Cache-Content-Duplication for Valid Blocks

Marios Kleanthous[*],
Yiannakis Sazeides[*,1]

[*] *Department of Computer Science, University of Cyprus, Nicosia, Cyprus*

ABSTRACT

Cache-content-duplication (CCD) occurs when there is a miss for a block in a cache and the content of the missed block resides already in the cache but under a different tag. Caches aware of content-duplication can have smaller miss penalty by fetching, on a miss to a duplicate block, directly from the cache instead from lower in the memory hierarchy, and can have lower miss rates by allowing only blocks with unique content to enter a cache.

Previous work show that CCD is frequent for basic block and trace caches but rare at the granularity of an instruction cache block, never more than 3%. This work examines the potential of CCD for instruction caches at the granularity of valid blocks. We show that CCD is a frequent phenomenon and that an idealized duplication-detection mechanism for instruction caches has the potential to increase performance of an out-of-order processor, with a 2-way eight instruction per block 16KB instruction cache, often by more than 5% and up to 20%.

KEYWORDS: cache compression

1 Introduction

The importance of caches and memory hierarchy has increased over time due to the growing gap between processor and memory performance [Wulf95]. Caches, consequently, have been central to numerous research studies. Several techniques have been proposed to improve various aspects of caches by reducing their miss rates, size, latency and energy. Most of these techniques attempt to exploit different types of properties of memory addresses and data, such as locality [Denn70], predictability [Baer91, Lipa96], and redundancy [Kjel96, Coop99].

Previous work identified a new cache property that may influence cache performance: the cache-content-duplication (CCD) [Klea05]. This phenomenon occurs when there is a miss for a block in a cache and the content of the missed block resides already in the cache but under a different tag. CCD can happen when cache blocks with different tags have exactly the same content. Therefore, CCD is a manifestation of redundancy in the cache content.

[1]E-mail: {mklean,yanos}@cs.ucy.ac.cy

In [Klea05] the phenomenon was measured for instruction caches, basic block caches and trace caches and found that CCD is frequent for basic block and trace caches but rare at the granularity of an instruction cache block, never more than 3%. One way to increase the frequency of CCD for regular a instruction caches, is to consider the duplication between *valid* instruction sequences send down the pipeline on an I$ access, instead of *entire* instruction cache blocks. In [Cont95] a *valid* sequence is defined as the static consecutive instruction sequence starting from the current PC until: the first conditional branch that is predicted taken, or the first unconditional branch, or fetch bandwidth number of instructions are read from the cache. A valid block is identified by the starting PC and a bit mask. The mask size is equal to the cache fetch bandwidth. The bit mask can be produced each cycle, in a pipeline, using the BTB and the direction predictor [Cont95]. The mask indicates the location of the first taken branch in a sequential instruction sequence. A valid block represents, therefore, the predicted instructions that are send down the pipeline after a cache access, and we will refer to it henceforth as a *valid block*.

This work shows that CCD for valid blocks is a frequent phenomenon and that an idealized duplication-detection mechanism for instruction caches has the potential to increase performance of an out-of-order processor, with a 2-way eight instruction per block 16KB instruction cache, often by more than 5% and up to 20%.

2 CCD Applications: DAC and UCC

Cache latency can be reduced through the detection of misses to blocks with a duplicate in the cache. We refer to such cache as the Duplicate-Aware-Cache (DAC). Latency can be reduced by fetching the block from the cache instead of reading it from lower in the memory hierarchy. Therefore, a DAC can reduce the miss penalty of a duplicated miss down to a cache hit. Because the latency of a duplicated miss is likely small, henceforth, we refer to it as a secondary hit (primary hits are those that hit directly in the cache).

CCD can also be used to reduce misses by allowing only blocks with unique content to enter a cache. We refer to such cache as the Unique-Content-Cache (UCC). A UCC can reduce conflict misses because it allows a smaller number of blocks to enter a cache. A UCC needs to be also duplicate-aware to detect misses to duplicated blocks.

Fig. 1 shows the performance potential of DAC and UCC in terms of normalized IPC for 16KB DAC and UCC instruction caches over a 16KB regular instruction cache.

Results are presented for secondary hit latencies of 0, 1, 2 and 3 cycles (denoted in the graph as DAC-0, DAC-1, DAC-2 and DAC-3 or UCC-0, UCC-1, UCC-2 and UCC-3 respectively) and for various L2 miss latencies (10, 15, 20, 25 and 30 cycles). All other processor parameters are as in Table 1. The data show both DAC and UCC to have performance potential up to 15% and 20% respectively. The benchmarks *gcc95*, *perl95* and *vortex00* are the ones with the largest potential. Benchmark *mesa00* does not benefit from CCD because it has very few misses for duplicated blocks. The potential improves with increasing L2 miss latency for both DAC and UCC. The DAC performance is rather insensitive to secondary hit latency, however, for UCC the effects of secondary hit latency can be detrimental. For example with UCC-3 latency many configurations for *go95* and *perl95* suffer a performance degradation.

The performance potential analysis suggests that, for good performance and room for duplicated hit latency, the DAC with two or three cycle latency is a good compromise but

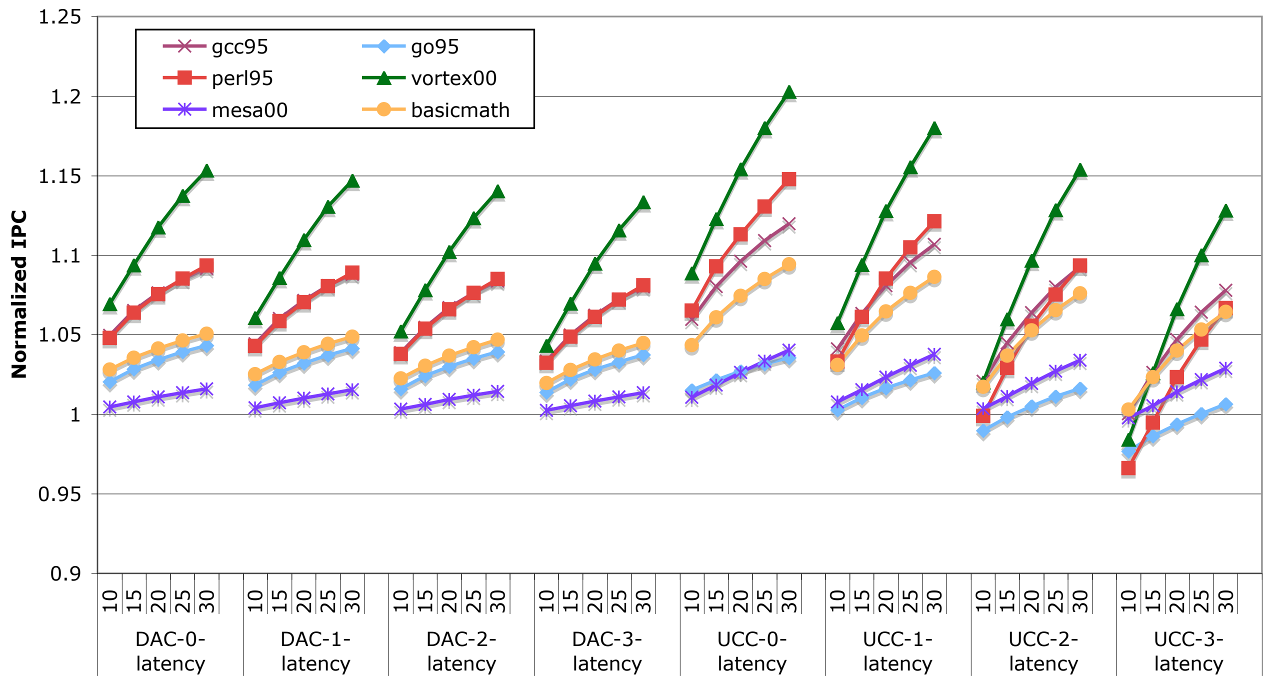

Figure 1: Performance potential of DAC and UCC for a 2-way, 16KB, 8 instructions per block, instruction cache, for valid blocks (Normalized IPC)

for better performance with some room for detecting CCD, UCC with at most one cycle latency is better.

In the case of a single cycle latency cache, a single cycle secondary hit latency can be achieved by accessing the CCD mechanism and cache in parallel. By the end of the cache access cycle the CCD mechanism will provide an alternative tag-index to access the cache for a secondary hit. An extra cycle is needed to access the cache for a secondary hit. A zero cycle secondary hit latency is possible, but may require more pervasive changes in the processor front-end.

fetch/issue/commit	4/4/4
Queue/LSQ/ROB	64/32/64
Stages	14
ALU/Data Cache Ports	4/2
L1 instruction cache	16KB 2-way 32B/block, 1 cycle
L1 data cache	32KB 2-way 64B/block, 2 cycles
L2 unified cache	2MB 8-way 128B/block, 20 latency
Main memory latency	200 cycles
Cond. branch predictor	8KB combining predictor
BTB	1024 entries
RAS	32 entries
Indirect predictor	512 entries

Table 1: Out of Order Processor Configuration

References

[Baer91] J. BAER AND T. CHEN. An effective on-chip preloading scheme to reduce data access penalty. In *Proceedings of the 1991 ACM/IEEE conference on Supercomputing*, pages 176–186, November 1991.

[Cont95] T. CONTE, K. MENEZES, P. MILLS, AND B. PATEL. Optimization of instruction fetch mechanisms for high issue rates. In *Proceedings of the 22nd International Symposium on Computer Architecture*, June 1995.

[Coop99] K. COOPER AND N. MCINTOSH. Enhacning Code Compression for Embedded RISC Processors. In *Proceedings of PLDI*, May 1999.

[Denn70] P. DENNING. Virtual Memory. *ACM Computing Surveys (CSUR)*, 2(3):153–189, September 1970.

[Kjel96] M. KJELSO AND S. M. GOOCH. Design and Performance of a Main Memory Hardware Data Compressor. In *Proceedings of the 22nd EUROMICRO Conference*, pages 423–430, September 1996.

[Klea05] M. KLEANTHOUS AND Y. SAZEIDES. The Duplication of Content in Instruction Caches and its Performance Implications. Technical Report TR-CS-01-2005, University of Cyprus, January 2005.

[Lipa96] M. LIPASTI, C. WILKERSON, AND J. SHEN. Value Locality and Load Value Prediction. In *International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 138–147, October 1996.

[Wulf95] W. WULF AND S. MCKEE. Hitting the memory wall: implications of the obvious. *Computer Architecture News*, 23(1):20–24, March 1995.