1P−9

# Machにおけるデマンドページングの高速化手法
## −プリページングとクラスタリング−

数藤義明 宮本剛 岩本信一 柴山茂樹
キヤノン(株) 情報システム研究所

## 1 はじめに

カーネギメロン大学で開発されたMachオペレーティングシステムは，研究用のプラットフォームとしてだけでなく，商用のワークステーションのオペレーティングシステムとして注目されている．Machの仮想記憶に関しては，コピーオンライト等の遅延評価技法によって十分な速度が得られている[1]．しかし，ワークステーションの実用OSとしてMachを捉えた場合に，外部ページャのサポートなどのためにデマンドページングのページインの処理速度が低いことや，ページアウトが発生した場合のシステム全体の処理能力の低下は，大きな問題点である．

本稿ではデマンドページングの高速化手法について考察し，効率の高い複数ページのローディング手法のMach 2.5への実装について述べる．また，これを用いて，これらの手法の性能評価を行った結果を報告する．

## 2 デマンドページングの高速化

Machのページング動作において高速化を考えた場合に，以下のような方法が考えられる．

- ページャとの通信にIPCを用いずに関数コールで行う．
- ページングの単位を複数のページとする．

ページャとの通信にIPCを用いずに関数コールで行うと外部ページャの使用が不可能になり，Machの機能を限定してしまう．次のページングの単位を複数のページとする方法は，VMモジュール内のページ単位(仮想ページ)をハードウェアが使用するページ単位(物理ページ)の複数倍にすることによって簡単に実現することができる．ただし，この方法ではすべてのメモリの使用が仮想ページ単位となり，主記憶の使用効率を著しく低下してしまう[2]．そこで次章に述べるようなページイン時にだけ複数ページを一度にローディングするプリページング方式とクラスタリング方式によりページング動作の高速化を図る．

## 3 プリページングとクラスタリング

デマンドページングのページイン時に，複数のページを一度にローディングする方法として，プリページング方式とクラスタリング方式が提案されている[3,4]．プリページング方式は，デマンドページングのページイン時に要求されたページだけでなく，それ以降のページも同時にローディングする方法である．また，クラスタリング方式は，実行ファイルを数ページ単位のページクラスタと呼ばれるブロックに分割して，ページインの単位をページクラスタとする方法である．これらの二つの方法はどちらもページイン時にページ単位でローディングするのに比較して，一度に複数ページをローディングしてくる場合のほうが1ページあたりのローディング時間を短くできることを利用している(表1)．Machの場合には，IPCによるページャとの通信回数が少なくなり，かなりの効果が得られると思われる．

表1: 複数ページのローディング処理時間

| ページ数 | 処理時間(ms) | 1ページあたりの処理時間(ms) |
|---|---|---|
| 1 | 25.3 | 25.3 |
| 2 | 36.8 | 18.4 |
| 4 | 66.6 | 16.6 |
| 8 | 120.6 | 15.1 |

Mach 2.5のVMモジュールに実装する場合に次のような点に注意が必要となる．まずMachがマルチプロセッサ対応であることから，他のプロセッサが先読み中のページに対してページインの要求を出さないように，fictitiousページを用いなければならない．また，外部ページャが複数のページ単位での要求をサポートしていない場合や，外部ページャが分散共有メモリを実現するものであった場合に，複数ページ単位での要求が余分なネットワークトラフィックを引き起こす可能性がある．従って，外部ページャに要求を出す場合にはページ単位でデマンドページングを行う．また，プリページングやクラスタリングによって先読みされたページが利用されない限り，そのページがページ置換の対象として優先的に選択されるようにinactive queueに入れておくことが必要であろう．

## 4 実験

前章で述べたようなプリページング方式とクラスタリング方式をMach 2.5のVMモジュールに組み込んで性能を評価した．計測には当研究所で試作したワークステーションStonehigh(ページサイズ8KB)を用いた[5,6]．図1にプリページング方式を実装し，gccによるsedのコンパイル時間を計測した結果を示す．さらに図2にクラスタリング方式を実装し，同じ計測を行った結果を示す．これらの図から分かるように，両方式とも単純なデマンドページングを行った場合に比較して3〜5%の実行時間を低減できた．また両方式の相違は，実行時間に関してはほとんどない．一度にページインするページ数の変化は，両方式ともほとんど影響がないということが分かった．これは，表1に示すように一度にページインするページ数に対して1ページのローディングの処理時間を計測した結果にも現れている．1ページの場合

---

High Speed Demand Paging Mechanisms for Mach
− Pre-Paging and Clustering −
Yoshiaki Sudo, Tsuyoshi Miyamoto, Shinich Iwamoto and Shigeki Shibayama
Information Systems Research Center, Canon Inc.

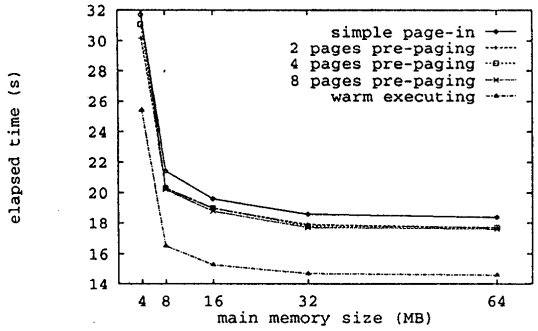


図 1: プリページング方式のコンパイラ実行時間

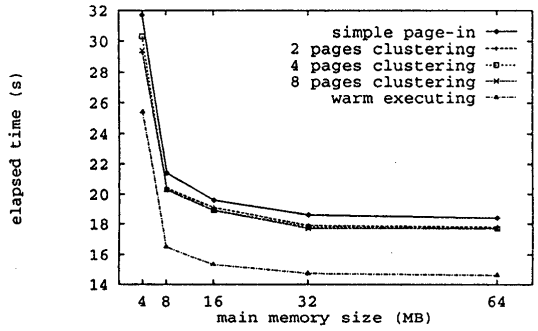


図 2: クラスタリング方式のコンパイラ実行時間

と比較して複数ページでローディングしてくる場合の方が1ページあたりの処理時間が小さくて済むが，2〜8 ページの場合に変化はほとんどない．以上から，要求されたページの後のページを先読みすることと，周囲のページを先読みすることの違いは重要ではなく，一度に複数のページをローディングすることが重要であることが分かった．

表2には，プリページングを実装して上記のように gcc によるコンパイルを行った場合に，コンパイラのページ (総数 64 ページ) で最終的にローディングしたページ数を示す．この表からコンパイラのように割合に大きな実行ファイルに関しても，実行時にほとんどのページは要求され，プリページング方式を実装した場合でも余分なページは数ページしか要求されていないことが分かる．

さらに同時に複数のプロセスが動作している場合にプリページングが与える影響について計測した．表3は emacs の起動と gcc によるコンパイルを同時に行った場合の両方の実行時間を示したものである．これによると，フリーページが多く存在する場合には，どちらの実行時間も短縮される．フリーページが少ない場合には，無駄なページインを行なう場合もかなり存在し，プリページングを行なったことによる利点が失われてしまう．しかし，単純にページ単位でローディングする場合よりも実行時間が延びることはなかった．

表 2: プリページング方式のプリページ数とローディングしたページ数

| プリページ数 | ローディングしたページ数 |
|---|---|
| 1 | 62 |
| 2 | 63 |
| 4 | 64 |
| 8 | 64 |

表 3: 複数プロセス実行時のプリページの影響

| プリページ数 | | 実行時間 (s) 主記憶 4MB | 実行時間 (s) 主記憶 64MB |
|---|---|---|---|
| 1 | emacs | 14.9 | 8.3 |
| | gcc | 32.4 | 23.6 |
| 2 | emacs | 14.9 | 7.0 |
| | gcc | 32.2 | 22.0 |
| 4 | emacs | 14.8 | 7.1 |
| | gcc | 32.2 | 21.4 |

## 5 おわりに

本稿では Mach のデマンドページングの高速化手法として，プリページング手法とクラスタリング手法の実装について考察し，それらの手法を実際に Mach 2.5 の VM モジュールに実装して性能評価を行った．今回の実験によると2ページ以上のページ単位でローディングした場合には単純にページ単位でローディングした場合よりも 3〜5%の実行時間を低減できた．プリページングとクラスタリングという2つの手法についての実行時間の違いはなかった．また複数プロセスの動作中では，フリーページが十分な場合にはプリページングの効果が現れるが，フリーページが少ない時にはプリページングを行なうことの利点とページフォールトの欠点が相殺し，実行時間の向上は見られなくなった．

今後の課題としては，NFS などのリモートファイルに対して，これらの手法がどれぐらい有効であるかの評価や，ページアウト動作における高速化手法についての研究をする必要がある．

### 参考文献

1) Tevanian, A., "Architecture-Independent Virtual Memory Management for Parallel and Distributed Environments, The Mach Approach," PhD thesis, CMU (1987).
2) 数藤, 宮本, 岩本, 柴山, "Mach における仮想記憶のページサイズによる影響," 情報処理学会第 44 回全国大会 (1992).
3) Trividi, K. S., "Prepaging and Application to Array Algorithms," *IEEE Trans. Comput.*, Vol C-25, pp. 915-926 (1976).
4) Black, D., Carter, J., Feinberg, G., MacDonald, R., Mangalat, S., Shienbrood, E., Sciver, J. V. and Wang, P., "OSF/1 Virtual Memory Improvements," in *Proc. Mach Symposium*, pp. 87-103 (1991).
5) 伊達, 濱口, 出井, 柴山, "マルチプロセッサワークステーション "Stonehigh" –コンセプトとハードウェア概要–," 情報処理学会第 45 回全国大会 (1992).
6) 鈴木, 宮本, 伊達, 岩本, 柴山, "マルチプロセッサワークステーション "Stonehigh" –機能分散型 OS の設計と実現–," 情報処理学会第 45 回全国大会 (1992).