システムソフトウェアと
オペレーティング・システム 76－3
（1997．8．21）

プログラム実行時間を調整する入出力制御法

坂口 修　谷口 秀夫　牛島 和夫
九州大学 大学院システム情報科学研究科

ハードウェア性能が機種毎に異なるので、ＡＰの処理速度も違ってくる。また、処理速度が速過ぎて、ゆっくり実行させたい時もあるだろう。既存のＯＳでは、主に資源の有効利用を目的としているため、プログラムの実行時間の調整はＡＰ側で行なわなければならない。このようなことから、いくつかのＡＰでは実行時間を調整する処理を行なっている。しかし、これはＡＰの寿命を縮めるといった問題を起こしてしまう。本稿では、プログラムの実行時間の調整を、ＯＳ側で行なう方法について述べる。サービス処理は、プロセッサ処理と、外部装置との入出力処理からなる。ここでは入出力処理に着目し、プログラムの実行時間を調整する入出力制御法について提案する。さらに提案した入出力スケジュール法を実装評価し、その有効性を示す。

I/O Control Mechanism for Adjusting Program Execution Time

Osamu SAKAGUCHI , Hideo TANIGUCHI and Kazuo USHIJIMA
Graduate School of Information Science and Electrical Engineering,
Kyushu University

The difference of performance among hardwares causes the difference of execution time of AP's. We sometimes want to execute a program more slowly because processing speed is too much fast. Existent OS's are supposed only to use resources efficiently. So some AP's adjust the execution time by themselves. But a problem is that it may shorten the life of AP. In this paper, we propose an I/O control mechanism for adjusting program execution time by OS. A service process consists of the computing process and the I/O process with other machines. Focussing to the latter, we propose an I/O control mechanism, and prove its effectiveness

1. はじめに

計算機ハードウェアの性能向上により、ソフトウェアの処理時間は著しく短縮されている。これに伴い複雑なソフトウェア処理が可能になり、様々な利便が得られている。

しかし、ソフトウェア処理はハードウェア性能に依存するため、ハードウェア性能の向上により、いくつかの問題を生じている。例えば、ハードウェア間で性能が異なることから、ハードウェア機種毎の速度調整が必要になってきている。また、プロセッサ性能が高くなり過ぎて、ソフトウェア処理の速度が人間の時間感覚を大幅に超越する恐れがあることから、人間とのインタフェースを実現しているソフトウェア処理において、人間に合わせた処理速度の調整が必要になってきている。今後、ハードウェア性能が高まるにつれて、さらに様々な問題が発生すると思われる。

このような背景から、筆者らは、計算機の性能調整を目的としたスケジュール法の研究を行なっている。これまでに、プロセッサ性能を調整する演算スケジュール法 [1]、およびそのスケジュール法と既存スケジュール法との共存法 [2] を提案した。これによって、応用プログラム (以降 AP と略す) が直接にオペレーティングシステム (以降 OS と略す) へ速度調整の要求をすることが可能になった。しかし、通常の AP はディスクアクセスや他機器との入出力などを行なっているため、プロセッサ性能を調整するだけでは、AP のサービス速度の調整には不十分である。

そこで、本論文では、入出力の性能を調整する機能について述べる。具体的には、入出力回数を調整することにより、入出力の性能を調整する入出力スケジュール法を提案する。さらに、提案した入出力

スケジュール法を既存の OS に実装し、ディスクアクセスに関して評価する。

2. 入出力の性能調整における制御方式

入出力の性能を調整する方式には、プロセスの入出力の性能をどのような観点に従って調整するかにより、大きく次の2つが考えられる。

(1) 入出力時間の調整
(2) 入出力回数の調整

各々について以下に説明する。

2.1 入出力時間の調整

入出力時間の調整とは、各々の「入出力に要する時間」を調整することである。具体的には、入出力の終了時期を調整する、いわゆる応答時間の調整である。

例を図1に示し、説明する。プロセスが100%の性能で入出力を要求する場合、普通に入出力が行なえる。しかし、50%の性能で要求する場合、実際の入出力が終了してもすぐには復帰しない。50%の性能に見合った時間だけ待たせることによって、プロセスに2倍の入出力時間がかかったように見せる。これにより、性能調整が実現できる。

この方法は、入出力要求の1つ1つを調整するため、非周期的な入出力要求にも対応できる。しかし、複数の入出力要求が衝突した場合、入出力時間内の終了を満足できない可能性がある。これを避けるには、複数のプロセス間の相関を考慮したスケジュールを行なわなければならず、そのため入出力要求が比較的少ない状況に向いている。

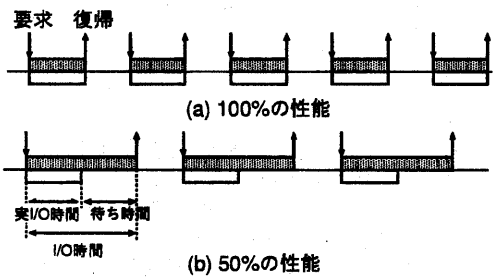

図1 入出力時間の調整

2.2 入出力回数の調整

入出力回数の調整とは、「単位時間における入出力の回数」を調整することである。具体的には、入出力の開始時期を調整する、いわゆるスループットの調整である。

例を図2に示し説明する。単位時間(T)あたり4回の入出力を要求をするようなプロセスがある。このプロセスが4(回/T)の性能で入出力を要求する場合、要求通りに入出力が行なわれる。しかし、2(回/T)の性能で要求する場合、同じ単位時間内に3回目の要求が来た時点でその入出力要求は、次の単位時間の開始まで待たされる。これによって、プロセス側には、2(回/T)の性能で入出力が行なわれているように見せることができる。

この方法は、各入出力要求毎ではなく、複数の入出力要求の実行の割合を調整できるので、周期的な入出力要求に向いている。また、スケジュールにおいて、入出力時間の調整ほど他プロセスとの相関を考慮する必要はない。そのため、入出力要求が頻発する状況にも対応できる。

以降では、入出力回数を調整する観点からの入出力の性能調整について述べる。

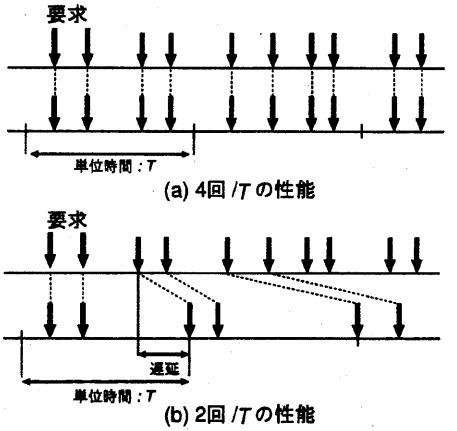

図2 入出力回数の調整

3. 入出力回数の調整法

3.1 基本方式

入出力回数の調整による入出力スケジュール法は、方式を以下の観点から分類できる。

(1) 単位時間の扱い

単位時間をシステムで一意とし、すべてのプロセスを同じ単位時間の周期で扱う方式（以降、固定周期方式と呼ぶ）がある。一方、単位時間はシステムで一意ながらも、各プロセス毎に単位時間の周期を設定する方式（以降、可変周期方式と呼ぶ）がある。さらに、単位時間をプロセス毎に設定する方式も考えられるが、この方法は、単位時間の違いを入出力回数で調整することにより前記2方式と同様になる。

(2) 入出力の割り当て

入出力の割り当てについては、実行前に事前に割り当てる方式と、入出力要求時に割り当てる方式が考えられる。しかし、現在の計算機によるソフトウェアの処理は、プロセッサによる命令の実行が基盤になっており、必要に応じて入出力を行なう。入出力の実行を基盤として、必要に応じてプロセッサによる命令の実行を行なっているものではない。このため、プロセッサ性能を調整する場合 [1] と同様に、実行前に事前に入出力を割り当てることは簡単ではない。また、実行前に事前に入出力の契機を予測することは多くの場合において不可能である。

以上の観点の組み合わせから4方式が考えられるが、先に述べたように、入出力の割り当てを実行前に事前に行なうことは、難しい。このため、入出力の割り当ては入出力要求時に行ない、単位時間の扱いとして固定周期方式と可変周期方式を考える。

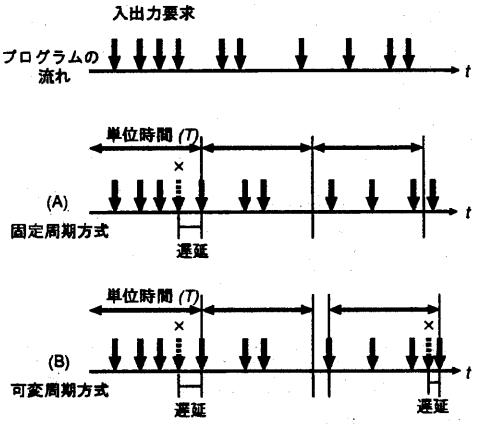

図3　固定周期方式と可変周期方式（要求：3回/T）

各方式を図3に示す。図3(A)は固定周期方式を示したものであり、固定周期方式は単位時間の周期が固定であるため、入出力性能を調整するために実入出力の遅延が1回発生している。図3(B)は可変周期方式を示したものであり、可変周期方式は単位時間の周期が可変である。このため、入出力要求がしばらくないと、単位時間と単位時間の間に隙間ができる。入出力性能を調整するために実入出力の遅延が2回発生している。

3.2 入出力優先度

入出力要求が同時に多数行なわれた時、それらの要求の実行順序を決定するため、入出力優先度を導入する。各変数を、

T : 単位時間 (単位:秒)
n_i : プロセス i の入出力要求性能 (回/T)
$m_i(t)$: 時刻 t におけるプロセス i の実行済入出力回数 (回) $0 \leq m_i \leq n_i$

とすると、時刻 t における残入出力回数は、

$n_i - m_i(t)$

となる。ここで、残時間を、$\alpha_i(t) \cdot T$とする。$\alpha_i(t)$はプロセス i の残時間係数、傾き-1の直線による鋸波である ($0 \leq \alpha_i(t) \leq 1$)。これらにより、時刻 t におけるプロセス i の入出力優先度を、

$$P_i(t) = \frac{(n_i - m_i(t))/n_i}{\alpha_i(t) \cdot T}$$

と定義する。この $P_i(t)$ は、可変周期方式の場合の入出力優先度である。固定周期方式の場合、すべてのプロセスで単位時間が同期しているため、

$$\alpha_i(t) = \alpha(t) = -\frac{t}{T+1} \quad (0 \leq t \leq T)$$

となり、この場合の入出力優先度は、

$$P_i a(t) = \frac{(n_i - m_i(t))/n_i}{T - t}$$

となる。

4. 設計と実装

固定周期方式によるこの入出力性能の調整機能をBSD/OS2.1のread(),write()システムコールに組み込みんだ。実装においては既存の機能やインタフェースの変更を極力抑えるようにした。

入出力の性能調整を要求するプロセスは、read(),write() システムコールを発行する前に、入出力の性能調整を要求していることをread(),write() システムコール側に知らせ、それに対応する処理をさせる必要がある。そこで、入出力性能管理表を作り、プロセス識別子 (pid), 入出力性能 (performance), 入出力済回数 (count) を管理する。この管理表に登録/抹消するようなシステムコール setiop() と resetiop() を作成した。これは、引数 pid に他プロセスのプロセス識別子を渡すと、他プロセスの入出力性能を登録/抹消できる。引数 pid に 0 を渡すと自プロセスの入出力性能を登録/抹消できる。各システムコールの仕様を表1、表2に示す。

表1　setiop() システムコールの仕様

形式	setiop(pid, performance)
内容	入出力性能管理表に pid、performance を登録し、count = 0 とする。(pid = 0 ならば、自プロセスの pid。)

図4と図5にread() システムコールの場合の処理の流れを示し、以下に説明する。

表 2　resetiop() システムコールの仕様

形式	resetiop(pid)
内容	入出力性能管理表から自プロセスが使用していたエントリを削除する。(pid = 0 ならば、自プロセスの pid。)

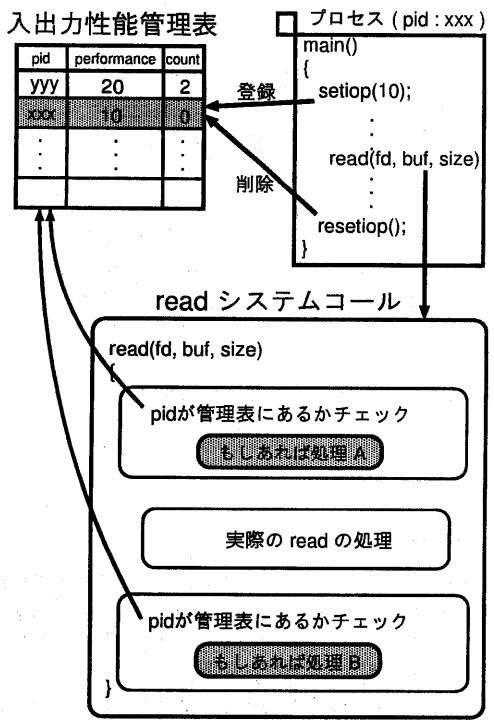

図 4　プログラムの流れのイメージ

read() システムコールが呼ばれる度に、呼んだプロセスの pid が、入出力性能管理表に登録されているかどうかをチェックする。登録されていれば、read() システムコールの最初と最後にある処理 A, 処理 B を実行する。これによって、登録されていないプロセスには従来と何ら変わらない処理を行なわせることができる。処理 A では、入出力要求から、実入出力の実行開始までの処理を行なう。他に実入出力を行なっているプロセスが無ければ、そのまま実入出力を行なう。他に実入出力を行なっているプロセスがある場合、実入出力待ちの全プロセスの入出力優先度を算出し、その入出力優先度に基づいてキュー作成を行なう。その後 sleep() で休眠し、キューから取り出されて実入出力が実行されるまで待つ。処理 B では、実入出力の終了から read() システムコールの終了までの処理を行なう。他に実入出力を待っているプロセスがあれば、キューから最大の入出力優先度を持つ入出力要求に wakeup() をかけて覚醒させる。

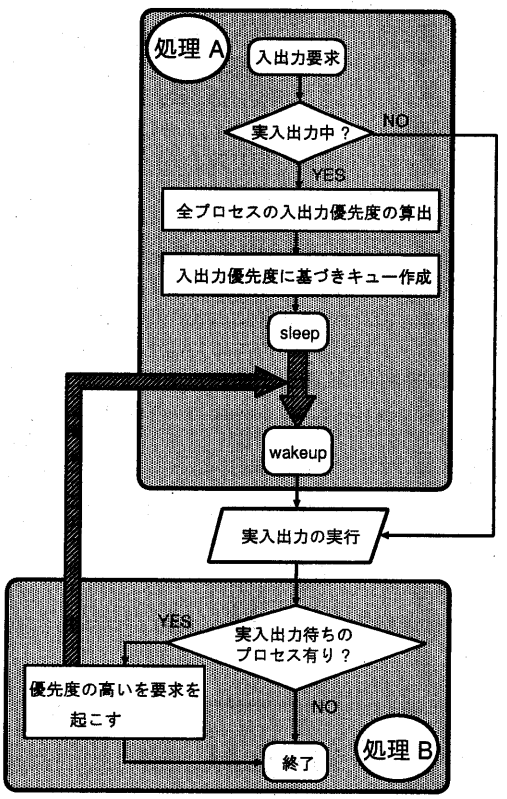

図 5　入出力優先度によるスケジュールと入出力処理の流れ

上記以外に必要な処理として、大きく以下の 2 つを実装した。様子を図 6 に示し、以下に説明する。

(1) キュー管理の処理

入出力要求は、入出力要求箱に入れてキュー管理し先頭から実行させるようにする。キューを 2 つ用意しそれぞれを実行キュー、待ちキューと名付ける。実行キューとは入出力スケジュール対象のキューで、先頭から取り出されて実行される。取り出された入出力要求は、実入出力が終了すると、対応する入出力性能管理表の count の値に 1 を加える。待ちキューとは入出力要求の対応する入出力性能管理表の count の値が performance に達した時につながれるキューで、次の単位時間の開始まで実行されることはない。この 2 つのキューを使って、入出力回数を調整する。

(2) 単位時間毎の処理

単位時間毎に、管理している全プロセスの入出力性能管理表の count の値を 0 に戻す。そして、待ちキューにつながっているすべての入出力要求を実行キューの最後につなぐ。

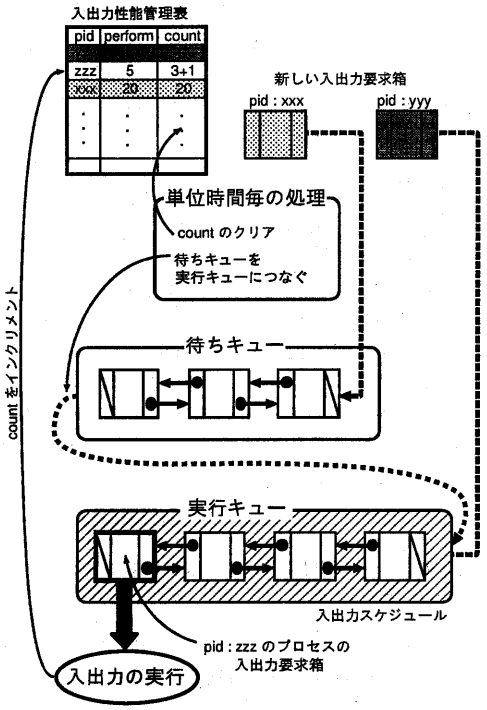

図 6　キュー管理と、単位時間毎の処理の流れ

5. 評価と考察

write() システムコールは、プログラムの画面への出力を使って視覚的に動作を確かめることができた。例えば、既存のゲームプログラムも処理速度を自由に調整することができた。しかし、要求性能を低くするほど、規定の回数だけ実入出力が単位時間内の最初の方に行なわれ、次の単位時間まで静止し、急に動き出すといった問題が起こった。

以下に、実測に基づいた評価を述べる。

5.1 測定環境

測定は、計算機 Pentium 90MHz (メモリ 32Mbyte) の BSD/OS2.1 上で、IDE ディスク (800Mbyte) を使って行なった。測定条件を以下に示す。

- 余計なディスクアクセスを避けるためにシングルユーザモードで測定する。
- テストプログラムの中の read 操作は、ファイル入出力のためのバッファのキャッシュヒットの影響をなくすためにディスクを raw デバイスとしてアクセスし、ディスクシーク時間の影響をなくすためにランダムなセクタから読み込むようにする。
- 単位時間は 1 秒とする。

なお、テストプログラムの内容は、read() システムコールを繰り返すだけのもので、ランダムなバイト数だけ lseek() システムコールでアクセス位置をずらし、データの読み込みを単純に繰り返すものである。

5.2 性能調整の限界

性能の調整がうまくできる範囲を明らかにする。テストプログラムにおいて、1 回の read() で読み込む入力データ長を 512byte とし、要求性能と実性能の関係を実測した。結果を図 7 に示す。

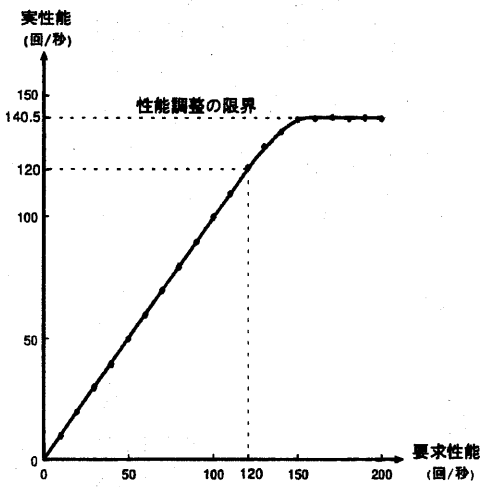

図 7　要求性能と実性能の関係

10〜120(回/秒) の要求性能であれば入出力の性能調整が良好であることがわかる。要求性能が 130(回/秒) 以上になると、要求性能と実性能との間に差が生じ、実性能は約 140.5(回/秒) が最大となる。この値はこのハードウェア環境が提供する最大の性能（以降、生性能と名付ける）である。したがって、生性能の８５%以下の要求性能については、うまく調整できたといえる。

5.3 入出力優先度の効果

入出力優先度の有効性を確認するため、以下の測定を行なった。入力データ長を 512byte とし、同時に 20 個のプロセスを走行させた。20 個の各プロセスの要求性能を 10,20,30,$\cdots$,200 とし、その時の各プロセスの実性能を実測した。入出力優先度に基づかないで入出力スケジュールを行なう場合と、入出力優先度に基づいた場合の 2 通りについて、要求性能と実性能の関係を図 8 と図 9 に示す。どちらの場合も、生性能を越えた要求性能であるため、全プロセスについて要求性能を満足していない。図 8 の入出力優先度に基づかない入出力スケジュールの場合、

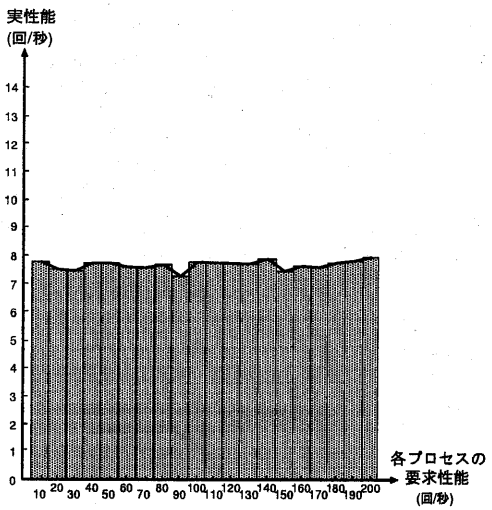

図８　入出力優先度に基づかない入出力スケジュール

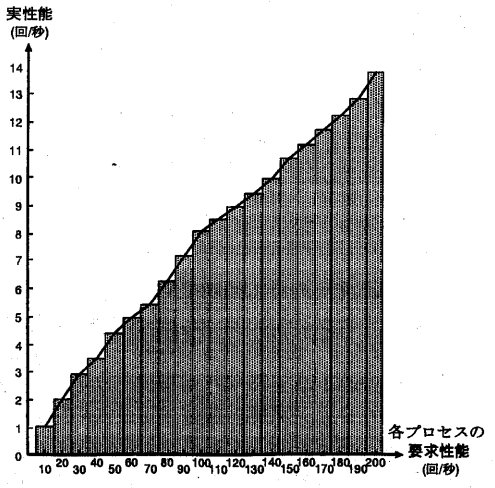

図９　入出力優先度に基づいた入出力スケジュール

各プロセスの要求性能に関係なく、プロセスの実性能はほぼ一定である。一方、図９の入出力優先度に基づいた入出力スケジュールの場合、要求性能が大きいほど実性能も比例して大きくなり、各プロセスの要求性能に見合った割合で入出力が行なわれていることがわかる。

したがって、入出力優先度に基づいたスケジュールは、有効といえる。

6. おわりに

プロセスが単位時間に行なう入出力回数に着目し、入出力の性能を調整する入出力スケジュール法について述べた。入出力の性能を調整する制御方式として、入出力時間を調整する方式と入出力回数を調整する方式について説明した。次に、入出力回数を調整する際のスケジュール方式を示し、入出力優先度を導入した。さらに、提案した入出力スケジュール法を実装し、評価した。実測により、最大性能の８５％までは性能を調整できることを示した。さらに、入出力優先度に基づいた入出力スケジュールの有効性を示した。

今後は、入出力時間の調整の観点から研究し、入出力回数の調整と比較する。さらに、この入出力スケジュールを、我々の研究室で開発している ***Tender*** オペレーティングシステムへ実装する。また、プロセッサ性能を調整する演算スケジュール法と入出力性能を調整する入出力スケジュール法との融合も行なう予定である。

参考文献

1) 谷口秀夫、可変なプロセッサ性能を提供するスケジュール法、コンピュータシステム・シンポジウム、pp.63-70(1994)

2) 村上大介, 谷口秀夫, 牛島和夫、異なるスケジュール法の共存制御法、情処研報 (Vol.95, No.79)、pp.105-112(1995)