負荷分散向けプロセス制御用システムコールの作成

日大生産工(院)　○山﨑　淳史
日大生産工　　松田　聖

１ はじめに

近年、高速かつ可用性の高い環境を構築するための負荷分散が注目されている。負荷分散とは1台のコンピュータで実行している処理を複数のコンピュータに分散して実行する事である。そのためにはコンピュータ同士をネットワークで接続し、負荷の高いコンピュータから負荷の低いコンピュータへ処理を移す必要がある。

本研究ではこの負荷分散を実現するためのプロセス制御を行うシステムコールを作成する。具体的にはMINIX上で動作中のプロセスの状態をファイルとして保存するfreezeシステムコールと、ファイルの状態のプロセスをプロセスとして処理を再開させるdefreezeシステムコールの作成を行う。

２ MINIX

MINIXはA・S・タネンバウムによって開発されたUNIXクローンのOS である。ソースコードはC言語で記述され、オープンソースソフトウェアとして全て公開されている。

２.１ MINIXの構成

MINIXはプロセス管理、入出力タスク、サーバプロセス、ユーザプロセスの4つのレイヤで構成される。

・レイヤ1　プロセス管理
　割り込み処理、スケジューリング、メッセージ通信等を受け持つ。
・レイヤ2　入出力タスク
　各デバイスごとにタスクが存在しデバイスドライバとして動作する。
・レイヤ3　サーバプロセス
　メモリ管理関連のシステムコールを実行するメモリマネージャ、ファイル関連のシステムコールを実行するファイルシステムが存在する。
・レイヤ4　ユーザプロセス
　シェル、エディタ等のユーザプログラムが動作する。

MINIXではレイヤ1とレイヤ2のコードがカーネルとしてまとめられ、リソース管理を行う。システムコールの処理はサーバプロセスとしてレイヤ3で行なわれる。各レイヤは上位レイヤに抽象化した概念を提供する。

２.２ システムコール

ユーザがディスクからファイルの読み書きを行うプログラムを作成する場合、ディスクの回転やヘッドの制御など、全ての動作をプログラミングするのは困難である。またユーザプロセス自身が新しくプロセスを作成する事は出来ない。このようなユーザがプログラミングする事が困難な処理やユーザプロセスが実行できない処理は、OSがシステムコールとしてその機能を提供している。システムコールはユーザプロセスよりも低いレイヤで実行されるため、より高い優先度と権限を持つ。

３ 本研究における負荷分散の概要

複数のコンピュータが存在する環境での負荷分散を考える。各コンピュータ上には負荷の監視とプロセスの受け渡しを行うデーモンを設置する。このデーモンは負荷の高いコンピュータで動いているプロセスに対してfreezeシステムコールを用いて一時停止させ、ファイルとして保存したプロセスの状態を他の負荷が低いコンピュータへ転送する。受け取ったコンピュータのデーモンは、その

Development of Process Control System Call for Load Balancing

Atsushi YAMAZAKI and Satoshi MATSUDA

データを元にdefreezeシステムコールを用いてプロセスを再開させる。

単一のコンピュータ上での負荷分散を考える。この場合はコンピュータの負荷が高まった時に長時間CPUを占有しているプロセスの一時停止を行う。そうすることで他のプロセスが優先的に処理されるので平均ターンアラウンドタイムは短くなり、全体として性能が向上すると考えられる。その後、コンピュータの負荷が軽減された時点で停止させたプロセスを再開させる。

これらの流れのうち、ユーザプロセスでは実行できないプロセスの一時停止と再開を行うシステムコールを作成する。

３.１ freezeシステムコール

実行中のプロセスにはそれぞれメモリが割り当てられ、プロセスに関連する情報はプロセステーブルに格納されている。プロセスを一時停止させ、他のコンピュータで処理を再開するためにはこれら全ての情報を保存しておかなければならない。そこでプロセス一時停止システムコールは指定したプロセスのメモリの内容とプロセステーブルをファイルとして保存し、プロセスの停止を行う。図2にその手順を示す。

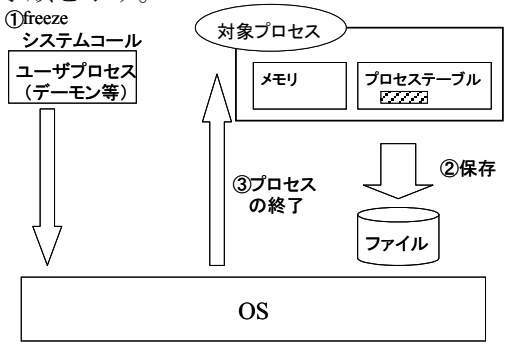

図2　freezeシステムコール

①デーモンは、処理を一時停止させ、状態を保存したいプロセスに対してfreezeシステムコールを実行する。
②freezeシステムコールはプロセステーブルの情報からメモリ空間を参照し、内容をファイルとして保存する。同時にプロセステーブルの情報もファイルとして保存する。
③freezeシステムコールは保存が全て終了したプロセスに対してkillシステムコールで処理を終了させる。

これらの処理を行い、プロセスをファイルの状態で一時停止させる。

３.２ defreezeシステムコール

ファイルの状態で停止させたプロセスを、停止した時の状態から処理を再開させるためのシステムコールである。このシステムコールは新しくプロセスを生成し、そのメモリとプロセステーブルをファイルとして保存したデータに置き換える。図3にその手順を示す。

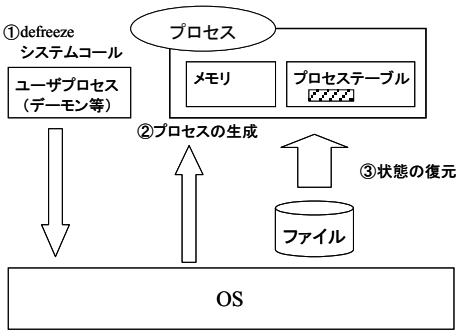

図3　defreezeシステムコール

①デーモンはdefreezeシステムコールを実行する。
②defreezeシステムコールは、forkシステムコールで新たにプロセスを生成する。
③ファイルから読み込んだプロセスのデータを、②で生成したプロセスのメモリとプロセステーブルに書き込む。

これらの処理を行うことで新たに生成されたプロセスは、一時停止される直前のプロセスと全く同じ状態になり、処理を再開する。

４ まとめ

本研究では、負荷分散向けプロセス制御用システムコールの作成を行なっている。今後は本システムコールを使用した負荷分散用デーモンを作成し、実際にネットワークで接続した複数のコンピュータ上での検証を行う。また作成したシステムコールを使い、負荷の監視と、自動的に負荷分散を行うシステムの構築も考えたい。

「参考文献」

１）A・S・タネンバウム，A・S・ウッドハル，「オペレーティングシステム第2版 設計と理論およびMINIXによる実装」, ピアソン・エデュケーション, (1998)
２）桜田幸嗣，「MINIX C プログラミング」，アスキー，(1993)
３）森田茂男，多田好克，利用者レベルで実現したプロセス移送ライブラリ，情報処理学会研究報告，vol.1991 no.063，(1991)，pp. 41-47