プロセッサ使用量の制御による プログラム実行速度調整法のライブラリ実装

境　講一[†]　田端　利宏[‡]　箱守　聰[‡*]　谷口　秀夫[‡]

[†]岡山大学工学部　[‡]岡山大学大学院自然科学研究科
*株式会社 NTT データ技術開発本部

1. はじめに

プロセッサ使用量の制御によりプログラム実行速度を調整する機能として，プロセスのスケジュール法を工夫する方法がオペレーティングシステム(以降, OS と略す)内に実現されている[1], [2]. しかし，OS 内実装では，多くの OS 上でプログラム実行速度を調整することは難しい.

そこで，この機能をライブラリとして実現できれば，OS の作成環境が不要になるため，多くの OS 上で本機能を利用できるようになる.

本稿では，プロセッサ使用量の制御によるプログラム実行速度調整法をライブラリとして実装し，その評価結果を報告する.

2. 基本的な制御法

基本的な制御法として，プロセスがプロセッサを使用した量を測定し，その量と測定の程度からプロセスを一定時間停止させる. この処理をライブラリとして実装する. 具体的には，プロセスが発行するシステムコール間の時刻差の時間をプロセッサ使用量とし，その量と指定された調整の程度から，制御量を決定する. また，制御量に基づきプロセスを一定時間停止させる.

3 実装

3.1 システムコール発行の流れ

FreeBSD 4.3-RELEASE(以下, FreeBSD)におけるシステムコール発行の流れを図 1 に示す.

応用プログラム(以降, AP と略す)内のシステムコール呼出およびライブラリコール呼出により，ライブラリ内の RSYSCALL(x)が呼び出される. ここで x は，AP で依頼されたシステムコール名を指す. RSYSCALL(x)は，KERNCALL を利用して，システムコールの処理を OS に依頼する. OS では，対応するシステムコールの処理を行う.

A library program for controlling the execution speed of a program by regulating amount of processor use
Koichi Sakai, Toshihiro Tabata, Satoshi Hokomori, and Hideo Taniguchi‡
† Faculty of Engineering, Okayama University
‡ Graduate School of Natural Science and Technology, Okayama University
* Research and Development Headquaters, NTT Data

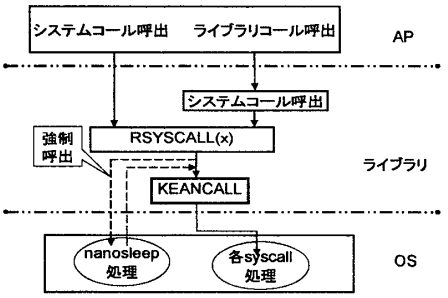

図 1　システムコール発行の流れ

ここでは，プロセスを一定時間停止させるために nanosleep システムコールを利用する. 具体的には，KERNCALL の直前に強制的に nanosleep システムコールを呼び出すようにする.

3.2 制御処理の流れ

制御処理の流れを図 2 に示し，以下に説明する.

(1) プロセッサのシステムカウンタを用いて現時刻を取得する.
(2) 共有メモリから，開始時刻(後述する)と調整の程度を取得する.
(3) 開始時刻，現時刻，および調整の程度から，制御量を算出する.
(4) 算出した制御量に基づき，nanosleep システムコールを呼び出し，一定時間停止する.
(5) AP に依頼されたシステムコール処理を呼び出す.
(6) プロセッサのシステムカウンタを用いて開始時刻を取得し，共有メモリに保存する.

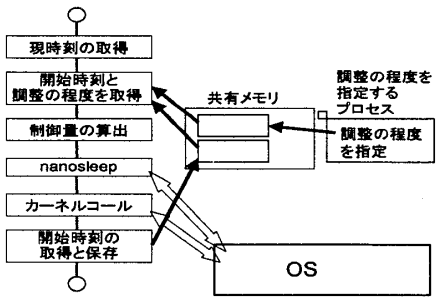

図 2　制御処理の流れ

上記のように，共有メモリを介して調整の程度を取得することで，プログラム実行速度を調整するプロセスと被調整プロセスを分離する．これにより，任意の AP プログラム実行速度の調整を可能にできる．

4 評価と考察

4.1 測定環境

Celeron(2.8GHz)プロセッサを搭載した計算機で FreeBSD を走行(シングルユーザモード)させ，実装したライブラリを使ってプログラムの処理時間を測定する．

プログラムとして 2 種類用意した．1 つは，CPU 処理(特定のメモリ領域のインクリメントを繰り返す処理)と getpid システムコール発行を繰り返し行うプログラム A である．もう 1 つは，CPU 処理と入力処理(DK の raw デバイスから 512byte 読み込む処理)を繰り返し行うプログラム B である．

4.2 結果と考察

プログラムの実行速度の程度を 10%から 90%まで調整したときの測定結果を図 3 から図 5 に示す．

図 3 はプログラム A において CPU 処理の時間が 10 ミリ秒の場合であり，図 4 はプログラム A において CPU 処理の時間が 100 ミリ秒の場合である．図 5 は，プログラム B において CPU 処理の時間が 100 ミリ秒の場合である．各図から，以下のことがわかる．

(1) 図 4 と図 5 から，CPU 処理の時間が大きいとき，実 I/O 処理の有無に関わらず，うまく性能調整が行えることがわかる．
(2) 図 3 から，CPU 処理の時間が小さくても，プログラム実行速度を大きく遅らせる場合(プロセッサ性能 40%以下)は，性能調整可能であるといえる．
(3) 図 3 から，CPU 処理の時間が小さく，かつプログラム実行速度をあまり遅らせない場合(プロセッサ性能 50%以上)は，性能調整を十分に行えないといえる．これは，nanosleep システムコールの精度が約 10 ミリ秒であるためと推測する．

5. おわりに

プロセッサ使用量の制御によるプログラム実行速度調整法をライブラリとして実装し，その評価結果を報告した．遅延時間が 10 ミリ秒以上のときうまく性能調整できる．しかし，10 ミリ秒以下のとき，性能調整を十分に行えない．これは，システムコール nanosleep の精度に起因する．

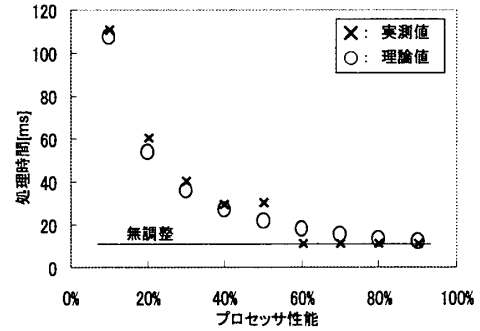

図 3　getpid 処理(CPU 処理時間:10 ミリ秒)

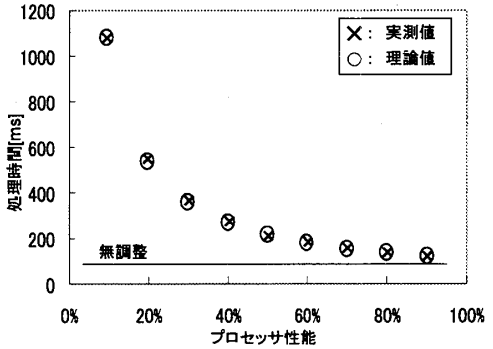

図 4　getpid 処理(CPU 処理時間:100 ミリ秒)

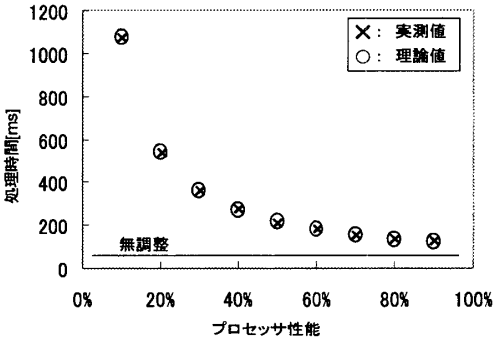

図 5　DK の read 処理(CPU 処理時間:100 ミリ秒)

残された課題として，入出力処理量が多いプロセスを走行させた場合，および他プロセスが共存した場合での性能調整についての評価がある．

謝辞　本研究の一部は，科学研究費補助金 若手研究(B)（課題番号 18700030）による．

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，“プロセススケジュールの制御によるプログラム実行速度調整法の評価，”信学論(D-I)，vol. J83-D-I，no. 1，pp. 184-193，Jan. 2000.