1J-08

共有メモリとカーネルスレッドを用いた非同期システムコール処理機構

中島 健† 芝 公仁†
†龍谷大学理工学部

1 はじめに

ユーザプロセスがカーネルへアクセスするためにはシステムコールを使用する．プロセスはシステムコールを発行するとシステムコールの実行完了を待つ必要がある．また，システムコール発行にはカーネルへ処理を移行するため，速度の低下が考えられる [1]． 本稿では非同期システムコール処理機構について述べる．本機構により，システムコールを非同期発行することでプロセスがシステムコールの処理を待つことなく処理することを可能にする．また，システムコールを発行するための情報をカーネルと共有する領域に書き込むことでシステムコールの回数を減らすことで速度低下を低減させる．

2 提案機構の構成

プロセスはカーネルの機能を使用するためにシステムコールを使用する．従来ではシステムコールを発行していたが，一度システムコールを発行するとプロセスがシステムコールの完了を待つ必要がある．これを解決する非同期システムコール処理機構を提案する．従来のシステムコールと提案した機構との比較を図 1 に示す．プロセスがカーネルにアクセスするためにはシステムコールを発行する必要がある．システムコールを使用してカーネル内の処理を行うことで意図しない動作を防ぐことができるが，システムコールを発行するとその完了を待つ必要があり，ファイルの読み書きのような長い時間を要する処理では速度の低下につながる．提案機構では，システムコールを発行するために必要な情報を syscall_info というメモリ上のデータで一括して管理する．syscall_info にはシステムコール番号や引数，返り値などシステムコール発行に必要な情報が含まれている．

syscall_info を用いたシステムコールの処理を図 2 に示す．syscall_info は state，ret，num の 3 つの値と arg の配列を保持する．state は当該 syscall_info の状態を

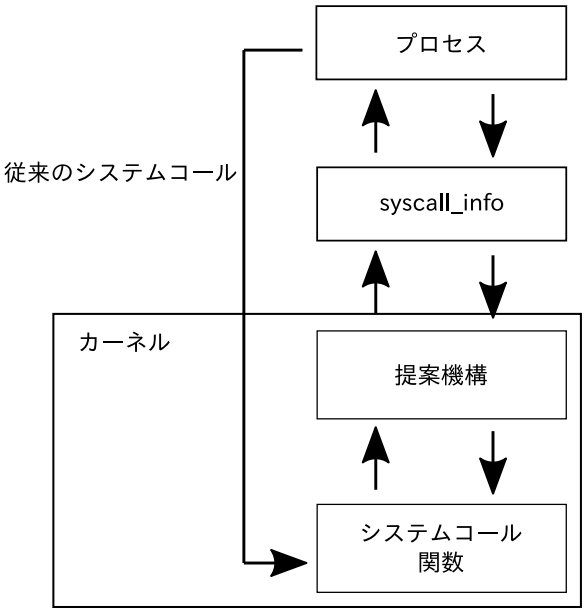

図 1 従来のシステムコールと提案機構の比較

表す．状態には，NONE，HASCALL，EXECUTING，COMPLETED，END の 5 つがある．NONE はその syscall_info に要求が入っていない状態を表す．HASCALL はまだ実行されていない要求が格納されている状態，COMPLETED は処理が完了して結果を保持している状態である．num はシステムコール番号を格納している．arg は最大 6 個あるシステムコール関数の引数を配列として所持している．ret はシステムコールの実行結果を保持する変数である．

提案機構は async_syscall_init, async_syscall_exec のシステムコールを提供する．async_syscall_init はプロセスが作成した syscall_info のメモリ領域を共有するためのシステムコールで，作成された領域の先頭アドレスを引数として取る．async_syscall_exec は syscall_info の state の状態に応じてシステムコールを実行するシステムコールである．提案機構は syscall_info の num に対応するシステムコール関数を実行し，実行結果を ret に返却する．システムコール関数を実行するためには引数をレジスタに設定する必要がある．do_syscall は arg に格納された引数をレジスタ構造体に設定する役割を持っている．

Asynchronous system call processing using shared memory and kernel threads
Ken Nakajima† and Masahito Shiba†
†Faculty of Science and Technology, Ryukoku University

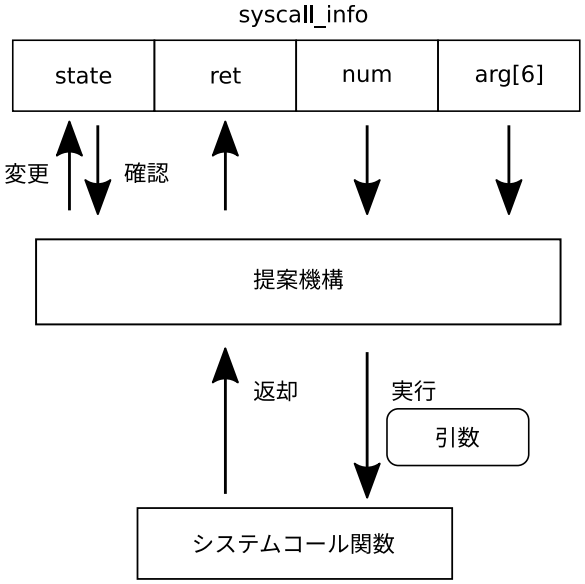

図 2　syscall_info を用いたシステムコールの処理

3　提案機構の動作

プロセスは syscall_info を作成するにあたり syscall_info 用のメモリ領域を決める．領域を決めた後，async_syscall_exec を用いて先頭アドレスをカーネルと共有する．プロセスはアドレスをカーネルと共有した後，システムコールの要求を処理するためのスレッドを作成する．作成したスレッドは async_syscall_exec により，カーネル内でメモリの監視と要求の処理を行う．

提案機構は主に 2 つの動作がある．1 つ目は syscall_info の state を操作すること，2 つ目は syscall_info の要求からシステムコール関数を実行して結果を syscall_info の ret に返すことだ．まず，async_syscall_exec は syscall_info の state を確認する．もし，state が HASCALL であれば state を EXECUTING に変えて要求の処理を開始する．また，NONE や EXECUTING，COMPLETED であればその要求は無視する．END であれば async_syscall_exec を終了する．async_syscall_exec はシステムコールの実行と syscall_info への実行結果の返却を行う．async_syscall_exec はシステムコール関数の実行前に syscall_info の arg に格納されている引数をレジスタ構造体に設定する．そして，先程設定したレジスタ構造体を引数として，syscall_info の num に格納された番号に対応するシステムコール関数を実行する．実行後，関数の実行結果が返却されるため，それを syscall_info の ret に格納する．これで提案機構の一連の動作が終了する．以上の処理によって syscall_info には処理の結果が格納され，state は COMPLETED の状態に変化する．プロセスは syscall_info の state を確認し，この state が COMPLETED の状態であればシステムコールの実行結果を取得する．プロセスはシステムコールの要求があれば syscall _info の state を NONE に変更し，要求を書き込む．async_syscall_exec を終了させたい場合は state を END に変更する．に変更する．従来のシステムコール発行に比べ，本機構ではプロセスのシステムコール syscall_info 作成と提案機構による syscall_info の処理を別スレッドで動作させている．これらを分離することで一度に複数のシステムコールを発行した場合であってもプロセスの処理を中断させることなくシステムコールの処理を行うことが可能となる．

4　おわりに

本稿では非同期システムコール処理機構について述べた．本機構はシステムコールの処理を専用のスレッドが行うことにより，プロセスの処理とシステムコールの非同期処理を行う．これにより，プロセスがシステムコールの完了を待つ必要なく処理を続けることが可能となる．

参考文献

[1] Soares, L. and Stumm, M.: FlexSC: Flexible System Call Scheduling with Exception-Less System Calls., *OSDI* (Arpaci-Dusseau, R. H. and Chen, B., eds.), USENIX Association, pp. 33–46 (2010).