

A Case for Compiler-driven Superpage Allocation

Joshua Magee
Department of Computer Science
Texas State University
San Marcos, TX
jm1576@txstate.edu

Apan Qasem
Department of Computer Science
Texas State University
San Marcos, TX
apan@txstate.edu

ABSTRACT

Most modern microprocessor-based systems provide support for superpages both at the hardware and software level. Judicious use of superpages can significantly cut down the number of TLB misses and improve overall system performance. However, indiscriminate superpage allocation results in page fragmentation and increased application footprint, which often outweigh the benefits of reduced TLB misses. Previous research has explored policies for smart allocation of superpages from an operating systems perspective. This paper presents a compiler-based strategy for automatic and profitable memory allocation via superpages. A significant advantage of a compiler-based approach is the availability of data-reuse information within an application. Our strategy employs data-locality analysis to estimate the TLB demands of a program and uses this metric to determine if the program will benefit from superpage allocation. Apart from its obvious utility in improving TLB performance, this strategy can be used to improve the effectiveness of certain data-layout transformations and can be a useful tool in benchmarking and empirical tuning. We demonstrate the effectiveness of this strategy with experiments on an Intel Core 2 Duo with a two-level TLB.

Categories and Subject Descriptors

D.3.4 [Software]: Processor—compilers, memory management

General Terms

Design, Performance, Experimentation

1. INTRODUCTION

As application data footprints grow larger, so too does the importance of the memory hierarchy in improving program performance. The translation lookaside buffer (TLB), which lies in the critical path of a memory access plays an

important role in improving application performance. Studies have shown that for many data-intensive applications, increased TLB misses can not only degrade performance but in many cases becomes the principal bottleneck [7]. Because of its importance to improving application performance, the TLB has received considerable attention from industry and academia alike. Many strategies have been proposed for improving TLB performance both at the hardware and software level.

Among the different strategies proposed, the use of superpages has been the dominant one. Most micro-processor based systems today support multiple page sizes, from smaller pages of size 4K-16K to larger pages of size 2M-16M. The use of large pages increases TLB coverage and reduces TLB misses. However, indiscriminate use of large pages leads to unwarranted increase in application data footprint and causes internal page fragmentation. To ameliorate the effects of fragmentation, several strategies have been proposed that use heuristics for intelligent allocation of superpages. Since the operating system is primarily responsible for allocating physical pages, most techniques for managing superpages have been OS-centric. However, because of the role the compiler plays in program analysis and in setting up the run-time environment, it can help the operating system in efficient allocation of superpages. There are two key advantages to having a compiler-based scheme for superpage allocation:

(i) Programmer productivity and code portability: On most systems, to acquire superpages, the programmer needs to insert code that sends an explicit request to the operating system through the memory allocator. Although there are APIs that can streamline the code to be inserted, the responsibility to request pages still remains with the programmer. Having automatic support for allocation into superpages at the compiler-level relieves the programmer from this responsibility. Moreover, since support for superpages vary across systems, programs that contain explicit code for superpage requests become less portable. Having the compiler insert the platform-specific code increases portability and maintainability.

(ii) Enhanced information for allocation decisions: A key advantage the compiler has over the operating system is that it has knowledge of the memory access behavior of an application. The OS, when allocating memory for a process only considers the data footprint - that is the amount of memory required by the program. However, the data footprint does not necessarily indicate how much pressure the application is going to put on the TLB. The actual TLB usage

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ACMSE '09 March 19-21, 2009, Clemson, SC, USA.

©2009 ACM 978-1-60558-421-8/09/03 ...\$10.00

