3P-2　並列処理用ＯＳ・ＳＫＹ－１のメモリ管理方式

齊藤　雅彦　　山口　伸一朗　　上脇　正
(株) 日立製作所　日立研究所

１．はじめに

我々は並列処理用ＯＳ・ＳＫＹ－１ (System Kernel for You-1) の開発を行っている。ＳＫＹ－１では同一仮想空間上で並列に実行されるスレッドを導入する。これによれば、処理の切換え時間を短縮して計算機のスループットを向上させることができる反面、スレッド間ではデータの保護が存在しないというユーザインタフェース上の問題がある。本稿では従来プロセスと同程度のデータ保護が可能なメモリ管理方式：動的スタック拡張方式を提案し、その実現手法について言及する。

２．スレッドの導入によるメモリ管理の問題点

同一プロセス内のスレッドは仮想空間を共有するため、プロセス内の共通データを用いて高速かつ柔軟な同期、通信が可能となる。しかし、本来スレッドに固有なデータであるスタック領域も同一仮想空間上に存在する（図１）。スタック領域は関数（サブルーチン）呼出しなどによって動的に伸縮するため、あるスタック領域が他のスタック領域に侵入してデータを破壊してしまう可能性がある。

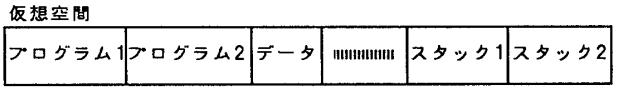

図１　スレッド導入時の仮想空間の使用法

スタック領域間で互いにデータを保護するためには以下の２つの手法が考えられる。
①あるスレッドのスタック領域が他のスタック領域に侵入しようとしたとき、そのスレッドあるいはプロセス全体を異常終了させる。
②スタック領域が他のスタック領域に侵入しようとしたとき、そのスタック領域を拡張する。
前者の手法はスレッドのスタック領域の境界に参照禁止領域（レッドゾーン）を設けることにより簡単に実現できる。しかし、スレッドの使用するスタック領域の最大値をユーザが予め計算するか、コンパイラ等にある程度の不正確さを許容して計算させる必要がある。ユーザインタフェースとして、従来プロセスと同程度のデータ保護を行わせるためには、上記②の手法の実現が望まれる。この手法を実現すれば、ユーザやコンパイラがスタック領域の大きさを考慮することなくスレッドを使用できる。

３．動的スタック拡張方式

３．１．動的スタック拡張方式の概要

動的スタック拡張方式とは仮想空間を複数個の領域に分割し、スタック領域にこれを単位として適宜仮想空間を割り当てる方式である。動的スタック拡張方式ではこの割当て単位となる領域をユーザページと呼ぶ。図２に動的スタック拡張方式の実行例を示す。図２は３個のスレッドが並列に動作する例である。各スレッドにはそれぞれ、初期状態として１つのユーザページが割り当てられている。

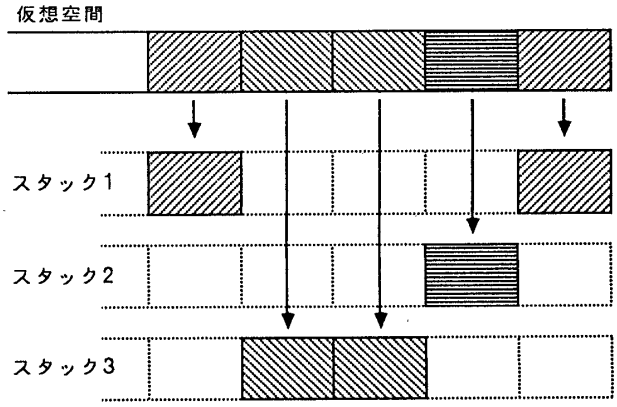

図２　スタック拡張手法

関数呼出しなどによりスタック領域が拡張され、割り当てられたユーザページをオーバフローする場合には、新たなユーザページを確保し、スレッドに割り当てる。関数からの復帰によって使用中のユーザページが不必要となった場合には、そのユーザページを解放する。

以上に示した動的スタック拡張方式を実現するに当たって、以下の問題点がある。従来プロセスではスタック領域は仮想空間の最後尾から前方へ連続して拡張されるものであったが、動的スタック拡張方式を使用すると、ユーザページ間で論理アドレスが不連続になる可能性がある。この問題を解決するため、スタック領域を、「ユニット」と呼ぶ、論理アドレス上連続しなければならない領域単位で拡張する。スタック領域拡張時に確保すべきユニットと使用中のユーザページの残量を比較し、ユニットを確保できない場合には、新たなユーザページを割り当てて、そこにユニットを確保する。具体的に、ユニットとは関数の引数と内部変数の領域である。これらの内部は配列などのデータにより論理アドレス上連続している必要があるが、引数と内部変数の領域間で論理アドレスが連続している必要はさらさらない。

３．２．動的スタック拡張方式の実現手法

Ｉ）構成要素

ユーザページは仮想空間を均等に分割する。ＳＫＹ－１の制限事項「スレッド数は１０２４以下」を考慮し、ユーザページは１Ｍバイトである。ユーザページのオーバフロー検出を行うため、ユーザページの最後尾４Ｋバイト（仮想記憶管理の１ページ）は参照禁止としてレッドゾーンに設定する。レッドゾーンの大きさはユーザページの０．４％程度にすぎない。

Multiprocessor Operating System:SKY-1 -Memory Management-
Masahiko SAITOH, Shinichiro YAMAGUCHI, Tadashi KAMIWAKI
Hitachi Research Lab., Hitachi Ltd.

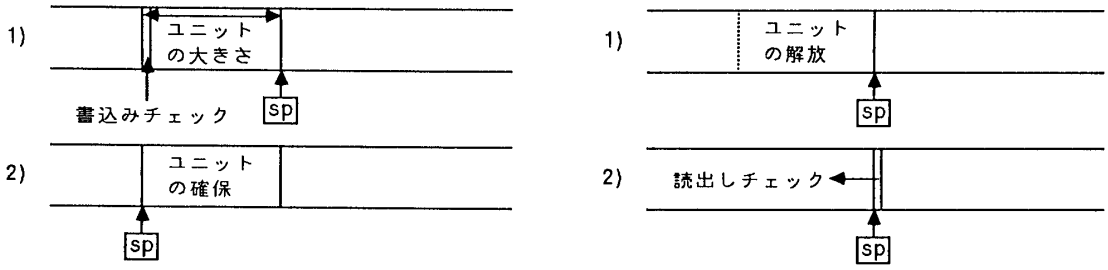

図3　ユニットの確保/解放処理

ユニットは関数の引数領域と内部変数領域である。各々を指すためのポインタとして、引数ポインタ（ａｐ）とフレームポインタ（ｆｐ）を用意する。この他に、現在のスタック領域の先頭を指すポインタとしてスタックポインタ（ｓｐ）がある。引数領域には実際の引数の他に、旧引数ポインタの値、関数のリターンアドレスを退避する場所を設ける。また、内部変数領域にはフレームポインタとユーザレジスタの旧値を退避する場所を設けている。

ユーザページが使用中か否かを示す空ユーザページ配列を用意する。この配列はユーザページが複数個連続して使用されているか否か（関数の内部変数で１Ｍバイト以上の領域を使用する場合、ユーザページを複数個まとめて確保する）の情報も有する。

Ⅱ）実現手法

ＳＫＹ－１上での動的スタック拡張方式はコンパイラ、ＯＳ両者によって実現される。図３に動的スタック拡張方式の関数呼出しおよび関数からの復帰に伴う、ユニットの確保／解放の方式を示した。コンパイラは図３に示した動作を行う命令列を生成する。基本的なアルゴリズムは以下の通りである。

①ユニットの確保：確保すべきユニットの先頭に何らかのデータを書き込んで、オーバフローか否かを検査した後、ユニットを確保する。

②ユニットの解放：ユニットを解放した後、スタック領域の先頭（スタックポインタの指す位置）を読み出してアンダフローを検査する。

オーバフローおよびアンダフローはユーザページ内に存在するレッドゾーンを参照するか否かによって検出できる。なお、レッドゾーンの大きさは４Ｋバイトのため、４Ｋバイト以上のユニットを確保する場合、複数回の検査が必要である。

レッドゾーンの参照によって割込みが発生すると、制御がＯＳに移行し、ユーザページの割当てまたは解放を行う。この処理は以下のように行っている。

①’ ユーザページの割当て：空ユーザページ配列を参照して未使用のユーザページを確保し、スタック領域に割り当てる。割り当てたユーザページ内にスタックポインタを移動するとともに、旧スタックポインタの値をレッドゾーン内に設けられたリンク領域に退避する。

②’ ユーザページの解放：上記①’でリンク領域に退避したスタックポインタの旧値を復帰する。さらに使用していたユーザページに対応する空ユーザページ配列のエントリを未使用に設定する。

ＳＫＹ－１では上記のように動的スタック拡張方式を実現している。動的スタック拡張方式を使用することによって、スレッドのスタック領域間のデータ保護を従来プロセスと同程度に高めることが可能である。また、スレッドの使用しているスタック領域の総和が常に最小に近い値（仮想記憶管理のページング方式と同様、内部フレグメンテーションが発生する）とすることができ、プロセス内に存在するスレッド数を増やすことができるという利点もある。

４．性能評価

動的スタック拡張方式はある程度の性能を犠牲にして、ユーザインタフェースを改善する方式である。あるスレッドのスタック領域が他のスタック領域を破壊しないように、関数呼出しおよび関数からの復帰時にオーバフローおよびアンダフローの検査処理を行う。これが性能低下の要因である。表１に関数呼出しの頻度の異なるプログラムを用意して、性能低下の度合いについて調査した結果を示す。関数呼出しを頻発するAckerman関数では２０％強も性能が低下している。しかし、通常の頻度で関数呼出しを行うプログラムでは、性能低下は５～６％以下で済んでいる（キャッシュやパイプライン状態のせいか、性能が向上してしまうものもある）。スレッドを導入することにより、処理の切換え時間を高速化できることを考えると、ほとんど問題とならない程度である。

表1　動的スタック拡張方式の性能低下率

	Ackerman関数 Acker(3,10)	文書変換処理	素数計算 2000000まで
従来方式	44.5sec	74.3sec	55.3sec
動的スタック拡張方式	54.2sec	78.3sec	54.8sec
性能低下率	22%	5.4%	-0.96%

注）文書変換処理は同一プログラムを1000回繰り返して実行した

５．まとめ

本稿ではスレッドを導入した場合にスタック領域を保護、拡張する動的スタック拡張方式とＳＫＹ－１上での実現手法について述べた。動的スタック拡張方式を使用すれば、従来プロセスと同程度のデータ保護が可能である。性能低下も５％程度に収まる見通しを得た。ＳＫＹ－１上での実現手法ではスタック領域のオーバフロー等の検査をソフトウェアで行っているが、これをハードウェアで行えば性能低下をほぼ０に近付けることも可能である。

文献

山口ほか「並列処理用ＯＳ・ＳＫＹ－１　－開発構想－」
本大会論文集（１９８９年１０月）
上脇ほか「並列処理用ＯＳ・ＳＫＹ－１のスケジューリング方式」
本大会論文集（１９８９年１０月）