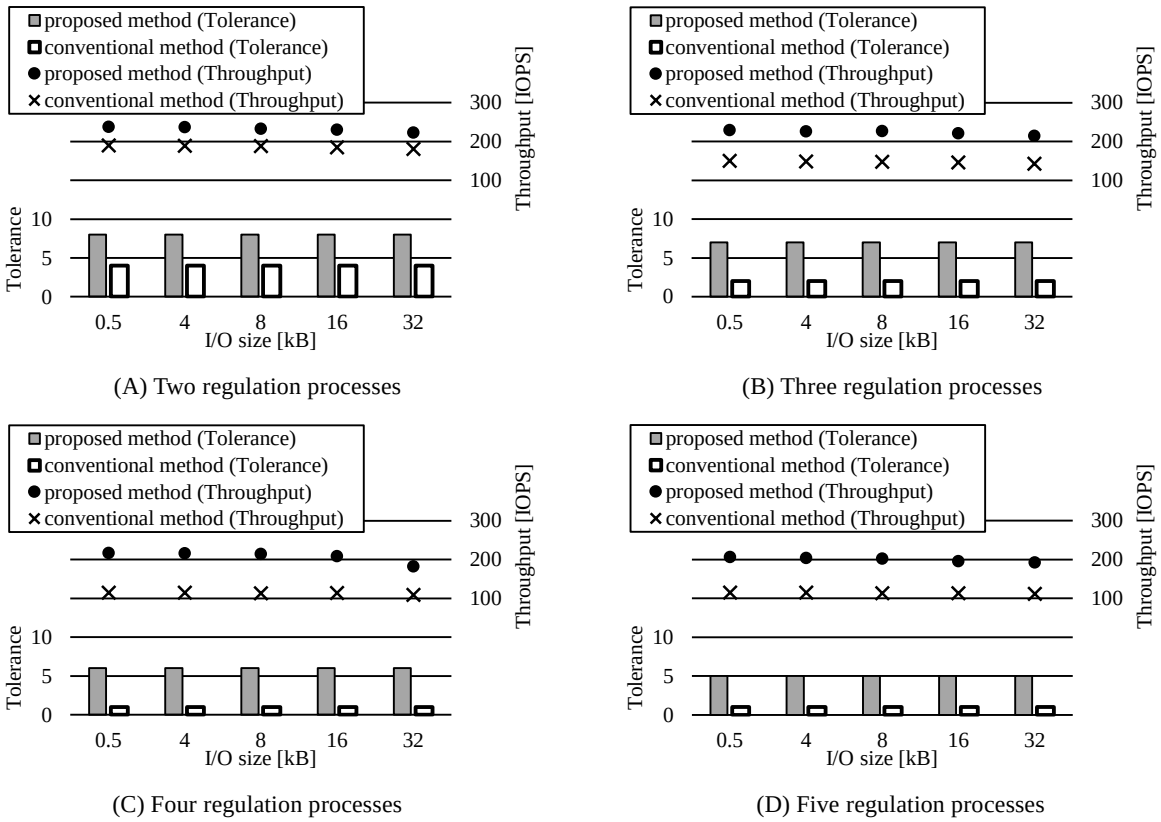
5.1 条件

まず，評価指標を述べる．入出力スループットは，単位時間当たりに OS が入出力デバイスから受け取った入出力要求の完了通知数（I/O per seconds, IOPS）で評価する．入出力時間を精度良く調整できたか否かは，各入出力要求における実際の入出力時間が理想の入出力時間にどの程度近いかで評価する．具体的には，各入出力要求について式(7)で調整精度を求める．調整精度は，1 に近いほど調整の精度が良く，1 より大きいほど入出力時間が過大，1 より小さいほど入出力時間が過小であることを示す．

$$\text{調整精度} = \frac{\text{実際の入出力時間}}{\text{理想の入出力時間}} \quad (7)$$

次に評価環境を述べる．本調整法を FreeBSD Ver11.0 に実装し，表 1 の計算機を用いて評価した．評価プログラムとして，2GB のテキストファイルに対してランダム読み込みを 10000 回繰り返す処理を用いた．このプログラムを 10 つのプロセスで走行させた．ファイルアクセスの競合を防ぐため，プロセスごとにテキストファイルを用意した．10 つのプロセスのうち調整対象プロセスに指定する個数と入出力サイズを表 2 の通りに変化させた．これは調整対象プロセスの数による調整能力の変化と入出力サイズによる影響を調べるためである．

最後に，測定データの加工方法を述べる．ハードディスクドライブ（HDD）は入出力要求を受け取ってない時間が続くと，省電力化のためにディスクの回転速度を低下または回転を停止する．このため，プロセスの起動直後は実 I/O 時間が安定しない場合がある．また，プロセッサのスケジューリングによって，プロセスの終了時刻にばらつきがある．一部のプロセスが終了すると，入出力要求の発行数が少なくなってしまう．そこで，これらの影響を除くために，プロセス起動後の 60 秒間とプロセス終了前の 60 秒間を除いた期間の平均値を用いた．



(A) Two regulation processes　(B) Three regulation processes

(C) Four regulation processes　(D) Five regulation processes

図 3　平均の入出力スループット

Figure 3　Average of I/O Throughput.

5.2 入出力スループット

調整対象プロセス数を 2 から 5 まで変化させたときの許容値と入出力スループットを図 3 の(A)から(D)に示す．提案方式と従来方式が算出した許容値を棒グラフで，入出力スループットを点で，それぞれ表す．図 3 より，次のことがわかる．

（1）提案方式の許容値は従来方式よりも大きい. 例えば，調整対象プロセス数 2, 入出力サイズ 8kB の場合（図 3 (A)），提案方式の許容値は 8 であり, 従来方式の 4 よりも大きい．

（2）提案方式の入出力スループットは従来方式よりも高い．例えば，調整対象プロセス数 2，入出力サイズ 8kB の場合（図 3 (A)），提案方式の入出力スループットは 233 IOPS であり，従来方式の 188 IOPS よりも高い．

（3）従来方式の許容値が小さいほど，提案方式による入出力スループットの向上率が高い．例えば，調整対象プロセス数 2，入出力サイズ 8kB の場合（図 3 (A)），許容値の差は 8-4=4 であり, 入出力スループットの向上率は 233/188 ≒1.24 である．一方で，調整対象プロセス数 5，入出力サイズ 8kB の場合（図 3 (D)），許容値の差は 5-1=4 と同じであるものの，入出力スループット率は 202/114≒1.78 と高い．なお，最も入出力スループットの向上率が高いケースは，調整対象プロセス数 4，入出力サイズ 8kB（図 3 (C)）であり，1.90 となる．

（4）上記の関係は入出力サイズに依存しない．

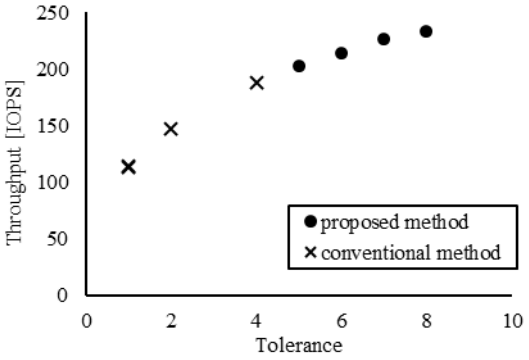
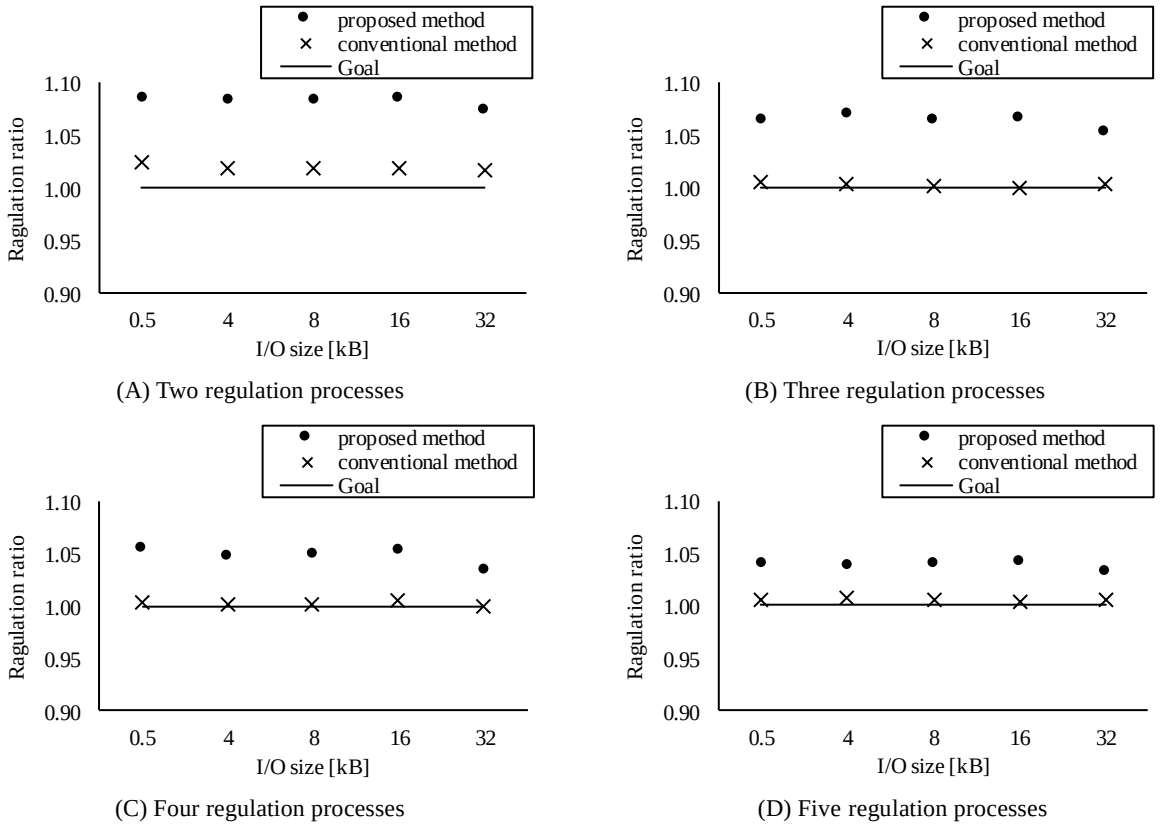


図 4　許容値と入出力スループット（入出力サイズ 8kB）

Figure 4　Tolerance vs. I/O Throughput. (I/O size 8kB)

以上より，提案方式は従来方式よりも許容値を大きくすることができ, 入出力スループットを向上できるとわかる．

続いて，上記（3）の原因を明らかにするために，許容値と入出力スループットの関係を図 4 に示す．図 4 より許容値が小さい場合，許容値の違いが入出力スループットに与える影響が大きいとわかる．例えば，許容値が 7 と 8 の場合，入出力スループットの差は 233-226=7 IOPS である．一方で，許容値が 1 と 2 の場合，入出力スループットの差は 147-113=34 IOPS と大きい．これは，入出力デバイスが同時に受け取る入出力要求が少ないほど，新たな入出力要求の受け取り待ちとなる可能性が高いためと考えられる．



(A) Two regulation processes　(B) Three regulation processes

(C) Four regulation processes　(D) Five regulation processes

図 5　平均の調整精度

Figure 5　Average of Regulation Ratio.

5.3 入出力性能の調整精度

上記 5.2 節の測定における調整精度を図 5 に示す．提案方式と従来方式の調整精度を点で，調整精度が最も良い状態（調整精度 1）を実線で，それぞれ表す．図 5 より，次のことがわかる．

（1）従来方式の調整精度は提案方式よりも良い. 例えば，調整対象プロセス数 2, 入出力サイズ 8kB の場合（図 5 (A)），従来方式は調整精度 1.02 である一方で，提案方式 1.08 とわずかに大きい．

（2）従来方式と提案方式ともに調整対象プロセス数が多いほど，調整精度が良い．例えば，調整対象プロセス数 5, 入出力サイズ 8kB の場合（図 5 (D)），従来方式の調整精度は 1.01，提案方式は 1.04 であり，上記（1）よりも良い．

（3）上記の関係は入出力サイズに依存しない．

わずかな差であるが，上記（1）の原因を明らかにするために，許容値と調整精度の関係を図 6 に示す．図 6 より許容値が大きいほど，調整精度が悪くなることがわかる．これは次の理由による．入出力デバイスが HDD であるため，シーク時間によって実 I/O 時間にばらつきが生じる．調整対象プロセスの実 I/O 時間が短い場合，本調整法の遅延処理によって理想の入出力時間まで待たされる．つまり調整精度は 1 に近い値となる．一方で，実 I/O 時間が長い場合，理想の入出力時間を超過すると調整精度が 1 より大きくなる．したがって，実 I/O 時間の長時間化によって理想の入出力時間を超過する頻度が高いほど調整精度が悪くなる．ここで，許容値が大きいほど，他プロセスの入出力要求による待ち時間が長い．つまり，調整対象プロセスの実 I/O 処理開始時における理想の入出力時間までの残り時間が少ない．この結果，実 I/O 時間のばらつきによって理想の入出力時間を超過しやすくなる．このように，許容値が大きいほど調整精度が悪くなりやすい．

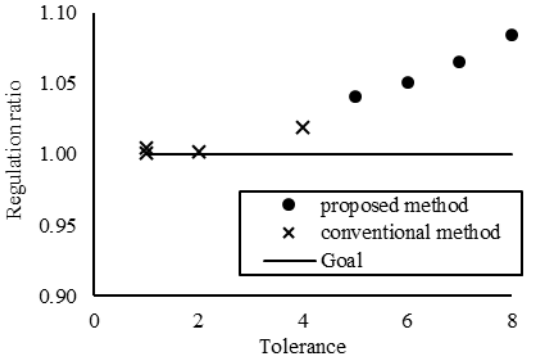


図 6　許容値と調整精度（入出力サイズ 8kB）

Figure 6　Tolerance vs. Regulation Ratio. (I/O size 8kB)

以上より，提案方式は従来方式よりも許容値を大きくするために，入出力デバイスの処理能力に変動がある場合には，調整精度が悪くなりやすいとわかった．

6. 関連研究

入出力スループットの向上を目的として，入出力要求を並び替えて HDD のシーク時間を最小とする研究[5]がある．ところで，近年普及している SSD は入出力要求の処理時間が短いため，OS と入出力デバイス間の入出力要求の授受に要する時間が入出力時間に影響を与えやすい．そこで，入出力要求をまとめて発行することで授受の回数を削減する研究[6][7][8]，入出力デバイスからの通知をポーリング方式で受領することで完了通知による割込みの処理時間を削減する研究[9]がある．また，SSD ではメモリセルへの読み書きにおいて排他が必要となる．読み書き対象の領域が分散するように入出力要求を並び替えて排他の発生を防ぐ研究[10][11][12][13]がある．さらに，プロセス間で入出力要求の発行時期がずれるよう，プロセスへのプロセッサ割当てを制御する研究[14]がある．

一方で，個々のプロセスの入出力性能を保証する方法として，優先度制御と資源予約，デッドライン制御，および比率処理のスケジューラが研究されている．優先度制御には文献[15][16]，資源予約には文献[17]の研究がある．デッドライン制御に関しては Liu らが提唱した EDF[18]を基盤として，リアルタイムシステム向けに拡張した RBED[19]や Deferrable Schedule と組み合わせることで平均応答時間を改善する DS-EDF[20]がある．さらに，デッドラインを超過した入出力要求を優先して処理する研究[21][22]，デッドライン制御と競合回避のための入出力要求の並び替えを組み合わせた研究[23]がある．比率処理に関しては，Tsai らが重み付きラウンドロビン SCAN (WRR-SCAN)[24]，Valente らが Budget Fair Queueing (BFQ)[25]を提案している．低付加時に要求性能の比率で制御し，高負荷時には優先制御に切り替える研究[26]がある．また，Wu らは重要なプロセスが使用できる単位時間当たりの資源量を固定化するスケジューラ[27]を提案している．

このように，入出力スループット向上やプロセスの入出力時間の保証に関する研究がある．一方で，入出力デバイスの処理能力に余裕があるときであっても，プロセスの入出力時間を調整して，実行速度を制御する研究として[2]がある．しかし，入出力デバイスの処理能力を十分に引き出すことができない．

7. おわりに

複数の調整対象プロセスの入出力性能を精度良く調整し，かつ入出力スループットを向上する入出力性能の調整法を提案した．提案方式では，他の調整対象プロセスの入出力要求による待ち時間の増加分だけ，入出力デバイスに発行できる入出力要求数を減少させる．これにより，従来方式よりも入出力デバイス内の入出力要求数が多くなり，入出力スループットを向上できる．

評価では，要求入出力性能 10%の調整対象プロセスが 4 つあるとき，従来方式に比べて入出力スループットが 1.90 倍に向上することを確認した．一方で，調整対象プロセス数が 2 のとき，実 I/O 時間のばらつきによって，従来方式の調整精度が 1.02 に対して提案方式では 1.08 とわずかに大きくなることが分かった．このように，提案手法は調整対象プロセスの入出力時間を精度良く調整し，かつ入出力スループットを向上できる．

残された課題として，実 I/O 時間のばらつきの影響を抑えることができる許容値の計算方法の検討がある．

参考文献

[1] 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）, vol.J83-D-I, no.5, pp.469-477, May 2000.
[2] 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，" 信学論（D）, Vol.J94-D, No.7, pp.1047-1057, July 2011.
[3] Seelam, S., Romero, R., Teller, P., and Buros, B. "Enhancements to linux i/o scheduling," In Linux Symposium. 2005.
[4] https://www.intel.com/content/www/us/en/io/serial-ata/ahci.html
[5] G. Basavaraju, P. K. Kondugari and Shankaraiah, "Accelarting the performance of Disk with re-oredering of an input/output requests at virtual machine monitor level," Proceedings of IEEE International Conference on Computer Communication and Systems (ICCCS), pp.245–247, Aug. 2014.
[6] M. Kesavan, A. Gavrilovska, and K. Schwan. "On Disk I/O Scheduling in Virtual Machines," The 2nd Workshop on I/O virtualization (WIOV), pp6–6, Mar. 2010.
[7] J. Kim, S. Seo, D. Jung, J. Kim and J. Huh, "Parameter-Aware I/O Management for Solid State Disks (SSDs)," IEEE Transactions on Computers, Vol.61, No.5, pp.636–649, May 2012.
[8] Y. Son, H. Y. Yeom and H. Han, "Optimizing I/O Operations in File Systems for Fast Storage Devices," in IEEE Transactions on Computers, Vol.66, No.6, pp.1071–1084, June 2017.
[9] Y. J. Yu, D. I. Shin, W. Shin, N. Y. Song, J. W. Choi, H. S. Kim, H. Eom, and H. Y. Yeom. "Optimizing the Block I/O Subsystem for Fast Storage Devices," ACM Transactions on Computer Systems (TOCS), Vol.32, No.6, pp1–48, June 2014.
[10] S. Park and K. Shen. "FIOS: A Fair, Efficient Flash I/O Scheduler," 10th USENIX Conference on File and Storage Technologies, Feb. 2012.
[11] S. He, Y. Wang, X. Sun, C. Huang and C. Xu, "Heterogeneity-Aware Collective I/O for Parallel I/O Systems with Hybrid HDD/SSD Servers," IEEE Transactions on Computers, Vol.66, No.6, pp.1091–1098, June 2017.
[12] M. H. Jo and W. W. Ro, "Dynamic Load Balancing of Dispatch Scheduling for Solid State Disks," IEEE Transactions on Computers, Vol.66, No.6, pp.1034–1047, June 2017.
[13] Y. Won, J. Jung, G. Choi, J. Oh, S. Son, J. Hwang, and S. Cho. "Barrier-enabled IO stack for flash storage," 16th USENIX Conference on File and Storage Technologies (FAST 18), pp.211-226, Feb. 2018.
[14] X. Zhang, K. Davis and S. Jiang, "Opportunistic Data-driven Execution of Parallel Programs for Efficient I/O Services," 2012 IEEE 26th International Parallel and Distributed Processing Symposium (IPDPS), pp.330-341, May 2012.
[15] E. Betti, S. Bak, R. Pellizzoni, M. Caccamo and L. Sha, "Real-Time I/O Management System with COTS Peripherals," IEEE Transactions on Computers, Vol.62, No.1, pp.45-58, Jan. 2013.
[16] S. Kim, H. Kim, J. Lee, and J. Jeong. "Enlightening the i/o path: A

holistic approach for application performance," 15th USENIX Conference on File and Storage Technologies (FAST 17), pp.345-358, Mar. 2017.
[17] 谷村勇輔，鯉江英隆，工藤知宏，小島功，田中良夫，"ユーザによる明示的な予約に基づき I/O 性能を保証する分散ストレージシステム，" 情報処理学会論文誌コンピューティングシステム（ACS），Vol.5, No.3, pp.42-56, May 2012.
[18] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," Journal of ACM, Vol.20, No.1, pp.46-67, Jan. 1973.
[19] A. Povzner, T. Kaldewey, S. Brandt, R. Golding, T. M. Wong, and C. Maltzahn. "Efficient guaranteed disk request scheduling with Fahrrad," EuroSys'08: Third ACM European Conf. on Computer Systems, pp.13-25, Apr. 2008.
[20] S. Han, D. Chen, M. Xiong, K. Lam, A. K. Mok and K. Ramamritham, "Schedulability Analysis of DeferrableScheduling Algorithms for Maintaining Real-Time Data Freshness," IEEE Transactions on Computers, Vol.63, No.4, pp.979-994, Apr. 2014.
[21] Q. Zhang, D. Feng, F. Wang and Y. Xie, "An Interposed I/O Scheduling Framework for Latency and Throughput Guarantees," Journal of Applied Science and Engineering, Vol.17, No.2, pp.193-202, Feb. 2014.
[22] D. Kang, S. Jung, R. Tsuruta and H. Takahashi, "Range-BW: I/O Scheduler for Predicable Disk I/O Bandwidth," 2010 2nd International Conference on Computer Engineering and Applications (ICCEA), pp.175-180, Mar. 2010.
[23] A. Povzner, D. Sawyer, and S. Brandt, "Horizon: Efficient deadline-driven disk I/O management for distributed storage systems," Proc. 19th ACM Int. Symp. High Perform. Distrib. Comput., Jun 2010.
[24] C. Tsai, T. Huang, E. Chu, C. Wei, Y. Tsai, "An Efficient Real-Time Disk-Scheduling Framework with Adaptive Quality Guarantee," IEEE Transactions on Computers, Vol.57, No.5, pp.634-657, May 2008.
[25] P. Valente and F. Checconi, "High Throughput Disk Scheduling with Fair Bandwidth Distribution," IEEE Transactions on Computers, vol.59, no.9, pp.1172-1186, Sept. 2010.
[26] A. Merchant, M. Uysal, P. Padala, X. Zhu, S. Singhal, and K. Shin. "Maestro: Quality-of-Service in Large Disk Arrays," Proceedings of the 8th ACM International Conference on Autonomic computing (ICAC), pp.245-254, June 2011.
[27] Y. Wu, B. Jia and Z. Qi, "IO QoS: A New Disk I/O Scheduler Module with QoS Guarantee for Cloud Platform," 2012 4th International Symposium on Information Science and Engineering (ISISE), pp.441-444, Dec. 2012.