【 108 】

氏名	安倍広多（あべこうた）
博士の専攻分野の名称	博士（工学）
学位記番号	第 15081 号
学位授与年月日	平成12年2月22日
学位授与の要件	学位規則第4条第2項該当
学位論文名	UNIX上でのマルチスレッド機構の実現
論文審査委員	（主査） 教授 谷口 健一 （副査） 教授 萩原 兼一　教授 井上 克郎　教授 東野 輝夫

論文内容の要旨

1つのプロセスの中に、複数の制御の流れ（スレッド）を作り出し、それぞれに独立した処理をさせることができる機構（マルチスレッド機構）が注目されている。マルチスレッドを用いたプログラムでは、複数のスレッドがCPUを切り替えながら見掛け上同時に動作する。マルチスレッドは、複数のプロセスを用いるよりも実行効率が良い。

UNIX上で動作する既存のマルチスレッド機構（マルチスレッドを実現するプログラム）は、いずれも特定のCPUに依存しているため、移植が困難であった。本論文では、移植性の良いマルチスレッド機構の実現法を提案している。移植性を保つ上で、CPUのスタックポインタレジスタの設定を、CPUに依存せずに行う方法が問題となるが、この問題をUNIXのシグナルスタック機構（割り込み処理を、指定したメモリ領域をスタックとして実行する機構）を利用して解決する方法を提案している。また、将来拡大する余地を残しながらスレッドのスタック領域を確保することで、スタック領域が不足した場合に自動的に拡大する方法を提案している。

UNIX上に実装したマルチスレッド機構では、1つのスレッドが発行するシステムコールによって、他のスレッドの実行が中断する問題があるが、システムコールの発行を他のプロセスに代行させることで解決する方法を提案している。また、この時に発生するプロセス間通信のコストを削減するため、プロセス間でアドレス空間を共有する方法、プロセス間通信のシステムコールを可能な限り省略する方法を提案している。

マルチメディア処理等で、ユーザが指定した時間でなるべく正確にスレッドが処理を行えるようなマルチスレッド機構が関心を集めている。本論文では、このようなマルチスレッド機構を、非実時間OSであるUNIX上で実現する方法についても述べている。UNIXでは処理に必要なCPU資源を予約できないため、実際に利用できたCPU資源を測定し、アプリケーションに伝えることで、スレッドの周期を調整する方法を提案している。また、デッドラインに基づいてスケジューリングを行う方式も実装し、応用として時間拡張LOTOS（形式記述言語LOTOSに対し、イベントが実行可能な時刻の範囲を指定できるように拡張したもの）のコンパイラの実装法を示している。

単一のスレッドで実行されることを想定している既存のプログラムコードを、複数のスレッドから実行するためには、大域データをスレッド毎に分離する必要がある。従来、このためにはコードを修正する必要があったが、この方

法は手間がかかり、またソースコードが必要である。本論文では、この問題の解決法として、スレッド毎にメモリ領域を割り当て、その中にスレッドが実行するコードと大域データを配置する方法を提案している。大域データはスレッド毎に存在するため、既存のコードを修正することなく呼び出すことが可能である。さらに、既存のプログラムをスレッドとして実行する機構、プロセス間通信と同一のインタフェースでスレッド間通信を行う機構を設けることで、複数プロセスで構成された既存のアプリケーションを、軽微な修正でマルチスレッド化する方法も提案している。

論文審査の結果の要旨

本論文は、UNIX 上で動作するマルチスレッド機構に関する研究をまとめている。

まず、UNIX 上で動作する移植性の良いマルチスレッド機構の実現法を提案している。CPU 依存性が高いスタックポインタの設定などの処理をC言語とシステムコールだけを用いて行う方法を考案して実装し、それにより、マルチスレッド機構を様々な CPU 上の UNIX で動作させることが可能となった。また、スレッドごとに予め割り当てたスタック領域が不足した際にそれを自動的に拡大する方法も考案し、メモリ効率を向上させている。これらの方法を用いて実装したマルチスレッド機構の実行速度は、既存の CPU 依存のマルチスレッド機構と比べても遜色ないことを確認している。また、スレッドからのシステムコールを他のプロセスに代行させ、スレッド群の実行が中断されないような方法も提案している。この時生じるプロセス間通信を極力減らして性能の低下を防ぐ方法も考案している。提案する方法をファイル I/O に適用した結果、I/O 時間の大半(例えばローカルディスクへの 8 K バイト書き込み時で80%以上の時間)を他のスレッドが利用できること、I/O 完了時間は多少増えるが実用範囲内に収まっていることを確認している。

次に、非実時間 OS である UNIX 上で、可能な限り時間制約を満たしながら動作する実時間マルチスレッド機構の実現法について検討し、そのプロセスで実際に使用できた CPU 時間を測定してそれに基づいてスレッド周期を制御する方法を提案している。この方式により他プロセスの影響下でも安定したスケジューリングが可能なことを確認している。また、デッドライン時間順にスケジューリングを行なう方式も実装し、時間拡張 LOTOS コンパイラへ適用して、実装した実時間スレッド機構の有用性を示している。

さらに、単一のスレッドとして実行されることを想定している既存のプログラムコードを、複数のスレッドから利用できるようにするための方法を提案している。それにより、既存の膨大なプログラムコード群を、マルチスレッドプログラムから容易に利用することが可能となる。また、複数のプロセスで構成された既存のアプリケーションを、容易にマルチスレッド化する方法も提案している。マルチスレッド化することにより、アプリケーションの実行速度を向上させることができる。実際にこの方法を用いて UNIX のシェルパイプラインをマルチスレッド化することで、速度が向上することを確認している。

以上の研究成果は、マルチスレッドに関する技術の発展に貢献しており、本論文は博士（工学）論文として価値あるものと認める。