Vol.2010-ARC-189 No.10
Vol.2010-OS-114 No.10
2010/4/22

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評価

長尾 尚[†1] 山内利宏[†1] 谷口秀夫[†1]

計算機の性能向上により，1 台の計算機上で複数のサービスを提供できるようになった．しかし，個々のサービス時間は，ハードウェア性能や他のサービスの影響を受ける．一方，ソフトウェアの実行速度を自由に調整できれば，サービスの利便性は向上し，資源競合時におけるソフトウェアの実行速度の低下を抑制することも可能になる．このため，ハードウェア性能の範囲で，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プログラムを実行する制御法の確立が必要である．著者らは入出力性能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．提案制御法は，入出力デバイスへの入出力要求の数を制限することにより，入出力性能の調整精度を向上させている．ここでは，提案制御法の評価として，他プロセスによる調整精度への影響および資源競合時の影響抑制の効果を示す．

Evaluation of Mechanism for Regulating the Service Time Based on Controlling the Number of I/O Request

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

By performance improvement of computers, a computer can provide multiple services. However, individual service time is affected by hardware performance and other services. On the other hand, if we can regulate software execution speed freely, the convenience of the service improves and it is possible to keep software execution speed with competing resources. So, it is necessary to establish a mechanism for regulating program execution speed that fit user or the service contents. We proposed mechanism for regulating the number of I/O request as mechanism for regulating I/O performance. Proposed mechanism improves precision of I/O performance by limiting the number of I/O request for a I/O device. In this paper, we show the influence on precision by the other process and the effect of influence restraint with competing resources.

1. はじめに

計算機性能の著しい向上により，1 台の計算機上で複数のサービスを提供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．

これまでに，著者らは，計算機ハードウェアの性能の範囲でプログラムの実行速度を自由に調整する制御法として，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，プロセッサ性能を調整するスケジュール法は，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，入出力性能を調整する制御法[2),3)] は，調整精度が入出力要求の頻度の影響を受ける問題がある．

そこで，調整精度が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として入出力要求数を調整する制御法[4)] を提案し，他プロセスが存在しない場合における調整精度が高いことを示した．本稿では，提案制御法の他プロセスによる影響の評価結果を示し，資源競合時におけるプログラムの実行速度低下の抑制効果を明らかにする．

I/O スケジューリングについては，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ことで磁気ディスクの入出力性能を向上させる研究[5),6)] 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．また，プロセスの優先度に基づき入出力要求の順番を並べることで優先度の高いプロセスの入出力処理を優先的に実行する研究[7)]，予約した資源量の多いプロセスの入出力処理を優先的に実行する資源予約型 I/O スケジューリング法の研究[8),9)]，入出力優先度を付与し，DBMS 内の要求の順番を入れ替える研究[10),11)]，および実時間性を保証する I/O スケジューリング法の研究[12)] がある．これらの研究は，より重要度の高い入出力要求を優先的に実行することにより，多数のプロセスが入出力処理を行う場合において，重要なプロセスの入出力性能の低下を抑制する．一方，提案制御法は，他プロセスの影響を受けず，ユーザが要求する入出力性能に合わせて入出力時間を調整する．

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

© 2010 Information Processing Society of Japan

Vol.2010-ARC-189 No.10
Vol.2010-OS-114 No.10
2010/4/22

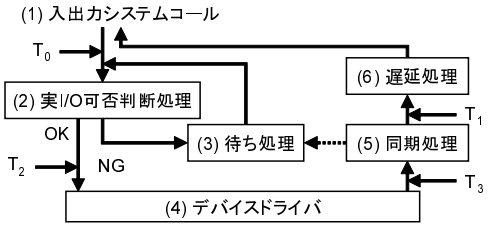


図 1 基本方式

2. 入出力要求数を調整する制御法

2.1 基 本 方 式

入出力要求数を調整する制御法（以降，提案制御法と呼ぶ）は，要求入出力性能から算出される理想の入出力時間（以降，理想の入出力時間と呼ぶ）内に被調整プロセスの実 I/O 処理が終了できるよう，デバイスドライバへの実 I/O 要求数（実 I/O 処理を行うプロセス数）を制限する．これにより，入出力性能の調整精度を保証する．また，被調整プロセスの実 I/O 処理終了後に遅延処理を実施する．これにより，被調整プロセスの入出力性能を制限する．提案制御法の基本方式を図 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）実 I/O 要求数の限界値（詳細は 2.2 節で述べる）に基づき，実 I/O 処理を許可するか否か判断する．許可しない場合，(3-1) の処理を行う．許可する場合，(3-2) の処理を行う．

（3-1）プロセスを待ちキューにつないで待ち状態にする．また，同期処理からの同期があれば，待ちキューの先頭のプロセスを起床する．起床したプロセスは，(2) の処理を行う．

（3-2）実 I/O 処理を行う．

（4）待ち処理に同期を送信する．

（5）理想の入出力時間となるよう，遅延処理を実施する．

個々のデバイスドライバに調整機構を組み込むと，工数が増大して好ましくない．したがって，個々のデバイスドライバを変更せず，OS カーネル内に提案制御法を実現する．

2.2 実 I/O 可否判断処理

理想の入出力時間内に被調整プロセスの実 I/O 処理を終了できるよう，デバイスドライバへの実 I/O 要求数を制限する．具体的には，要求入出力性能から実 I/O 要求数の限界値を算出し，算出した限界値に基づき，デバイスドライバへの実 I/O 要求数を制限する．被調整プロセスが 1 つだけ存在する場合における限界値の算出式について考える．理想の入出力時間は，占有状態における入出力時間を該当被調整プロセスの要求入出力性能（100 %を 1 として換算した値）で割った値となる．1 つの入出力要求の実行に要する時間は，占有状態における入出力時間であるため，デバイスドライバ内の入出力要求の数が該当被調整プロセスの要求入出力性能の逆数以下であれば，被調整プロセスの入出力性能の調整精度を保証できる．ただし，この入出力要求の数には，被調整プロセスの入出力要求も含まれる．したがって，該当被調整プロセスの要求入出力性能の逆数から 1 減算した値を限界値とする．なお，共存プロセスが入出力処理を実行できなくなることを防ぐため，最小の限界値を 1 とする．以上より，被調整プロセスが 1 つだけ存在する場合における限界値の決定式は以下のようになる．

$$MAX(1, \frac{100}{\text{被調整プロセスの要求入出力性能（\%）}} - 1) \quad (1)$$

次に，複数の被調整プロセスが存在する場合について考える．限界値の決定に利用する要求入出力性能について，以下に留意する．

（1）実 I/O 処理中の被調整プロセス

UNIX 系 OS に代表される多くの既存 OS は，入出力処理を同期型としているため，入出力処理が同期型であると仮定する．同期型の入出力処理の場合，プロセスは入出力処理の終了を待つため，入出力処理が終了するまで入出力要求を発行しない．したがって，限界値の決定式において，入出力中の被調整プロセスの要求入出力性能を考慮しない．

（2）保証する被調整プロセス数

全ての被調整プロセスの入出力性能の調整精度を保証するためには，限界値の決定式において，全ての被調整プロセスを考慮する必要がある．しかし，低い要求入出力性能をもつ被調整プロセスを考慮しなくても良い場合がある．例えば，2 つの被調整プロセスが存在し，それぞれの要求入出力性能が 20 %と 10 %である場合を考える．要求入出力性能 10 %の被調整プロセスを考慮しない場合，限界値は 4 となる．このとき，要求入出力性能 10 %の被調整プロセスのデバイスドライバ内における待ち時間は最長で実 I/O 時間の 5 倍となり，どちらの要求入出力性能の調整精度も保証される．

したがって，限界値の決定式において，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 を保証係数と名付ける．つまり，入出力中でない被調整プロセスの上位 i 番目の要求入出力性能を P_i とすると，限界値の決定式は以下のようになる．

© 2010 Information Processing Society of Japan

Vol.2010-ARC-189 No.10
Vol.2010-OS-114 No.10
2010/4/22

$$MAX(1, \frac{100}{\sum_{i=1}^{k} P_i} - 1) \quad (2)$$

ここで，スループット低下を抑制するためには，保証係数を小さくすることが望ましい．一方，多くの被調整プロセスの入出力性能の調整精度を保証できるためには，保証係数は大きい方が良い．

2.3 待ち処理と同期処理

待ち処理は，プロセスを待ちキューにつないで待ち状態にする．このとき，入出力優先度に基づき，待ちキュー内の要素を並べる．入出力優先度の決定規則を以下に示す．

（1）要求入出力性能が高いものほど，高い入出力優先度

（2）同入出力優先度は LRU 方式

また，同期処理からの同期があれば，待ちキュー内の入出力優先度の最も高いプロセスを起床する．なお，起床後のプロセスの優先度を高く設定することにより，プロセスが実行可能状態となってから CPU が割り当てられるまでの待ち時間を短縮する．同期処理は，待ち処理へ同期を送信する．

2.4 遅 延 処 理

要求入出力性能に基づき，プロセスの休眠と覚醒（以降，wait 方式と名付ける）により，遅延処理を実施する．遅延処理は，理想の入出力時間から実 I/O 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．

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

ただし，デバイスドライバに変更を加えないため，実 I/O 時間の測定をデバイスドライバ外（OS カーネル内）で行う必要がある．また，wait 方式を利用した遅延処理では，実効単位と実効精度が問題となる．実効単位は，実際にプロセスを停止できる単位である．つまり，実効単位が 10 ミリ秒の場合，プロセスの停止は 10 ミリ秒の倍数で行われる．このため，例えば，要求遅延時間の端数（10 ミリ秒よりも小さい部分）を減算した値（以降，依頼停止時間と名付ける）でプロセスの停止を依頼することになる．この結果，調整精度は低下する．一方，実効精度は，プロセスを停止した際の精度である．プロセスを停止する際，依頼停止時間と実際の停止時間（以降，実停止時間と名付ける）は等しいことが望まれる．しかし，wait 方式のシステムコールは，タイマ割込を利用しているため，依頼停止時間と実停止時間に誤差が生じ，調整精度は低下する．したがって，実効単位と実効精度による影響を抑制する必要がある．

そこで，OS カーネル内でデバイスドライバへの実 I/O 要求発行の時刻と結果返却の時刻を取得し，実 I/O 時間を測定する．また，遅延可否閾値と遅延時間くりこし値を導入し，実効単位と実効精度の問題に対処する．それぞれの対処を以下に説明する．

（1）実 I/O 時間の測定

デバイスドライバ内に実 I/O 開始の待ち時間がない場合，図 1 中の T_2 と T_3 の差分が実 I/O 時間となる．一方，デバイスドライバ内に実 I/O 開始の待ち時間が存在する場合，つまり，複数の実 I/O 要求がある場合，実 I/O 処理を連続して行うため，T_3 の間隔が実 I/O 時間となる．したがって，T_2 と T_3 を比較し，以下の式により実 I/O 時間を決定する．

$$\text{実 I/O 時間} = MIN(T_3 - T_2, T_3\text{の間隔}) \quad (4)$$

（2）遅延可否閾値の導入

実効単位の問題に対処するため，遅延処理をまとめて実施する．具体的には，要求遅延時間を（T_S）とした場合に，最小遅延時間（W），遅延可否閾値（L），および合計要求遅延時間（S）を設けて制御を行う．ここで，最小遅延時間（W）は利用するシステムコールの実効単位であり，遅延可否閾値（L）は最小遅延時間（W）よりも大きいとする．算出した要求遅延時間（T_S）を合計要求遅延時間（S）へ加算し，合計要求遅延時間（S）と遅延可否閾値（L）を比較する．合計要求遅延時間（S）が遅延可否閾値（L）未満の場合，待ち処理は行わない．合計要求遅延時間（S）が遅延時間閾値（L）以上の場合，合計要求遅延時間（S）から最小遅延時間（W）未満の値を減算したものを依頼停止時間として待ち処理を行い，合計要求遅延時間（S）を初期化する．これにより，調整精度の低下を抑制する．

（3）遅延時間くりこし値の導入

実効精度の問題に対処するため，依頼停止時間と実停止時間の差分を次回遅延処理へ反映する．具体的には，遅延時間くりこし値（T）を設けて制御を行う．依頼停止時間と実停止時間の差分を遅延時間くりこし値（T）として保存し，次回遅延処理時に算出する合計要求遅延時間（S）に加算する．これにより，調整精度の低下を抑制する．

上記の対処を採用した遅延処理の流れを図 2 に示す．

3. 評　　価

3.1 基 本 評 価

3.1.1 評 価 条 件

本制御法を FreeBSD 6.3-RELEASE（以降，FreeBSD と呼ぶ）の OS カーネル内に実装し，Celeron（1.8GHz）プロセッサを搭載した計算機で走行させた．遅延処理には tsleep()

© 2010 Information Processing Society of Japan

Vol.2010-ARC-189 No.10
Vol.2010-OS-114 No.10
2010/4/22

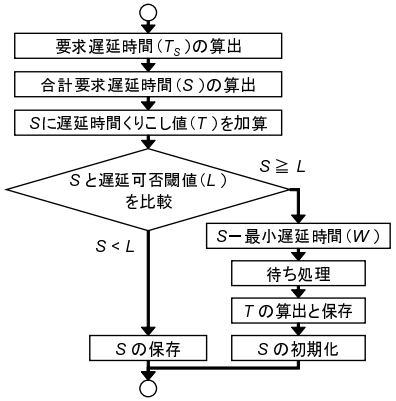


図 2 遅延処理の流れ

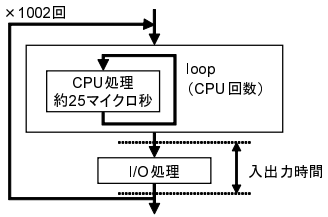


図 3 評価プログラムの内容

を用いたため，最小遅延時間 (W) は 10 ミリ秒である．そこで，遅延可否閾値 (L) も 10 ミリ秒とした．

多くのプログラムは，CPU 処理と I/O 処理を繰り返しながら処理を進める．そこで，図 3 に示すプログラムで評価した．評価プログラムは，CPU 処理として特定のメモリ領域の値のインクリメント処理，および I/O 処理として read() システムコール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，read() システムコール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．各評価では，インクリメントの回数を変更することにより，CPU 処理の時間を変更し，CPU 処理時間と I/O 処理時間の比率（以降，処理比率と名付ける）を変化させた．

本制御法の調整精度を評価するため，調整比を用いる．調整比は，

$$\text{調整比} = \frac{\text{実測値}}{\text{理論値}} \tag{5}$$

である．ここで，理論値は当該の要求入出力性能で理想的に実行速度の調整を行ったと仮定して算出した入出力時間である．

3.1.2 他プロセスの影響

共存プロセスを 1 つだけ同時走行させた場合における要求入出力性能と調整比（平均値）の関係を図 4 に示す．図 4 より以下のことがわかる．

（1）被調整プロセスの処理比率に関係なく，以下のことがいえる．

（A）被調整プロセスの要求入出力性能が低い場合，共存プロセスの影響は小さく，調整精度は高い．例えば，要求入出力性能が 50 %以下の時の調整比は 1.0〜1.2 である．

（B）被調整プロセスの要求入出力性能が高い場合，共存プロセスの入出力処理の比率が高くなるにつれ，共存プロセスの影響は大きくなる．例えば，被調整プロセスの処理比率が 0：1，要求入出力性能が 90 %の場合，共存プロセスの処理比率が 2：1 の時の調整比は約 1.2 であるが，0：1 の時は約 2.1 になってしまう．これは，共存プロセスの入出力処理の比率が高いほど，被調整プロセスと共存プロセスが同時期に入出力要求を発行する確率が高くなり，被調整プロセスの実 I/O 開始の待ち時間が発生するためである．

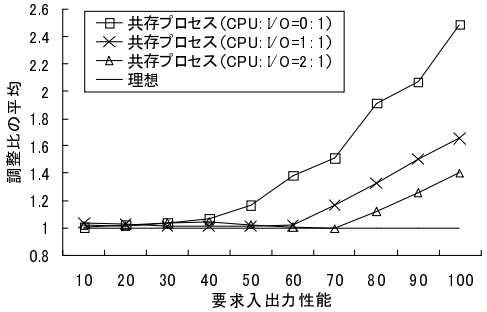
（2）被調整プロセスの入出力処理の比率が高くなると，共存プロセスの影響は大きくなる．例えば，被調整プロセスの要求入出力性能が 90 %で共存プロセスの処理比率が 0：1 の場合，被調整プロセスの処理比率が 2：1 の時の調整比は約 1.6 であるが，0：1 の時は約 2.1 になってしまう．これは，(1)(B) と同様に，被調整プロセスの入出力処理の比率が高いほど，被調整プロセスと共存プロセスが同時期に入出力要求を発行する確率が高くなり，被調整プロセスの実 I/O 開始の待ち時間が発生するためである．

次に，共存プロセス数による影響を明らかにするため，共存プロセス（処理比率は 1：1）の同時走行数を変化させた場合について，共存プロセス数と調整比（平均値）の関係を図 5 に示す．なお，被調整プロセスの処理比率は 1：1 である．図 5 より，要求入出力性能が低いほど，共存プロセス数による影響は小さいことがわかる．例えば，共存プロセス数が 7 の場合，要求入出力性能が 70 %の時の調整比は約 2.2 であるが，30 %の時は約 1.2 である．これもまた，(1)(B) や (2) と同じ理由による．

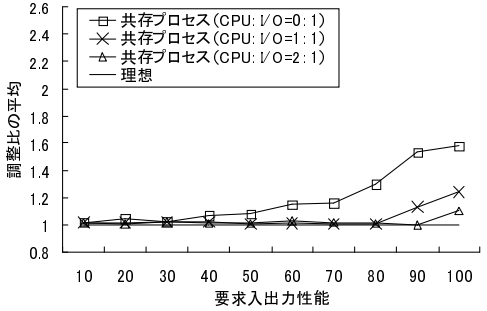
以上のことから，被調整プロセスの要求入出力性能が低い場合は入出力性能を保証できるといえる．

© 2010 Information Processing Society of Japan

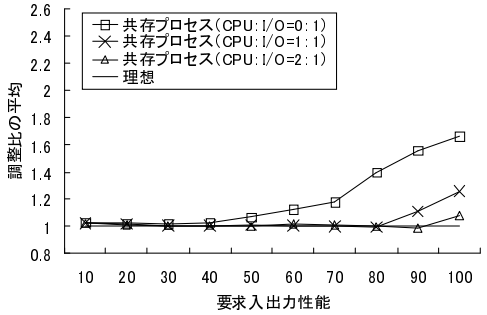
Vol.2010-ARC-189 No.10
Vol.2010-OS-114 No.10
2010/4/22



(A) 被調整プロセス（CPU:I/O=0:1）



(B) 被調整プロセス（CPU:I/O=1:1）



(C) 被調整プロセス（CPU:I/O=2:1）

図 4 共存プロセスを 1 つだけ同時走行させた場合の調整比

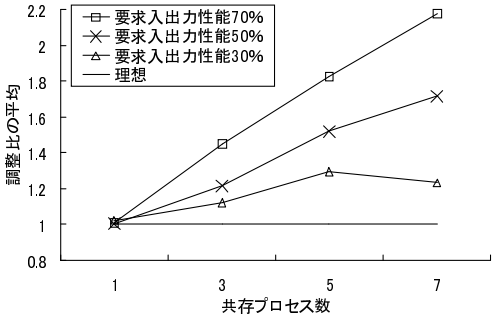


図 5 共存プロセス数と調整比

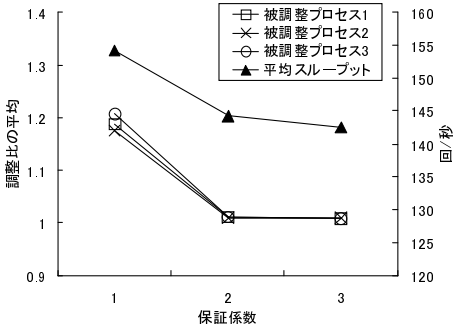


図 6 保証係数と被調整プロセスの調整比，およびスループット

3.1.3 保証係数とスループットの関係

保証係数とスループット（1 秒当たりの実 I/O 回数）の関係を明らかにする．3 つの被調整プロセス（いずれも処理比率は 2：1，要求入出力性能は 15 %）と 6 つの共存プロセス（最大の I/O 処理を要求する場合，処理比率は 0：1）を走行させた場合における保証係数，調整比（平均値），およびスループットの関係を図 6 に示す．2.2 節で述べたように，保証係数が小さいとスループットは向上し，大きいと調整精度を保証できることがわかる．例えば，保証係数が 3 の時のスループットは約 142 回/秒であるが，1 の時は約 154 回/秒である．一方，保証係数が 3 の時の調整比は約 1.0 であるが，1 の時は約 1.2 である．

© 2010 Information Processing Society of Japan

Vol.2010-ARC-189 No.10
Vol.2010-OS-114 No.10
2010/4/22

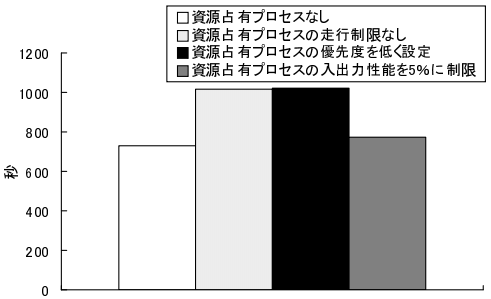


図 7 カーネルコンパイル時間

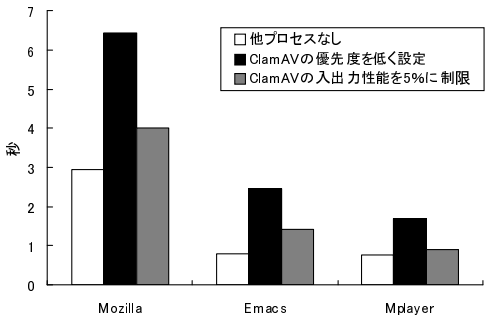


図 8 サービス起動時間

3.2 サービス評価

3.2.1 カーネルコンパイル時間

資源競合時の影響を抑制する効果を明らかにするため，提案制御法または優先度制御によりバックグラウンドプロセスの走行を制限した場合について，フォアグラウンドプロセスの実行時間を比較する．具体的には，バックグラウンドプロセスとして磁気ディスク装置からのランダム読み込みを行うプロセス（以降，資源占有プロセスと名付ける），フォアグラウンドプロセスとしてカーネルコンパイルを行うプロセスを走行させ，カーネルコンパイルに要する時間（以降，カーネルコンパイル時間と名付ける）を比較する．

測定結果を図 7 に示す．図 7 より以下のことがわかる．

（1）資源占有プロセスの優先度を低く設定しても，カーネルコンパイル時間の増加を抑制できない．例えば，資源占有プロセスの優先度を低く設定する場合，資源占有プロセスによるカーネルコンパイル時間の増加は約 40 %であり，資源占有プロセスの走行を制限しない場合とほぼ同じである．これは，資源占有プロセスの CPU 割り当てを抑制しても，資源占有プロセスの入力処理が頻繁に行われ，カーネルコンパイルの処理における入力処理を妨害するためである．

（2）資源占有プロセスの入出力性能を制限することにより，カーネルコンパイル時間の増加を抑制できる．例えば，資源占有プロセスの入出力性能を 5 %制限する場合，カーネルコンパイル時間の増加は約 6 %になる．

以上のことから，資源競合時において，フォアグラウンドプロセスの実行速度低下を抑制できるといえる．

3.2.2 サービス起動時間

次に，提案制御法または優先度制御によりバックグラウンドプロセスの走行を制限した場合について，フォアグラウンドプロセスを起動してから利用者にウィンドウを表示するまでの時間（以降，サービス起動時間と名付ける）を比較する．具体的には，バックグラウンドプロセスとしてウィルス対策ソフトである Clam Antivirus 0.92（以降，ClamAV と呼ぶ）を走行させた．また，フォアグラウンドプロセスとして，Web ブラウザである Mozilla 1.7.13（以降，Mozilla と呼ぶ）とテキストエディタである Emacs 22.1.2（以降，Emacs と呼ぶ），および動画再生ソフトである Mplayer 1.0rc1（以降，Mplayer と呼ぶ）を起動した．測定結果を図 8 に示す．図 8 より以下のことがわかる．

（1）バックグラウンドプロセスによるサービス起動時間の増加は大きい．例えば，ClamAV の優先度を低く設定する場合，Mozilla および Mplayer のサービス起動時間は約 2.2 倍，Emacs では約 3.0 倍となる．

（2）バックグラウンドプロセスの入出力性能を制限することにより，サービス起動時間の増加を抑制できる．例えば，提案制御法により ClamAV の入出力性能を制限する場合，サービス起動時間を平均して約 1.5 倍，最大でも，約 1.8 倍に抑制できる．これは，ClamAV の入出力性能を制限することにより，フォアグラウンドプロセスの入出力処理が妨害されなくなるためである．

以上のことから，資源競合時において，フォアグラウンドプロセスの起動時間を抑制できることがわかる．

© 2010 Information Processing Society of Japan

Vol.2010-ARC-189 No.10
Vol.2010-OS-114 No.10
2010/4/22

4. おわりに

調整精度が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として，入出力要求数を調整する制御法を提案した．提案制御法は，要求入出力性能から算出される理想の入出力時間内に被調整プロセスの実 I/O 処理が終了できるよう，デバイスドライバへの入出力要求数を制限する．これにより，入出力性能の調整精度を保証する．また，被調整プロセスの実 I/O 処理終了後に遅延処理を実施し，被調整プロセスの入出力性能を制限する．

実装と評価により，提案制御法は，共存プロセスが存在する場合でも，要求入出力性能が 50 %未満であれば，高い精度で調整できることを明らかにした．なお，要求入出力性能が 50 %以上であっても，磁気ディスク装置からの入力処理の比率が低い，または走行する共存プロセス数が少ないと，高い精度で調整できる．さらに，複数のプロセスが共存しても実 I/O 要求数を制限することで，ややスループットは低下するものの高い精度で調整できることを示した．資源競合時の影響についても，優先度制御では，影響を抑制できない（約 40 %のカーネルコンパイル時間増加）が，提案制御法は影響を約 6 %に抑制できることを示した．また，提案制御法では，サービスの起動時間を優先度制御の約半分に抑制でき，バックグラウンドプロセスの影響を抑制できることを示した．

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) 長尾 尚, 谷口 秀夫：“ 入出力性能の制御によりプログラム実行速度を調整する制御法, ” 電子情報通信学会技術研究報告書, vol.109, no.237, pp.33-38 (2009.10).
5) Margo I. Seltzer, Peter M. Chen and John K. Ousterhout, “ Disk Scheduling Revisited, ” Proceedings of the Winter 1990 USENIX Technical Conference, (1990.01)
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) R. Abbott and H. Garcia-Molina, “ Scheduling Real-Time Transactions: A Performance Evaluation, ” Proceedings of the 14nd International Conference on Very Large Databases, pp.1–12, (1988.08)
8) Anna Povzner, Tim Kaldewey, Scott Brandt, Richard Golding, Theodore M. Wong and Carlos Maltzahn, “ Efficient Guaranteed Disk Request Scheduling with Fahrrad, ” EuroSys ’08, pp.13–25, (2008.04)
9) Ting Yang, Tongping Liu, Emery D. Berger, Scott F. Kaplan and J. Eliot B. Moss, “ Redline: First class support for interactivity in commodity operating systems, ”8th USENIX Symposium on Operating Systems Design and Implementation, pp.73–86, (2008.11)
10) Bianca Schroeder, Mor Harchol-Balter, Arun Iyengar, Erich Nahum, Adam Wierman, “ Priority in DBMS Resource Scheduling, ” , pp.1–12, (2006.04)
11) M. J. Carey, R. Jauhari, M. Livny, “ How to determine a good multi-programming level for external scheduling, ” Proceedings of the 15th VLDB Conference, pp.397–410, (1989.07)
12) 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)

© 2010 Information Processing Society of Japan