アプリケーションと連動する OS カーネルレベルでの Huge page 割り当て機構

清水　祐太郎 †　山 田 浩 史 †

近年のサーバマシンは数百ギガバイトからテラバイトサイズのメモリを搭載できるようになってきた．たとえば，HP ProLiant シリーズでは最大 6TB のメモリを搭載できる．また，不揮発性メモリの発達により，x86 がサポートしている 256 TiB の仮想アドレス空間よりも大きいサイズのメモリを搭載可能なマシンが登場するといわれている[4]．また，こうした環境では TLB ミスが問題となる．TLB サイズの成長とはメモリサイズのそれと比べると遥かに小さく，巨大なメモリを利用可能であるにもかかわらず TLB ミスによって性能が劣化する場合がある．実際に現行のシステムにおいて，TLB ミスによるレイテンシによって，55% の性能劣化を被るアプリケーションが存在する[8]．

TLB ミスを避ける手法として，Huge page 機能を利用する方法がある．Huge page とは，通常のページサイズよりも大きいページサイズで管理が可能となる CPU の機構である．たとえば x86 では，ページサイズが通常 4 KB であるのに対し，Huge page 機構を利用することによって 2 MB，1 GB までサイズを大きくすることができる．これによって，ページテーブルの 1 エントリあたりが指すアドレスを大きくすることができ，その分 TLB がカバーするサイズも増加する．

これまでに Huge page を管理する手法が提案されてきた．オペレーティングシステム (OS) が API を提供する方式で[5),6)] は，アプリケーションが API を介して Huge page をリクエストできる．しかしながら，予め huge page プールに登録されているページからしかフェッチすることしかができない．また，OS カーネルがアプリケーションから透過的に Huge page に置換する手法がある[7),9),11)]．これらの手法はアプリケーションを変更することなく適用できるものの，アプリケーションが自身でメモリを管理する場合には適用が難しい．なぜなら，アプリケーションレベルでのメモリ管理では，最初に大量のメモリプールを確保してそれの割り当ておよび回収を繰り返すため，OS レベルからするとメモリがプールにあって利用されていないのか，それとも実際に利用されているのかがわからない．そのため，利用されていないメモリ領域に Huge page を割り当てるなどの非効率的な振る舞いが生じてしまう．その他にも，Huge page を割り当てることによる外部断片化を防ぐ手法[3),10)] などが提案されている．

本研究では，OS カーネルにおいて Huge page の割り当てを統一的に管理しつつ，アプリケーションレベルでのメモリ管理に親和性の高い手法を提案する．提案手法では，OS カーネルがアプリケーションに対して Huge page の利用を促進するようヒントを与える．具体的には，OS カーネルが各ページのアクセス状況を監視し，頻繁に利用されているページをアプリケーションに通知する．通知を受けたアプリケーションは，通知されたページのアドレスをヒントに自身のメモリオブジェクトをコンパクションし，Huge page 化できるよう連続した領域にオブジェクトをまとめ，使用頻度の高いページ群を作成した上で，OS カーネルが Huge page 化を行なう．これにより，無駄な Huge page の割り当てを避けつつ，OS カーネルが Huge page を管理することによって複数アプリケーションに公平に Huge page を割り当てることができる．

本研究では，第一歩として memcached[2)] および Linux を対象として提案手法を作り込んでいる．Linux カーネル内では，監視対象のプロセスの仮想アドレス空間を対象に，Idle Page Tracking 機構を利用することでページのアクセス頻度を抽出する．その後，アクセス頻度の高いページの仮想アドレスをプロセスに対して通知する．Memcached ではスラブを利用して自身のメモリオブジェクトを管理しているため，スラブレベルでのコンパクションを行なう．Linux カーネルから通知のあったページに属するスラブを連続したページ領域に可能な限り移動させる．アクセス頻度の高いページが連続したら，Linux カーネルはその領域を Huge page へと変換する．

現在，本方式を実装しているところである．今後は，GET アクセスに偏りのあるワークロードを意図的に用意して提案方式の挙動を詳細確認する．それが終わり次第，YCSB[1)] などの複雑なワークロードを利用して提案方式を評価する予定である．

参 考 文 献

1) B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. Benchmarking

† 東京農工大学
Tokyo University of Agriculture and Technology

Cloud Serving Systems with YCSB. In *Proc. of the 1st ACM Symp. on Cloud Computing (SoCC '10)*, pages 143–154, 2010.

2) Dormando. memcached – a distributed memory object caching system. `http://www.memcached.org/`.

3) M. Gorman and P. Healy. Supporting Superpage Allocation Without Additional Hardware Support. In *Proc. of the 7th Int'l Symp. on Memory Management (ISMM '08)*, pages 41–50, 2008.

4) I. E. Hajj, A. Merritt, G. Zellweger, D. Milojicic, R. Achermann, P. Faraboschi, W. mei Hwu, T. Roscoe, and K. Schwan. SpaceJMP: Programming with Multiple Virtual Address Spaces. In *Proc. of the 21st Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS '16)*, pages 353–368, 2016.

5) Huge page support in Mac OS X. `https://developer.apple.com/legacy/library/documentation/Darwin/Reference/ManPages/man2/mmap.2.html`.

6) Huge page support in Windows. `https://msdn.microsoft.com/en-us/library/windows/desktop/aa366720(v=vs.85).aspx`.

7) Y. Kwon, H. Yu, S. Peter, C. J. Rossbach, and E. Witchel. Coordinated and Efficient Huge Page Management with Ingens. In *Proc. of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16)*, pages 705–721, 2016.

8) C. McCurdy, A. L. Coxa, and J. Vetter. "investigating the tlb behavior of high-end scientific applications on commodity microprocessors". In *"Proc. of the IEEE Int'l Symp. on Performance Analysis of Systems and software (ISPASS '08)"*, pages 95–104, 2008.

9) J. Navarro, S. Iyer, P. Druschel, and A. Cox. Practical, Transparent Operating System Support for Superpages. In *Proc. of the 5th Symp. on Operating Systems Design and Implementation (OSDI '02)*, pages 84–104, 2002.

10) A. Panwar, N. Patel, and K. Gopinath. A Case for Protecting Huge Pages from the Kernel. In *Proc. of the 7th ACM SIGOPS Asia-Pacific Workshop on Systems (APSys '16)*, pages 15:1–15:8, 2016.

11) Transparent Hugepage. `https://lwn.net/Articles/359158/`.