複数のページサイズを扱う仮想記憶管理機構の実装と評価

5F−8

伊藤 修一　光澤 敦

NTT 情報通信研究所

1 はじめに

近年のプログラムは大規模化かつ複雑化してきているため，プログラムの実行時間に占めるアドレス変換の時間の割合が増大している [1]．プログラムの実行速度を向上させるためには，このアドレス変換の時間を削減することが重要である．

アドレス変換の時間は TLB（*translation lookaside buffer*）のヒット率を向上させることで削減できる．ヒット率を向上させる方法の一つは TLB のエントリ数を増やすことである．しかしこの方法はコストが高い．もう一つの方法としては一つのエントリでマップできる領域を大きくする方法がある．今日ではその方法を一歩進めて，エントリごとにマップできる領域の大きさを変える方法が考え出されている．この方法はスーパーページと呼ばれ，最近のプロセッサではこの機能を実装しているものが多い．

我々は文献 [2] でスーパーページを有効に利用するためには仮想記憶で複数のサイズのページを扱える必要があることを述べた．さらにそのための機構として，複数のサイズの未使用ページフレームを扱う機構と複数のサイズの仮想ページを管理することのできるページ表について提案した．

本研究では 4KB と 16KB の 2 種類のページサイズを扱えるページフレーム管理とページ表管理を MIPS R4400 プロセッサ [3] 上で動作する Mach 3.0 オペレーティングシステム（OS）[4] へ実装し，評価する．本稿では，まず R4400 の TLB アーキテクチャについて第 2節で概観し，Mach 3.0 での仮想記憶管理機構の設計について第 3節で述べる．そして実装と評価について第 4節で述べ，最後に第 5節で本稿をまとめる．

2 MIPS R4400 TLB アーキテクチャ

R4400 には 48 個の TLB エントリがある．各エントリのフォーマットを図 1に示す．エントリのサイズは 128 ビットであり，マップする領域の大きさを指定するフィールド（MASK），仮想ページ番号（VPN2），物理ページフレーム番号（PFN）などにより構成される．スーパーページの機能は MASK フィールドで実現される．指定できるサイズは最小のページサイズであるベースページサイズの 2 のべき乗であり，4KB（ベースページサイズ），16KB, 64KB, 256KB, 1MB, 4MB, および 16MB である．

ここで PFN フィールドは 2 つあるが，これはサブプロッキングと呼ばれ，R4400 では 1 つのエントリで 2 つのページをマップできる．例えば，MASK 値に 4KB が指定された場合，1 つのエントリで 8KB の領域をマップできる．

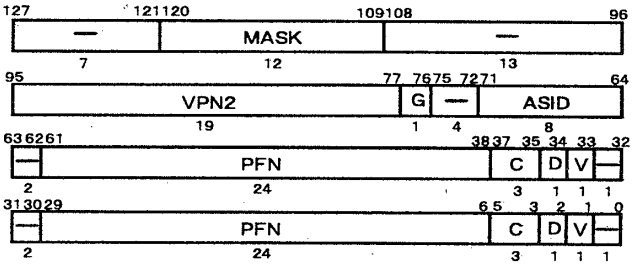

図 1: MIPS R4400 の TLB エントリ

3 設計

本節ではまず Mach 3.0 の仮想記憶管理機構について概観する．その後で仮想記憶管理の設計について述べる．

図 2に Mach 3.0 の仮想記憶管理機構を示す．Mach 3.0 の仮想記憶管理機構は仮想空間とページフレームを管理する VM 層と，ページ表と TLB の管理，そして TLB に関する例外を処理する PMAP 層に分かれる．

VM 層はタスクからのメモリ割当 / 解放要求に基づき，仮想空間の領域を割当て，その領域をエントリとして管理する．またその領域に対してフリーリストから / へ未使用ページフレームを取得 / 解放する．PMAP 層はページ表エントリを作成 / 削除し，TLB エントリを書込 / 消去する．また TLB 変更例外や TLB ミス例外などの TLB に関する例外が起きた場合も同様にページ表エントリおよび TLB エントリを処理する．

VM 層では，メモリの割当 / 解放を要求する関数として `vm_allocate` と `vm_deallocate` がある．ページフレームを取得 / 解放する関数として `vm_page_grab` と `vm_page_release` がある．PMAP 層では，ページ表エントリを作成 / 削除する関数として `pmap_enter` と `pmap_remove` がある．TLB エントリを書込 / 消去する関数として `tlb_map_random` がある．また TLB 変更例外を処理する関数として `tlb_modify` があり，TLB ミス例外を処理する関数として `tlb_refill` がある．

Mach 3.0 の仮想記憶管理機構で 4KB と 16KB のページサイズを扱えるようにするには，VM 層と PMAP 層を変更する必要がある．VM 層ではページフレーム管理を変更

Multiple Sized Pages in Virtual Memory System
Shuichi ITO and Atsushi MITSUZAWA
NTT Information and Communication Systems Laboratories

する．ページフレーム管理では，利用可能な物理メモリの一定の領域を4KBで分割し，残りを16KBで分割する．そしてサイズ4KBと16KBの未使用なページフレームをそれぞれのフリーリストで管理する．サイズ16KBのページフレームはメモリ割当要求がユーザ空間からの場合に取得され，サイズ4KBのページフレームはカーネル空間からの場合に取得される．Mach 3.0と同じく，カーネル空間でサイズ4KBのメモリを割当てた理由は，ページ表で使うメモリを大きくしたくなかったからである．

PMAP層ではページ表管理を変更する．ページ表管理では，16KBのスーパーページのページ表エントリを扱うために，ページ表でエントリを複製する．これによってサイズ16KBのページをマップするページ表エントリの探索はサイズ4KBのそれと変わらない時間でできる．

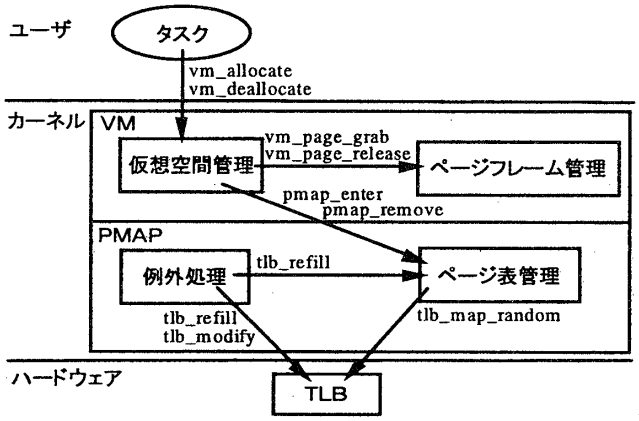

図 2: Mach 3.0の仮想記憶管理機構

4 実装と評価

本節ではR4400上で動作するMach 3.0への実装について述べる．変更した関数を表1に示す．ただしページフォールトを処理する関数はコーディング中であり，ここでは示さない．以下，変更した関数について説明する．

VM層では`vm_allocate`, `vm_page_grab`, そして`vm_page_release`を変更した．`vm_allocate`を変更した理由は，割当てるメモリの大きさがユーザ空間からの場合，メモリの割当サイズに対して16KBのアライメントを取り，カーネル空間からの場合，4KBのアライメントを取らなければいけないからである．VM層で変更した行数はC言語で300行程度であった．

PMAP層のページ表管理では`pmap_enter`, `pmap_remove`, そして`tlb_map_random`を変更した．`tlb_map_random`を変更した理由は，Mach 3.0ではTLBを使用する前に一度ページマスクレジスタをセットするだけで良かったが，複数のページサイズを用いる場合，TLBの書込時にページサイズをページマスクレジスタにセットしなければいけないからである．

PMAP層の例外処理では`tlb_modify`と`tlb_refill`を変更した．`tlb_modify`を変更した理由は，サブブロッキング機能を用いているため，サイズ16KBの仮想ページの番号を見て，2つのPFNフィールドのどちらにページフレーム番号を書込むかを判断しなければいけないからである．`tlb_refill`を変更した理由は，`tlb_map_random`の変更理由と`tlb_modify`の変更理由の両方である．PMAP層で変更した行数はC言語で600行，アセンブリ言語で90行程度であった．

現在実装中であり，変更した関数の行数の合計は1000行程度であった．評価として，TLBのヒット率，物理ページフレームの割当時間，そしてアプリケーションの実行時間を測定する予定である．

表 1: 変更した関数とその行数

層 / 機構	変更した関数	言語	行数

VM	`vm_allocate`
	`vm_page_grab`
	`vm_page_release`
		C	300

PMAP	`pmap_enter` `pmap_remove`	C	600

	`tlb_map_random`
	`tlb_modify`
	`tlb_refill`
		asm	90

5 まとめ

本研究では，アドレス変換の時間が増大しているという問題を解決するために，TLBのスーパーページ機能を利用した．そしてスーパーページ機能の有効利用のために，4KBと16KBのページサイズを扱う仮想記憶管理機構を設計し，R4400上で動作するMach 3.0に実装中である．変更した関数の行数の合計は1000行程度であった．

今後の課題は第一に使用するページサイズの種類を増やすことと，第二に動的に物理ページフレームを管理することである．

参考文献

[1] Mendel Rosenblum, Edouard Bugnion, Stephen A. Herrod, Emmett Witchel, and Anoop Gupta. The Impact of Architectural Trends on Operating System Performance. In *Proc.15th ACM Symposium on Operating System Principles*, December 1995.

[2] 伊藤修一, 光澤敦. 可変ページを用いた仮想記憶管理に関する研究. 第57回情処全大論文集 (1), pp. 97–98, October 1998.

[3] Gerry Kane and Joe Heinrich. *MIPS RISC Architecture*. Prentice Hall, 1992. ISBN 0-13-590472-2.

[4] Uresh Vahalia. *UNIX Internals*. Prentice Hall, 1996. ISBN 0-13-101908-2.