1Z－4

MKngプロジェクトにおける組み込みシステム技術：Machマイクロカーネル上のオンメモリ圧縮ページャの設計と実装†

石井秀浩　高野陽介　黒岩実　横田実

NEC C&C メディア研究所

1　はじめに

ディマンド・ページングをサポートしているOSの下では、プログラムは主記憶よりも大きな領域を仮想記憶として使うことができる。しかし、携帯機器、セットトップボックス、ゲーム機のようにディマンド・ページングに使える二次記憶を持たないハードウェア環境では一般に、プログラムがメモリとして使える領域の大きさは主記憶の大きさに制限されてしまう。この場合、記憶領域を多く消費するアプリケーションを実行することは困難になる。並行して実行するアプリケーションが予め定まっている場合には、オーバレイ処理などをアプリケーションが行えば実行可能ではあるが、その場合アプリケーション・プログラムの開発は難しくなり、また仮想記憶に比べて実行効率も悪くなりがちである。またオーバレイでは、複数のアプリケーションを対話的に並行して起動したり終了したりできるような環境を実現することは困難である。

そこで、ディマンド・ページングに使える二次記憶を持たない環境のために、データを圧縮しながらオンメモリでディマンド・ページングを行うシステムを、RT-Mach マイクロカーネル上に実装した。

2　ページャの実現

本ページャは、RT-Mach のデフォルト・ページャ[1]を拡張して実現した。

2.1　ページング処理

本ページャは、自タスク内の仮想記憶領域上にページング領域を設ける。ページング領域内の各ページには、そのページを使わない時には主記憶を割り当てない。使っているページには主記憶を確保し、ページアウトの対象とならないようにマイクロカーネルに申告しておく。ページング領域は、主記憶より少し少ない大きさとする。結果として、ページアウトされているページがないときにはページング領域として主記憶を消費せず、たくさんのページがページアウトされると主記憶のほとんどをページング領域として使うことになる。ページング領域内の各ページはそれぞれいくつかのブロックに分け、ブロック単位で使用する。

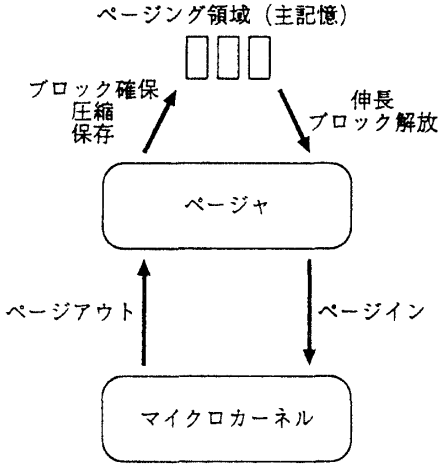

図 1: 本ページャに関するページの流れ

マイクロカーネルからページアウトの要求を受けると、本ページャはそのページの内容を圧縮し、それを収めるに必要なだけのブロックを確保して圧縮結果を保存する（図1）。後にページインの要求を受けると、対応するブロックから保存データを読んで元のデータを復元する。

2.2　ページング領域の大きさの動的調節

本ページャは、ページング領域のための主記憶を予めたくさん確保することはせず、必要な量だけを動的に確保・解放する。すなわち、ページング領域の中で確保済みで使われていない領域の量を少量の一定の範囲に保つ。余裕分がある程度少なくなると主記憶を追加確保し、余裕分が多くなると他タスクやマイクロカーネルが使えるように主記憶を解放する。

Embedding Technology of MKng Project: Design and Implementation of On-memory Compresssion Pager on a Microkernel

Hidehiro Ishii, Yosuke Takano, Minoru Kuroiwa, Minoru Yokota

C&C Research Labs, NEC

†この研究は，情報処理振興事業協会（IPA）が実施している創造的ソフトウェア育成事業「次世代マイクロカーネル研究プロジェクト」のもとに行われた．

[1]通常のページングを行うページャであり、マイクロカーネル外のタスクとして実現されている。

2.3 主記憶の割当てに関するデッドロック

Mach マイクロカーネルは、空いている主記憶が一定量より少なくなると、ページャにページアウトを要求する。

前節で述べたように本ページャは、ページング領域内の余裕分を一定範囲内に保つ。もしこの余裕分を持たないと、次のような現象によって矛盾が起こってしまう可能性があるからである。

1. 空いている主記憶が少なくなった時、マイクロカーネルがページャにページアウトを要求する。
2. ページャは、要求のページを圧縮・保存するために、保存用の主記憶の割当てをマイクロカーネルに要求する。
3. マイクロカーネルは、主記憶をページャに割り当てるために、使用中の主記憶ページを空けようとしてさらにページャにページアウトを要求する。

こうなると、ページャとマイクロカーネルの間でデッドロックが起こるか、パニックが起こる。

かと言って、余裕分を少なくしすぎると、主記憶確保の処理が頻繁に起こって効率の低下を招く。したがって本ページャは、デッドロックを起こさずかつ効率が悪くない程度の少量の余裕分の主記憶を持つ。

3 圧縮率の評価

圧縮率の評価のために、RT-Mach + Lites[2]の環境に本ページャを適用してみた。OS のブート後にテストプログラムの実行だけを行った場合、および OS のブート後に X window server, mule, kterm, xterm を立ち上げた状態で、Mach の threads ライブラリと本ページャのコンパイルを連続して行った場合の、ブート以来のページアウトに関する圧縮率の平均を計測したところ、その結果は表 1 の通りであった。テストプログラムは、仮想記憶上の 32 メガバイト分の integer (32 ビット) の配列に、順に 0, 1, 2, ... を書き込んで、次に順に読み出す、というものである (順序は FIFO にした)。表の各列はそれぞれ、ページサイズに対する圧縮後のバイト数の割合、およびページサイズに対する圧縮後のブロックの合計サイズの割合である。

表 1: 圧縮率の測定結果

	バイト単位	ブロック単位
テストプログラム	33.8%	36.9%
コンパイル	24.4%	32.2%

ここでページサイズは 4 キロバイトで、ページング領域のブロックサイズは 512 バイトである。実験機には 48 メガバイトの主記憶を搭載しており、ブート直後に空いていた主記憶の量はおよそ 22.7MB であった。

ページング領域として実際に消費する主記憶の量は、ブロック単位の方の圧縮率で決まる。ブロック単位の圧縮率が 32〜37%であるならば、計算機上の記憶領域をおよそ 3 倍程度 (圧縮率の逆数) に広げて使えるということになる。ただし、ページング領域の消費量を除いた主記憶の量がワーキングセットの大きさ以上である時は問題ないが、そうでない時はスラッシングが起こってアプリケーションの処理速度が落ちる。

また、ブロックサイズを小さくすると、ブロック単位の圧縮率が良くなる (バイト単位の圧縮率に近付く) 一方で、ブロック数が増える分、ブロックを管理するオーバヘッド大きくなる。今後、ブロックサイズを調節して、全体としての効率向上を図る予定である。

4 まとめ

以上、ディマンド・ページングに使える二次記憶を持たない機器においてディマンド・ページングを実現するメカニズムについて報告した。これによりそのような機器においても、比較的大きな記憶領域を必要とするアプリケーションを、あるいは同時に複数の任意のアプリケーションを、アプリケーションの特別の負担なく動作させることができ、小型機器におけるアプリケーションの幅が広がる。

5 関連する研究

Quarterdeck 社と Landmark Research 社の MagnaRAM シリーズ、および Connectix 社の RAM doubler は、Macintosh と Windows 系オペレーティングシステムにおいて、比較的使われなそうな主記憶領域を圧縮して主記憶に保存する。ただしいずれも、ページングに使える二次記憶のない環境にディマンド・ページングを提供するものではない。

参考文献

[1] Evangelos P. Markatos et al. “Implementation of a Reliable Remote Memory Pager”, In *Proceedings of USENIX 1996 Annual Technical Conference*, Jan. 1996

[2] Quarterdeck Corporation. “MagnaRAM 97: A White Paper”, In *http://arachnid.qdeck.com/qdeck/products/MagnaRam/mrwhite.html*, 1996

[2] Mach 用の 4.4BSD Lite Server.