単一仮想記憶の特徴を考慮したスケジューラ構成と評価

3F－3

寺本 圭一　岡本 利夫

(株) 東芝　研究開発センター　情報・通信システム研究所

1 はじめに

我々が現在研究開発中のOS(Cubix[CUBe of unIX])では，単一仮想記憶空間上で走行するスレッド毎に各領域に対する保護属性を個別に設定可能な保護モデルを採用している．これにより，各スレッドによるデータへのアクセスやプログラム実行に関する保護を安全かつ柔軟な形で実現する枠組みを提供している．

こうしたスレッド毎に設定可能なアクセス保護機構を既存の多くのアーキテクチャ上で実現する場合，スレッド単位に個別のページ・テーブルを保持させるという方式が考えられる．

本稿では，単一仮想記憶空間において個別の保護空間を有しうるスレッド間でスレッド切り替えが行われる際の最適化を考慮したスケジューリング手法を提案し，これにより効果的に処理される各種操作プリミティブに関する性能評価を提示する．

2 単一仮想記憶空間における保護空間切替え時のコストについて

単一仮想記憶空間システムであっても，スレッド単位に空間内の領域保護を実現する場合には，従来の多重仮想記憶空間システムにおけるプロセスと同様，スレッド毎に個別のページテーブルを保持させる必要がある．これは，既存のページレベルの保護機構を採用しているアーキテクチャの多くでは，TLB(Translation Lookaside Buffer)やページテーブルなどの各エントリ内にアドレス変換情報と共に保護属性が混在されているために，例えアドレス空間が単一であっても保護属性のみを分離して扱うことができないことに起因する．

よって，基本的にスレッド切替えの際には，保護空間を切替える必要が生じる(アドレス空間を切替える必要は無い)．この際，software loaded TLB を採用しているアーキテクチャ(UltraSPARC, MIPS など)のように，アドレス空間識別子のようなタグをTLBエントリ内に持つ場合，こうした空間切替え自体にコストは生じないが，hardware loaded TLB を採用しているアーキテクチャ(Pentium, PowerPC, Alpha など)の多くはこうしたタグを持たないため，空間切替え時にはTLBのflush操作を必要とする．さらに，TLB flush後には，メモリ参照によってTLB missが発生し，hardwareにより PTE(Page Table Entry)から必要な情報がTLBにloadされることになる[1]．また，タグつきTLBを持つアーキテクチャであっても，working setが大きく，TLBエントリや利用可能な識別子を少数しか持たない場合には，TLB missの発生頻度は高くなる．

このような空間切替え時のコストを低減させることは，システム全体の効率を向上させることに寄与する．単一仮想記憶空間システムでは，空間内に配置される各領域(プログラムテキスト，データ，ファイルなど)は，この領域に対してアクセス権を持つスレッドからアドレス変換情報を共有して参照することができるので，この点を積極的に利用して最適化が行なえるような仕組みを構築することを考える．

3 保護空間と thread view について

Cubixにおいて，あるスレッドに関する保護空間とは，スレッドがある時点で直接参照可能と判明している領域集合と潜在的に参照可能な領域集合の両方を含む．このうち，直接参照可能な領域集合を thread view と呼ぶことにする．

この thread view は，ある時点で直接参照可能と判明している領域の範囲とそれぞれの保護属性の他に，領域内のページがvalidかどうかなどの状態から構成され，あるスレッドの視点から見た空間に対する見え方を規定する．すなわち，保護空間がスレッドの存続期間中に動的に変更されうる参照可能な領域集合全体(潜在的に有効なものも含む)をあらわすのに対して，thread view はスレッド実行中のある時点でのスナップショット的な性質を有する．

ここで，thread view はスレッド固有の情報であるが，いくつかのスレッド集合内で，thread view 間の親和性(affinity)を考慮して，ページテーブルを共有化させるような最適化を施すことが可能である．ページテーブルの共有化は，ページテーブル資源の削減とキャッシング効率の向上につながる．

今回提案するスケジューラは，この最適化情報を利用し，ページテーブルの共有化可能なスレッド同士をでき

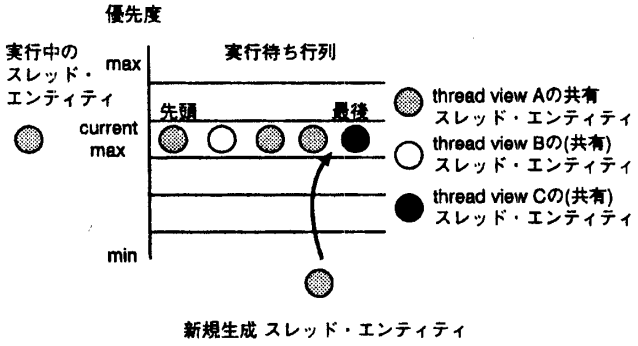
A Scheduler Performance for a Single Virtual Address Space Operating System
by Keiichi TERAMOTO, Toshio OKAMOTO,
Communication and Information Systems Research Labs.,
R&D Center, TOSHIBA CORPORATION
1, Komukai-Toshiba-cho, Saiwai-ku, Kawasaki 210, Japan

[1] この際，TLB エントリの load 対象となる PTE がキャッシュ内に存在するかどうかは，性能に影響を及ぼす．

るだけ連続して実行されるよう実行待ち行列内制御を行なうことを特徴としている．

4 thread view affinity scheduling

thread view affinity scheduling は，新規スレッド生成時にスレッド・エンティティを実行待ち行列内に登録する際，および，タイマ割り込み時の優先度再計算時に待ち行列を操作する際など，ユーザによって指定されるイベントを契機にして，スレッド制御ブロックの属性内に thread view affinity scheduling の適用を明記するフラグがセットされていれば，そのスレッド・エンティティを行列内の他の thread view 共有可能なスレッド・エンティティに連続して配置しようと試みる (飢餓の回避は考慮される)．



図：スレッド・エンティティの実行待ち行列への登録

スレッド・エンティティを thread view affinity scheduling の対象とすることをスケジューラに通知するには，本スケジューリング用に提供されるスレッド・インタフェースを用いてスレッド属性を記述することにより行なわれる．

本スケジューリングは，結果的に fair share スケジューリングとはならない．ページテーブル共有可能スレッド群が共有ページテーブルを持たないスレッドよりも比較的優先的に実行されうるポリシーである．従って，TLB フラッシュや TLB miss の発生頻度の低下による応答率の向上は期待できるが，thread view の親和性の高いスレッドが多数存在しうるような状況下で利用されることが望ましい．

5 測定 / 評価

本スケジューリング適用による性能向上の程度を調べるため，簡単な評価プログラムにより測定を行なった．

本測定に使用したターゲットマシンは，東芝 DynaBook SS-R590 であり，Intel Pentium プロセッサ (90MHz)，メインメモリ 16 M bytes，first-level キャッシュ 16K bytes (命令 / データ分離型で各 8K bytes，キャッシュラインサイズ 32 bytes，2 way set associative)，hardware loaded TLB(命令 (32 エントリ)/ データ (64 エントリ) 分離型，4 way set associative) を持つ．測定は，Pentium プロセッサに実装されているモデル固有レジスタ (MSR) を利用して，下記イベントの発生回数およびクロック・サイクルをカウントすることにより行った．

評価プログラムは，2 個のスレッドを生成し，単一仮想記憶空間上に配置されたある同一データ領域 (4M bytes) 内の先頭位置にある 4 bytes を読み込んだ後に，yield() を発行してそれぞれが自発的にスレッドを切替えていくというものである．ここでは，本スケジューリング方式の適用 / 非適用の影響の差異のみが明確に現れるよう，スレッド生成処理と参照先ページのページインは完了した状態での測定としている．なお，各種割り込み処理ハンドラの影響は排除している[2]．1st スレッドによる参照時には，適用 / 非適用の両者ともページフォルトを発生するが，2nd スレッドでは非適用の場合のみ発生することとなる．

	非適用時		適用時	
	1st	2nd	1st	2nd
data TLB misses	42	42	39	0
data read cache misses	132	123	132	11
data write cache misses	4371	4350	4371	200
code TLB misses	21	21	17	0
code cache misses	31	12	39	2
cycles	45506	45255	42575	8470

6 さいごに

本稿では，各領域の保護属性や状態を考慮した保護領域に対する見え方 (thread view) を導入し，この親和性 (affinity) に基づいて，ページテーブルの共有化に関する制御能力をユーザに与えるスレッド・インタフェース，および，これを利用した "thread view affinity scheduling" と呼ぶスケジューリング方針を提供することにより，ページ・テーブルの資源量 / 管理コストの削減，および，スレッド・ディスパッチ時のページ・テーブル切り替え頻度の低減など，スレッド実行時の軽量化を図った．これは，結果としてプログラム実行中の TLB miss 発生率の低減にも貢献することになる．ただし，本スケジューリングによる最適化は，空間内の領域の使用状況やアーキテクチャによってその有効性に関して大きく変化しうる．

参考文献

[1] Okamoto et al., "A Micro Kernel Architecture for Next Generation Processor", pp.83-94, Microkernels and Other Kernel Architectures Symposium, USENIX, 1992/4.

[2] Eric J.Kokdinger, Jeffrey S. Chase, and Susan J. Eggers, "Architectural support for single address space operating systems", In Proc. of 5th International Conference on Architectural Support for Programming Language and Operating Systems, pp. 175–186, Vol. 26 of ACM SIGPLAN Notices, 1992/10.

[3] Jochen Liedtke, "On μ-Kernel Construction", In Proc. of 5th ACM Symposium on Operating Systems Principles, pp.237–250, 1995/12.

[2] ページテーブルの共有率は，非適用時に 0%，適用時に 100% としている．