IA32 版 Linux Super Page の実現と評価

早坂晴康[†] 清水尚彦[††]

よく知られているように，今日のプロセッサとメモリの性能格差を埋めるためにレイテンシ隠蔽技術が開発されてきたが，メモリを多く使用するアプリケーションでは TLB ミスもまたパフォーマンスに大きく影響する．近年のプロセッサは，複数のページサイズで TLB をサポートし，TLB の有効範囲を拡張することが可能であるがほとんどのオペレーティングシステムは，1 つのページサイズしかサポートしていない．IA32 でも 4 KB ページ以外に 4 MB ページをサポートする．本稿では，ユーザから不可視な形で複数のページサイズをサポートする IA32 版 Linux Super Page の実装と性能を報告する．

An Implementation and Evaluation of IA32 Linux Super Page

Haruyasu Hayasaka[†] and Naohiko Shimizu[††]

It is very important for the application and/or the operation system to avoid TLB misses as much as possible. Many processeors have page size extension feature that extend the coverage of TLB significantly, but few operating sytems support it. IA32 is also supporting 4 MB page size as well as 4 KB page size. In this paper, we will present our implementation of transparent Super Page Kernel for IA32 Linux and performance.

1. はじめに

アプリケーションが高速に実行されないのには多くの理由がある．深刻な性能のボトルネックは，プロセッサのバッファ構造にあり，それはよく知られているようにメモリの問題である．1 つはキャッシュメモリであり，避けられないキャッシュミスのために遅延となる．近年，メモリの遅延はよりいっそう深刻で，メモリのレイテンシ耐性のために高性能プロセッサではヒットアンダーミスあるいは投機実行などの技術が用いられている．しかし同じく TLB ミスも，多くのメモリを使用するアプリケーションで大きな障害になっている．

そこで，近年のプロセッサは TLB の有効範囲を拡張するために複数のページサイズをサポートしている．しかし，ほとんどのオペレーティングシステムでは 1 つのページサイズしかサポートしていない．今回，我々は IA32 プロセッサの Linux Kernel にユーザプログラムから不可視な形で複数のページサイズをサポートする Super Page の実装を行った．

† 東海大学大学院工学研究科
Graduate School of Engineering, Tokai University
†† 東海大学電子情報学部コミュニケーション工学科
School of Information Technology and Electronics, Tokai University

本稿では，IA32 版 Linux Super Page の実装と性能評価を示す．以下，2 章で関連技術を紹介し，3 章で Linux の仮想メモリ管理を説明し，4 章で IA32 版 Super Page Kernel の設計を説明する．5 章で Super Page Kernel の性能評価を示し，6 章で Super Page Kernel のために追加したカーネルインタフェースについて説明し，7 章でまとめる．

2. 関連技術

IRIX [1)] と HP-UNIX [2)] がユーザプログラムから不可視な複数のページサイズをサポートしているが，両者ともユーザが環境変数もしくはコマンドによりページサイズを指定しなければならない．加えて，1 プロセスに 1 つのページサイズしか指定できないので，ページサイズをあまり大きくすると余分なワーキングセットを与えてしまう可能性がある．我々の Super Page Kernel では要求されたワーキングセットに対して動的にページサイズを割り当てるので余分なワーキングセットを与えることはない．

また，NSW のサモンらが Huge TLB Page として，拡張ページを Linux に組み込んでいるが，特別なシステムコールを要する上に Huge TLB Page 領域のスワップを禁止するなど実用上問題点が多い．

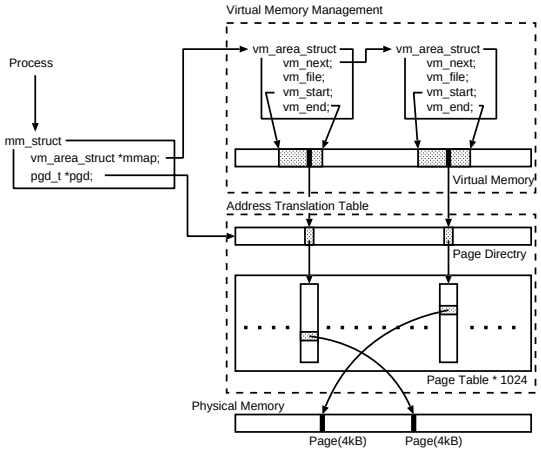


図 1 Linux のメモリ管理の構造モデル
Fig. 1 Linux's memory management.

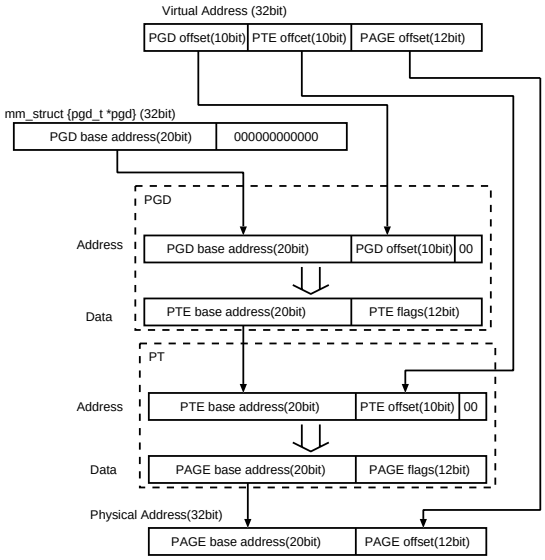


図 2 4 K ページのアドレス変換
Fig. 2 Address translation for 4 K page size.

3. Linux におけるメモリ管理

3.1 仮想メモリ管理のための単位（ページ）

仮想メモリ管理ではメモリをページという単位で管理している．これは，実装されている物理メモリをプロセッサごとに決まっている容量で区分けし，1 つ 1 つの独立した領域として扱う構造である．この区分けされた領域をページと呼び，ページを単位としてメモリの確保，解放などの管理を行っている．IA32 の場合 4 KB であり，またこれをベースページという．

3.2 mm_struct 構造体

mm_struct はプロセスに与えられた仮想メモリ空間を管理する構造体である．Linux ではすべてのユーザプロセスがそれぞれ独立した仮想メモリ空間を管理している．この管理を行うために使用されるのが mm_struct 構造体であり，ユーザプロセスは必ず 1 つの mm_struct 構造体を持つことで，1 つの仮想メモリ空間を操作することができる．ユーザプロセスが要求したメモリ領域は 1 つ 1 つがそれぞれ vm_area_struct 構造体で管理され，リスト化されており，メンバ変数の mmap がそのリストの先頭を指している．また，pgd はアドレス変換において最初の参照先となるページディレクトリのアドレスを格納しており，仮想アドレスを物理アドレスに変換するときに使用される．データ構造のモデルを**図 1** に示す．

3.3 vm_area_struct 構造体

vm_area_struct は仮想メモリ空間上で確保された個々のメモリとその保護情報を表す構造体である．プロセス上でメモリ領域を要求すると，この構造体が生成され管理情報として使用される．vm_start と vm_end がそれぞれ，確保された領域の始点と終点の仮想アドレスを表す．1 つのプロセスが持つすべての vm_area_struct 構造体はアドレスの順にリンクされたリスト構造になっており，メンバ変数の vm_next が次の vm_area_struct 構造体のポインタを保持している．

3.4 アドレス変換とアドレス変換テーブル

Linux では PGD（ページグローバルディレクトリ），PMD（ページミドルディレクトリ），PT（ページテーブル）の 3 段階の変換テーブルを参照して，仮想アドレスから物理アドレスへと変換を行う．これは 64 ビットプロセッサのメモリ管理機構にあわせたものであり，IA32 アーキテクチャでは物理アドレス拡張をしない場合 PGD，PT の 2 段階のアドレス変換となっている．そのため，PGD と PT の間に仮想的な 1 つの PMD を置くことで，アーキテクチャの相違を吸収し，3 段階の変換を行っているように見せかけている．**図 2** に IA32 でのアドレス変換を示す．

3.4.1 拡張ページ（ラージページ）

Intel 社の Pentium Pro モデル以降のプロセッサにはページサイズの拡張が導入されている．これは，4 KB ページの他に 4 MB ページをサポートするものである．拡張された 4 MB ページでは上位 10 ビットを物理アドレス，残り 22 ビットをオフセットとし PT を使用せずに 1 つの PGD だけで管理する．PT を省略し，PGD が直接 4 MB の先頭アドレスを指すことでページサイズの拡張を行うため，4 MB ページ使用時には物理メモリ，仮想メモリともに先頭アドレスを

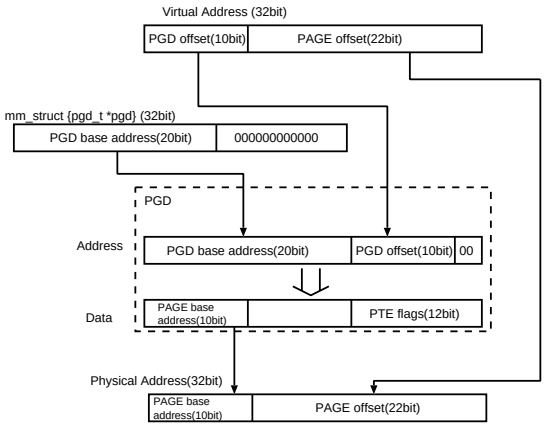


図 3 4 MB ページのアドレス変換
Fig. 3 Address translation for 4 M page size.

表 1 評価で使用したプロセッサの TLB エントリ数
Table 1 Number of TLB entries of processor used by evaluations.

	4 KB ページ用	4 MB ページ用
Celeron 900 MHz	32	8
Pentium4 1.6 GHz	64	

4 MB 境界に合わせる必要がある．**図 3** に 4 MB ページのアドレス変換を示す．

3.4.2 TLB

初めてメモリ中にあるアドレス変換テーブルを使用し，対応する物理アドレスが算出されると，このアドレス変換の結果は TLB に蓄えられる．TLB にバッファされている間の変換は，仮想アドレスから直接物理アドレスへと変換され高速に変換される．**表 1** に今回評価で使用したプロセッサの TLB のエントリ数を示す．

表からも分かるように Celeron では 4 KB ページ用の TLB と 4 MB ページ用の TLB が別々に用意され，Pentium4 の TLB は共用となっている．

3.5 Buddy System による物理メモリの管理

Linux の物理メモリ管理方式は Buddy System と呼ばれている．これは物理的に連続するページを複数のグループに分けて管理する方式である．

Linux の Buddy System では 2 のべき乗単位で 2^0 ページから 2^9 ページまでの物理的に連続した未割当てのページを，双方向循環リストで管理する．**図 4** に Buddy System の例を示す．実際には，連続した 1 つ 1 つのページの状態を管理している mem_map 配列の先頭アドレスをリスト化している．また，メモリは 3 つ（MAX_NR_ZONE = 3）の領域に分けられている．

もし，16 ページ分の連続した空き容量が要求されたときは，alloc_pages()，もしくは __get_free_pages() 関数でオーダーを 4（$2^4 = 16$）として呼び出す．このとき，オーダー 4 のグループに未割当てのページが有る場合はその先頭のページのアドレスを返し，無い場合はオーダー 5 のグループから連続した 32 ページを 2 つの連続した 16 ページに分割して片方のアドレスを返す．オーダー 5 のグループにも未割当てのページがなければ順に上のグループへ探しにいく構造となっている．

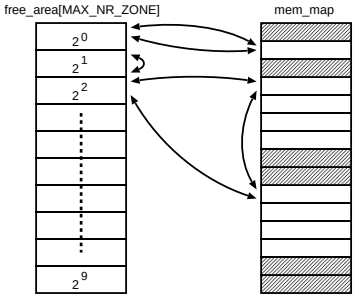


図 4 Buddy System のデータ構造
Fig. 4 Buddy memory allocator.

4. IA32 版 Super Page Kernel 設計[3),4)]

4.1 設計方針

我々は，シンプルで実用的な Super Page Kernel の設計を行いたいと考えた．そこで，以下のような設計方針を基礎とすることにした．

バイナリ互換性 日常使用するアプリケーションや，他の環境でコンパイルしたアプリケーションも変更なしにラージページを使用できるように特別なシステムコールなどを用いない．

ワーキングセット IRIX[1)]や HP-UNIX[2)]でも複数のページサイズをサポートし，複数のラージページが使用できるが，1 プロセスに対し 1 つのページサイズしか指定できない．そのために，余分なワーキングセットが出てしまう場合があり，またユーザがページサイズの指定をしなくてはならない．それに対し我々の提案では，プログラムのリクエストサイズ分の仮想アドレス空間だけを割り当て，4 MB 境界にある場合にのみ 4 MB ページの割当てを行うので余分なワーキングセットは出ない．また，その際の割当ては自動的に行うため，ユーザは 4 MB ページの割当てを示す必要はない．

allocate at the access 4 MB ページでもアプリケーションが実際に使用しないページを割り当てない “allocate at the access” ポリシーを守る．

ダウングレード オペレーティングシステムが 4 MB

ページの一部のページで属性を変更する可能性があるため，オペレーティングシステムによる 4 MB ページから 4 KB ページへの動的なダウングレードを実装する．

4.2 IA32 版 Super Page 実装

IA32 プロセッサでは CR4 レジスタの PSE フラグをオンにすることで 4 KB と 4 MB ページの混在が有効となる☆．このとき，PGD エントリのページサイズ（PS）フラグをオンにすることによって，プロセッサは PGD エントリの上位 10 ビットを 4 MB ページの物理アドレスとして扱う．

4.2.1 Buddy System の変更

上記したように Buddy System は連続した 512 ページ分（オーダー 9：4KB×2^9 = 2MB）のリストまでしか管理されていない．そこで，4 MB ページのために MAX_ORDER を 10 とし，連続した 1024 ページまで管理するように変更した．

4.2.2 シャドーページディレクトリ

IA32 版 Linux Super Page では，PGD エントリから物理アドレスへ直接参照することによって 4 MB ページの物理アドレスを扱う．しかし，Linux のメモリ管理上 PT が必要な場合がある．そこで，我々は PGD を対に持つことによってこの問題を解決した．元からある PGD はソフトウェアが使用し，もう 1 つの PGD をシャドーページディレクトリと呼び，TLB が使用するように実装した．シャドーページディレクトリはプロセスに 1 つだけでありメモリ使用量の増加はわずかである．

ユーザプロセスが mmap などのメモリ要求を行うときに，仮想アドレスの範囲のうち 4 MB 境界に重なっている部分にあたる 1024 すべての PT エントリの Super Page 予約フラグをオンにする（これを Super Page 予約と呼ぶ）．Super Page 予約ビットは PT エントリのページフラグにあるシステムソフトウェアのために予約されている場所の空きビットに実装した．**図 5** に Super Page 予約を示す．Super Page 予約ビットを立てたときにはまだ実ページの割当ては行われない．

PT エントリの Super Page 予約フラグの立ったアドレスにアクセスがあればページフォルトが発生し，このときの do_anonymous_page() 関数内の物理アドレス確保の段階で Buddy System から 4 MB の連続したメモリを要求する．4 MB の連続したメモリが確保できれば，そのアドレスをシャドーページディレクトリに 4 MB ページの割当てを行い（これを Super Page 割当てと呼ぶ），シャドーページディレクトリにのみ PS フラグを立てる．ソフトウェアが使用する PGD には PS フラグを立てず，PT に 4 MB ページを 1024 エントリに分け 4 KB ページを割り当てる．これによって，ソフトウェアが使用するページ変換テーブルは，従来の Linux と互換性を保つことになる．Buddy System からページの確保が失敗した場合（これを Super Page 割当て失敗と呼ぶ）には，ページをダウングレードした上で 4 KB ページの割当ての再試行を行う．

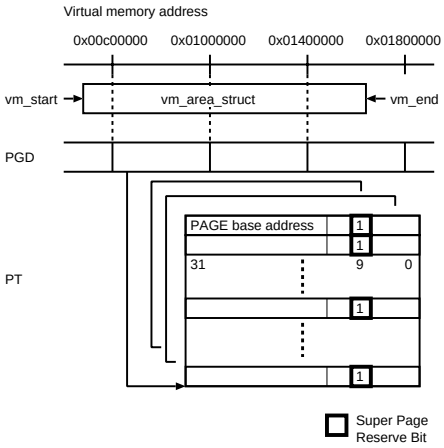
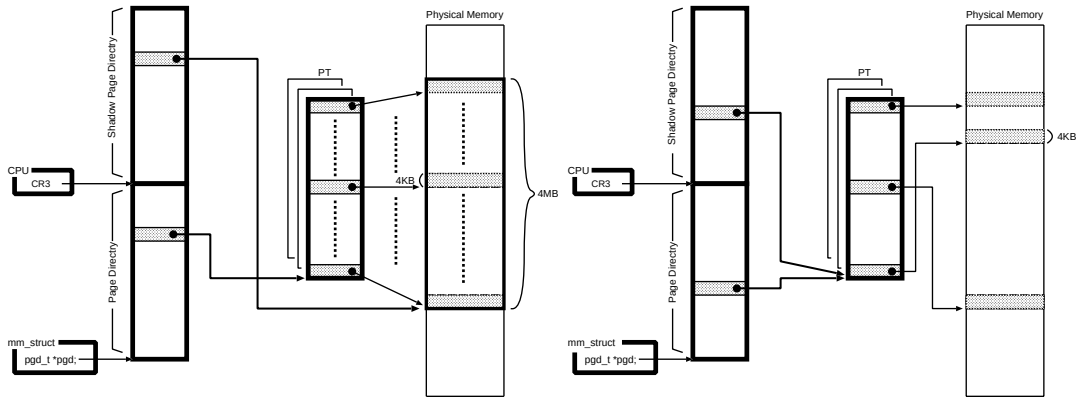


図 5 Super Page 予約
Fig. 5 Super Page reservation.

シャドーページディレクトリの実装には，まず通常の 2 倍の PGD を確保する．TLB は CR3 レジスタを参照し，ページディレクトリの物理アドレスを得ているので TLB が PGD の物理アドレスを得るときに，PGD 内でのエントリ数である PTRS_PER_PGD を加えることで，シャドーページディレクトリの物理アドレスを得ることができる．**図 6** に 4 MB ページの場合と通常の 4 KB ページの場合のシャドーページディレクトリを用いたアドレス変換例を示す．図 6 (a) は 4 MB ページを割り当てた場合の図である．シャドーページディレクトリのエントリは直接 4 MB ページ境界の物理アドレスを指している．一方，OS が参照する mm_struct 構造体によって示されるページディレクトリの対応するエントリは 1,024 個の連続する 4 KB ページを指示するページテーブル PT を指しており，ソフトウェアでのページ巡回が可能である．図 6 (b) は 4 KB ページを割り当てた場合の図である．この場合，シャドーページディレクトリのエントリとページディ

☆ Linux は CPU フィーチャに PSE ビットが立っている場合に PSE フラグをセットする．一部の AMD プロセッサは PSE ビットは 0 であり代わりに PSE36 ビットが 1 となるため，我々は，Linux Kernel に AMD プロセッサを対応する修正を加えている．



(a) Super Page の場合 (b) 通常の（4 KB）ページの場合

図 6 シャドーページディレクトリを用いたアドレス変換
Fig. 6 Address translation using Shadow Page Directory.

レクトリのエントリは同一の PT を指示して，4 KB 単位のアドレス変換を行う．

4.2.3 ダウングレード

4 MB ページとして割り当てられたページのスワップ発生による一部分のページの解放や，PT を必要とするアドレス変換テーブルの管理作業による 4 KB ページ単位の作業を行うために，ページサイズのダウングレードを実装した．ダウングレードは各ページテーブルエントリの PS フラグをクリアした後，ソフトウェアが使用しているページディレクトリの情報をシャドーページディレクトリにコピーすることによって行う．ダウングレードした 4 MB ページの TLB はフラッシュする．

4.2.4 4 MB ページアライン

一般にユーザのメモリ要求は Super Page 境界にならない．また要求メモリ量も Super Page の倍数にはならないため，シャドーページディレクトリでは前後に 4 MB に割り付けできないページが発生する．この様子を**図 7** に示す．

そこで，なるべく 4 MB ページが使用されやすくするように，anonymous mmap によるメモリ要求の仮想アドレスを 4 MB ページにアラインするオプションを実装した．

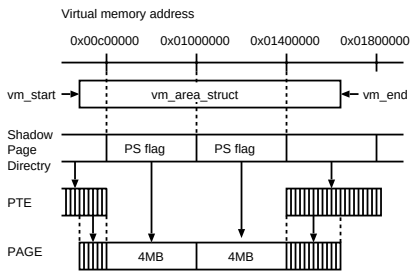


図 7 4 MB ページと 4 KB ページの混在したシャドーページディレクトリ
Fig. 7 Shadow Page Directory with 4 M and 4 K page sizes.

```
for(k=0; k<ITR; k++)
    for(i=0; i<dim; i++) {
        for(j=0; j<dim; j++) {
            a[i][j] = b[j][i];
        }
    }
```

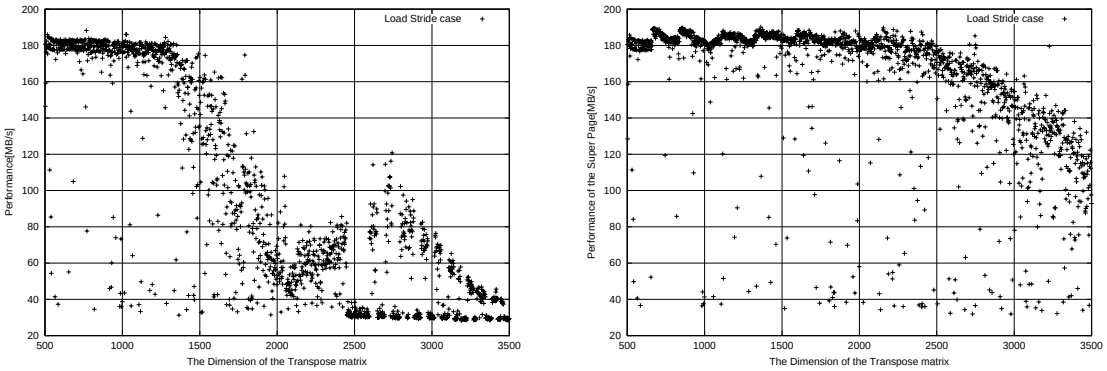
図 8 行列転置ベンチマークプログラム
Fig. 8 Matrix transposition benchmark program.

5. 性能評価

5.1 行列転置ベンチマーク

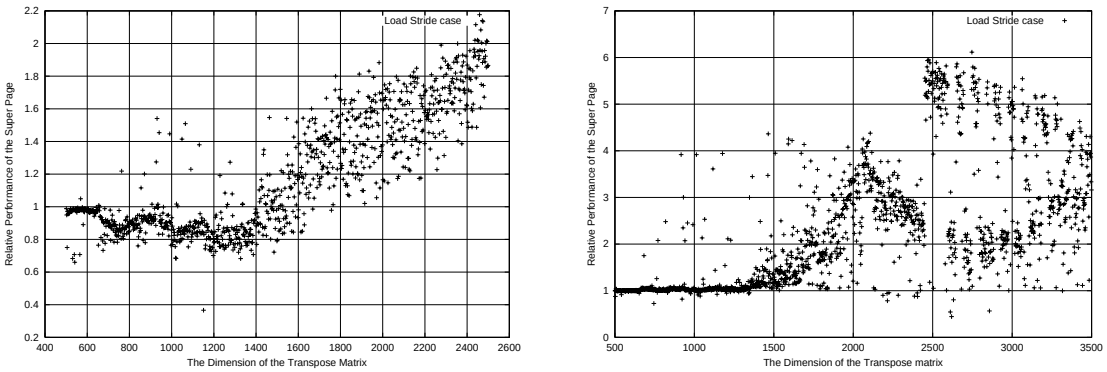
行列の転置を行うプログラムで，通常の Kernel，Super Page Kernel のメモリ性能の比較を行った．**図 8** が行列転置ベンチマークプログラムのロードストライドのコードである．

ベンチマークは *Dimension* × *Dimension* の正方行列 a，b の転置複製を行っている．この場合，i が十分に大きければ常に違う 4 KB ページのロードを行うことになる．i, j を変えることによってストアストライドのコードとなる．今回，Celeron 900 MHz/256 MB と Pentium4 1.6 GHz/1 GB でこのベンチマークの実行測定を行った．プログラムは gcc コンパイラで “-funroll-loops” オプションを付けてコンパイルを行った．できるだけプロセッサのメモリパファーマンスのみをテストするために，このコンパイラオプションを



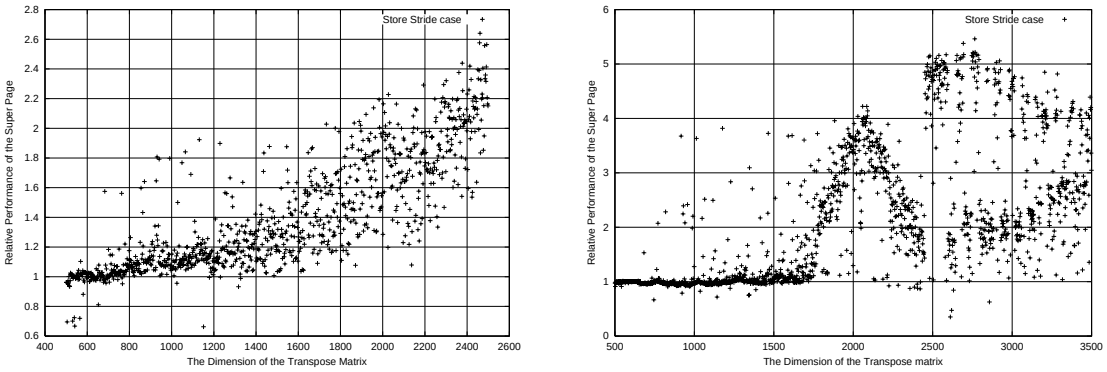
(a) 通常の Kernel (b) Super Page Kernel

図 9 Pentium4 のロードストライド転送性能
Fig. 9 Load stride transmission performance of Pentium4.



(a) Celeron (b) Pentium4

図 10 通常の Kernel 対 Super Page Kernel のロードストライド転送性能比
Fig. 10 Load stride transmission performance ratio Super Page Kernel vs. Normal Kernel.



(a) Celeron (b) Pentium4

図 11 通常の Kernel 対 Super Page Kernel のストアストライド転送性能比
Fig. 11 Store stride transmission performance ratio Super Page Kernel vs. Normal Kernel.

指定した．

図 9 は，Pentium4 プロセッサの通常 Kernel と Super Page Kernel のロードストライド転送性能を示している．

図 10 は，Celeron プロセッサと Pentium4 プロセッサでの通常の Kernel 対 Super Page Kernel のロードストライド転送性能比を示している．Celeron プロセッサでは，Super Page Kernel は約 2 倍ほどの性能が

表 2 SPEC ベンチマークを行ったシステム構成
Table 2 Evaluate system for SPEC benchmarks.

CPU	Intel Pentium4 1.6 GHz
Memory	DDR SDRAM PC2100 1 GB
ChipSet	SiS 650
OS	Linux 2.4.19
Compiler	gcc-2.95.3

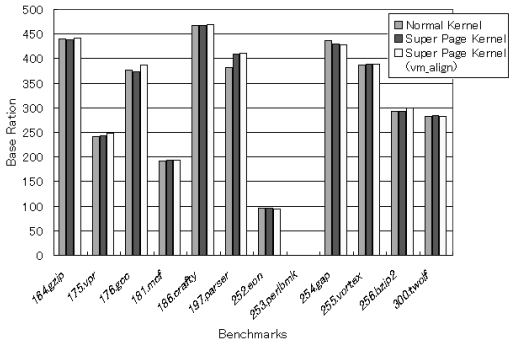


図 12 IA32 での SPEC CINT2000 の結果
Fig. 12 Result of SPEC CINT2000 in IA32.

出ている．しかし，一部性能の低下もみられる．Pentium4 では，Celeron プロセッサより性能の改善が良く，最大 5〜6 倍の性能が出ている．これは，4 MB ページがマッピングできる TLB の数が Celeron では 8 であるが，Pentium4 では 64 と多いのが原因の 1 つとして考えられる．

図 11 は通常の Kernel 対 Super Page Kernel のストアストライド転送性能比である．ロードストライドと同じような性能比となった．

5.2 SPEC CPU2000 [8)]

通常の Kernel，Super Page Kernel と 4 MB 境界にアラインした Super Page Kernel（vm_align）との性能を SPEC CPU2000 で測定した．ハードウェアおよびソフトウェア構成は**表 2** のとおりである．

図 12 は CINT2000 の結果である．197.parser で 5%ほど性能改善がみられる．197.parser は構文解析ベンチマークである．その他はほぼ同じような結果となった．

図 13 は CFP2000 の結果である．168.wupwise と 301.apsi で改善がみられる．4 MB 境界のアラインを有効にするとさらに，いくつかのベンチマークで性能の改善がみられる．文献 5) の alpha での Super Page Kernel 実装の性能評価と比べて，性能の改善は少なかった．原因は alpha が 8 KB，64 KB，512 KB と 4 MB ページをサポートしているのに対し，IA32 では 4 KB と 4 MB ページというようにサポートをしているサイズの間隔が大きいためではないかと考えられる．

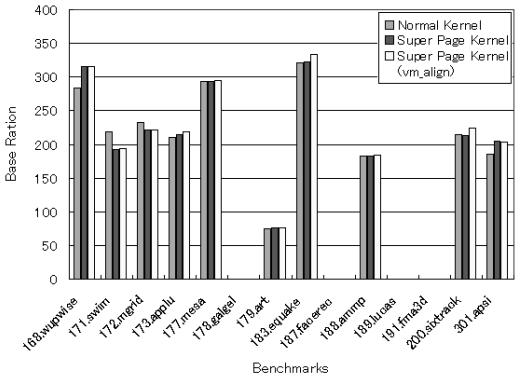


図 13 IA32 での SPEC CFP2000 の結果
Fig. 13 Result of SPEC CFP2000 in IA32.

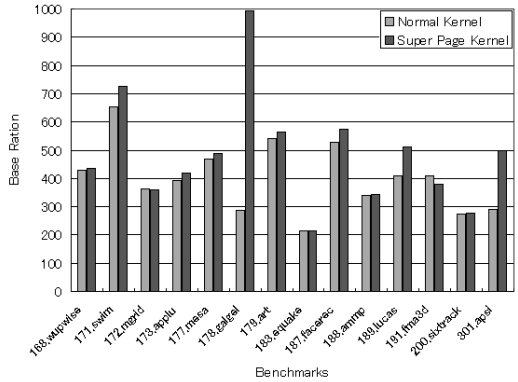


図 14 Alpha での SPEC CFP2000 の結果（文献 5) より）
Fig. 14 Result of SPEC CFP2000 in Alpha cpu.

図 14 は文献 5) の CFP2000 の結果である．

今回コンパイルできなかった 178.galgel にも性能向上が見込めるため，大規模シミュレーションで効果があると考える．しかし，Super Page 割当ての失敗をしたときの性能低下があり，通常の Kernel より低くなってしまう場合があった．その原因は，Super Page 割当て失敗後のページのダウングレードのペナルティが大きいためだと考えることができる．性能低下を抑えるために，Super Page 割当て失敗の際には，ページのダウングレードをせずに，失敗させたままにするアプローチも考えられ，その際の性能も測っていきたい．

6. カーネルインタフェースの追加

実行時のモニタ機構として，Super Page 予約数，割当て数，割当て失敗数を proc ディレクトリから取得できるようにした．また，Buddy System のページ残数の表示などを追加しモニタ強化を行っている．さらに，

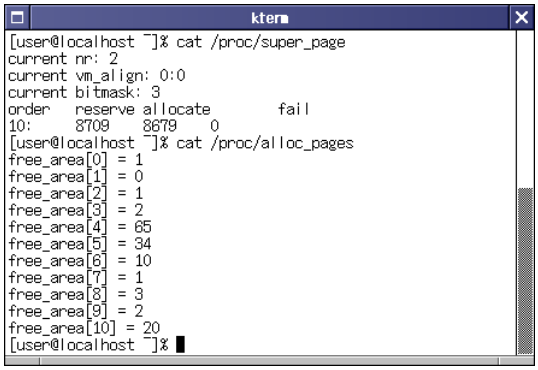


図 15 Super Page モニタ機構
Fig. 15 Super Page Monitor for Linux.

sysctl インタフェースによって Super Page の on/off ならびに仮想アドレスのアラインを OS を再起動することなく制御できるようにしている．**図 15** にモニタの様子を示す．

7. ま と め

我々は，IA32 にシャドーページディレクトリを用い Super Page を実装を行った．性能面では，行列転置ベンチマークプログラムで Celeron プロセッサで約 2 倍，Pentium4 プロセッサで約 6 倍の性能の改善がみられ，Super Page Kernel の有効性が確認できた．SPEC CPU2000 では，性能の改善があまりみられなかったが，サポートページサイズの間隔が大きいことが 1 つの原因と考えている．その間隔を埋めるためにキャッシュの使用効率を上げる Page Coloring が効果的であると考え，Super Page Kernel との統合を検討している．さらに今後，Super Page Kernel 性能低下ケースの解析を進め，より性能の向上を目指していく．

参 考 文 献

1) Ganapathy, N. and Schimmel, C.: General Purpose Operation System Support for Multiple Page Size, *Proc. USENIX 1998 Annual Technical Conference*, pp.91–104 (1998).
2) Subramanian, I., Mather, C., Peterson, K. and Raghunath, B.: Implementation of Multiple Pagesize Support in HP-UX, *Proc. USENIX 1998 Annual Technical Conference*, pp.105–118 (1998).
3) Shimizu, N. and Takatori, K.: A Linux Super Page Kernel for Alpha, Sparce64 and IA32 —Reducing TLB Misses of Applications, *MEDEA 2002 workshop* (2002).
4) 高取 研：IA32 版 Linux Super Page の開発，東海大学工学部 2001F 年度卒業研究論文 (2001).
5) Shimizu, N.: Multi-Granularity Page Size Support for Linux and the Performance Evaluation, *Wuhan University Journal of Natural Sciences*, Vol.6, No.1–2 (2001).
6) Bovet, D.P. and Cesati, M.: *Linux Kernel*, O'REILLY (2001).
7) Maxwell, S.: *Linux Core Kernel*, Serendip (2000).
8) SPEC CPU2000. http://www.specbench.org/

(平成 14 年 12 月 21 日受付)
(平成 15 年 4 月 1 日採録)

早坂 晴康
2002 年東海大学工学部通信工学科卒業．同年同大学大学院工学研究科電気工学専攻博士前期課程入学，現在に至る．オペレーティングシステムの研究に従事．

清水 尚彦（正会員）
1985 年上智大学大学院理工学研究科博士課程前期課程修了．同年（株）日立製作所入社，1994 年上智大学大学院理工学研究科博士課程後期課程修了．博士（工学）．1995 年より東海大学工学部，現在，東海大学電子情報学部助教授．コンピュータアーキテクチャ，アプリケーション指向アーキテクチャ，オペレーティングシステム等に興味を持つ．電子情報処理学会，ACM，IEEE 各会員．