1L-02

マルチコア環境におけるスループット向上のための システムコール並列処理機構

三木 祐二† 芝 公仁†
†龍谷大学理工学部

1 はじめに

アプリケーションプログラムは，カーネルの提供する機能を使用するためにシステムコールを用いる．システムコールの発行は，カーネルモードとユーザモードの遷移を発生させ，メモリキャッシュや TLB が汚染するため，アプリケーションプログラムの処理性能の低下を引き起こす可能性がある [1].

本稿では，これらの問題を解決するシステムコール並列処理機構を提案する．本機構では，システムコールを要求するスレッドとシステムコールを処理するスレッドに分け，共有メモリを用いてシステムコールを発行する．そのため，システムコールの発行はメモリの読み書きで行えるため，カーネルモードとユーザモードの遷移が発生しない．本機構では，システムコールを処理するスレッドを複数動作させることができるため，複数のシステムコールを並列に処理できる．また，システムコールは，アプリケーションプログラムとは別のスレッドで処理されるため，アプリケーションプログラムは，ディスク I/O などを発生させるシステムコールを要求しても，ブロックされない．

本稿では，システムコール並列処理機構の構成と動作を示し，従来のシステムコールと比較し，本機構ではシステムコール発行のオーバヘッドを削減できることを示す．

2 提案機構

2.1 構成

システムコール並列処理機構の構成図を図 1 に示す．本機構では，システムコールを処理するためのスレッド (以降，PSC スレッド) を作成し，PSC スレッドを管理するシステムコール処理部と，ユーザプロセスがデータの受け渡しをするために共有メモリを使用する．ユーザプロセスがシステムコール処理部に対して，命令を送るためのインタフェースとして以下のシステムコールを提供する．

- **pscregist:** 共有メモリをシステムコール並列処理機構に登録する．
- **pscenter:** pscenter システムコールを発行したスレッドが PSC スレッドとして動作し，要求されたシステムコールを処理する．
- **pscnotify:** ユーザプロセスが，システムコール処理部に対して，システムコールの要求をしたことを通知する．
- **pscwait:** ユーザプロセスが，システムコール処理部に対して，システムコールの要求をしたことを通知した後，結果が受け取り可能になるまで待機する．
- **pscexit:** システムコール並列処理機構に登録されている共有メモリを解除し，PSC スレッドを終了させる．

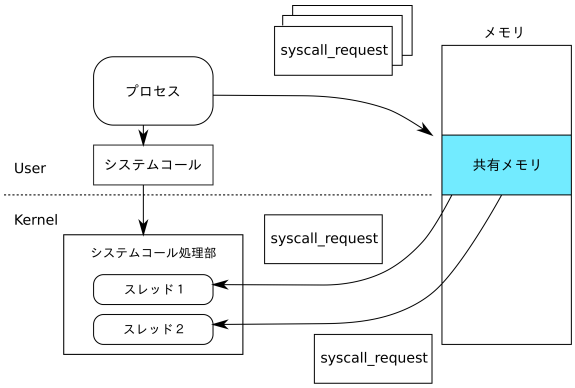

図 1 並列システムコール処理機構の構成図

共有するメモリの構造について図 2 に示す．ユーザプロセスがシステムコール処理部にシステムコールの処理を要求するときは，適切な値が代入された syscall_request 構造体をリクエスト領域に書き込む．syscall_request 構造体にはシステムコール番号，引数，戻り値，リクエストの状態が保存されている．リクエスト領域には，複数の syscall_request 構造体を持つことができ，syscall_request 構造体 1 つにつき，1 つのシステムコールが対応する．PSC スレッドはリクエスト領域内のデータを読み込み，システムコールを処理する．

2.2 動作

本機構を使用するためには，まず PSC スレッドを動作させる必要がある．PSC スレッドを動作させるには，共有メモリを確保し，pscregist システムコールを発行し，共有メモリを本機構に登録する．登録に成功すると，確保した共有メモリに対応した識別子が取得できる．その識別子を用いて，pscenter システムコールを発行することで，スレッドはカーネルモードに遷移し，PSC スレッドとしてシステムコールを処理し始める．

High-throughput parallel processing mechanism of system calls for multi-core environments
Yuji Miki†, Masahito Shiba†
†Faculty of Science and Technology, Ryukoku University

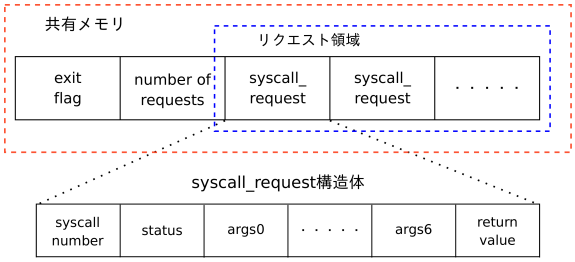

図 2　共有メモリの構造

PSC スレッドは，システムコールの要求がリクエスト領域にあるかどうかを確認し，あればその要求に応じてシステムコールを処理し，なければユーザプロセスから通知があるまで待機状態となる．

ユーザプロセスは，システムコール処理部にシステムコールの要求を渡すときはリクエスト領域に syscall_request 構造体として要求を書き込み，pscnotify システムコールを発行し，システムコールを処理するように要求する．要求を受けたシステムコール処理部が PSC スレッドを実行状態にし，システムコールの処理を開始する．PSC スレッドは，システムコールを処理し終えると syscall_request 構造体に保存されているリクエストの状態が処理の完了を示す値に更新される．ユーザプロセスは，syscall_request 構造体のリクエストの状態が処理完了となっているものを探し，当該 syscall_request 構造体からシステムコールの結果を受け取る．ユーザプロセスが，pscexit システムコールを発行すると，登録されていた共有メモリは解除され，PSC スレッドは終了する．

PSC スレッドは，複数のスレッドで動作することが可能であるため，それぞれのスレッドが別のシステムコールを並列に処理することができる．これによって，ユーザプロセスが発行するシステムコールを同時に実行でき，スループットの向上が期待できる．また，システムコールを要求するスレッドとシステムコールを処理するスレッドは異なるため，ディスク I/O などが発生するシステムコールを要求したとしても，システムコールを発行した側はブロックされない．

3　評価

システムコール並列処理機構を用いたシステムコールの発行と従来のシステムコールの発行にかかるオーバヘッドを測定し，比較するための実験を行った．実験環境を表 1 に示す．本実験では，127 回 gettid システムコールのリクエストを従来のシステムコールと提案手法で処理させ，それぞれの手法がすべてのリクエスト処理し終える時間を計測した．提案手法では，一度に要求するシステムコールの数を変化させた．gettid システムコールはカーネル内でほとんど処理をしないため，gettid システムコールの処理時間を測定することは，システムコールを発行する時間に相当する．図 3 に測定結果を示す．横軸は提案手法の一度に要求するシステムコールの数を示し，縦軸は処理時間を示す．

表 1　実験環境

項目	値
CPU	Intel Core i5–4690 CPU 3.5GHz
カーネル	Linux–4.19.5
ライブラリ	GNU C Library 2.27
共有メモリ	4096 バイト

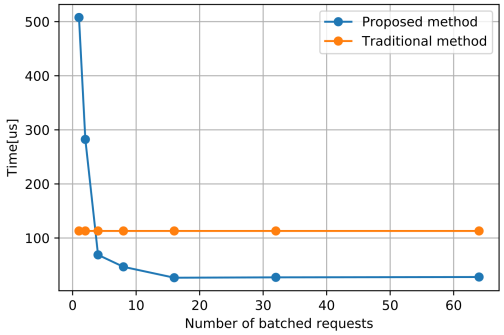

図 3　127 個の gettid システムコールを処理する時間

一度に要求するシステムコールの数が 2 以下のときは，提案手法は従来のシステムコールに比べ，処理時間が長くなる．これは，提案手法では，リクエストを要求する度に，pscnotify システムコールを発行しており，pscnotify は gettid システムコールに比べ，処理に時間がかかる．そのため，一度に要求するシステムコールの数が少ないと，pscnotify システムコールが発行される数が増えてしまい．処理時間の増加につながる．一度に要求するシステムコールの数が 3 以上のときは，従来のシステムコールに比べ，提案手法の処理時間は短くなる．これは，システムコール発行によるユーザとカーネルの遷移のオーバヘッドを削減できたためと考えられる．一度の要求数が 16 以上のとき処理時間を最大で，76% 削減できた．

4　おわりに

本稿では，システムコール発行によるオーバヘッドを削減するシステムコール並列処理機構を提案した．本機構では，システムコールを処理するスレッドを作成し，共有メモリを読み書きすることでシステムコールの発行を行う．本機構を用いたシステムコールの発行は，従来のシステムコールと比較し，一度に要求するシステムコールの数が多ければ，ユーザモードカーネルモードの遷移回数を削減し，システムコール発行のオーバヘッドを削減できる．

参考文献

[1] Livio Soares and Michael Stumm.:FlexSC:Flexible System Call Scheduling with Exception-Less System Calls, *OSDI* (2010)