Online Cache Modeling for Commodity Multicore Processors

Richard West
Boston University
Boston, MA 02215, USA
richwest@cs.bu.edu

Puneet Zaroo
VMware, Inc.
Palo Alto, CA 94304, USA
puneetz@vmware.com

Carl A. Waldspurger
VMware, Inc.
Palo Alto, CA 94304, USA
carl@vmware.com

Xiao Zhang
University of Rochester
Rochester, NY 14627-0226
xiao@cs.rochester.edu

ABSTRACT

Modern chip-level multiprocessors (CMPs) contain multiple processor cores sharing a common last-level cache, memory interconnects, and other hardware resources. Workloads running on separate cores compete for these resources, often resulting in highly-variable performance. To improve fairness and performance, it is helpful to co-schedule workloads having minimal cache and other forms of resource contention. In this work, we develop several cache modeling techniques to help make informed resource management decisions.

Using only commonly-available performance counters on existing processors, we introduce an efficient online technique for estimating the cache occupancies of software threads. We derive an analytical model that considers the impact of set-associativity, line replacement policy, and memory locality effects. We demonstrate the effectiveness of occupancy estimation with a series of CMP simulations using SPEC benchmarks. Our occupancy estimation technique is currently being used to develop online utility functions, such as miss-ratio curves (MRCs), which capture performance impacts as a function of resource usage. We are leveraging both online cache occupancy estimation and MRC construction in our ongoing studies of cache-aware scheduling.

Categories and Subject Descriptors

D.4.8 [**Operating Systems**]: Performance—*measurements, modeling and prediction*

General Terms

Measurement

1. CACHE OCCUPANCY ESTIMATION

For the purposes of our model, we consider a shared last-level cache that may be direct-mapped or n-way set associative. Our objective is to determine the amount of cache space occupied by some thread, τ, at time t, given contention for cache lines by multiple competing threads.

Since hardware caches reveal very little information to software, we use hardware performance counters to infer cache state. Using two commonly-available hardware performance events, namely the *local* and *global* last-level cache misses, we estimate the number of cache lines, E, occupied by τ at time t. Global cache misses are accumulated across all cores, rather than just the local core.

We start by assuming the shared cache is accessed uniformly at random and later relax this requirement in Section 1.1. We also assume each cache line is allocated to a single thread at any time. Data sharing is not considered in this paper, although it is part of our ongoing work.

Cache occupancy is effectively dictated by the number of misses experienced by a thread, because cache lines are allocated in response to such misses. Essentially, the current execution phase of a thread τ_i influences its cache investment, since any of its lines that it no longer accesses may be evicted by conflicting references to the same cache index by other threads. Evicted lines no longer relevant to the current execution phase of τ_i will not incur subsequent misses that would cause them to return to the cache.

In what follows, let m_l represent the number of misses experienced by the *local* thread, τ_l, under observation over some sampling interval. This term also represents the number of cache lines allocated due to misses. We denote m_o to represent the aggregate number of misses by every thread *other* than τ_l, on all cores of a CMP that cause cache lines to be allocated in response to such misses. Finally, τ_o represents all other threads, as though they were acting as a single aggregate thread.

Theorem. *Consider a cache of size C lines, with E cache lines belonging to τ_l and $C-E$ cache lines belonging to τ_o at some time, t. If, in some interval, δt, there are m_l misses corresponding to τ_l and m_o misses corresponding to τ_o, then the expected occupancy of τ_l at time $t+\delta t$ is: $E' = E + (1 - \frac{E}{C}) \cdot m_l - \frac{E}{C} \cdot m_o$*

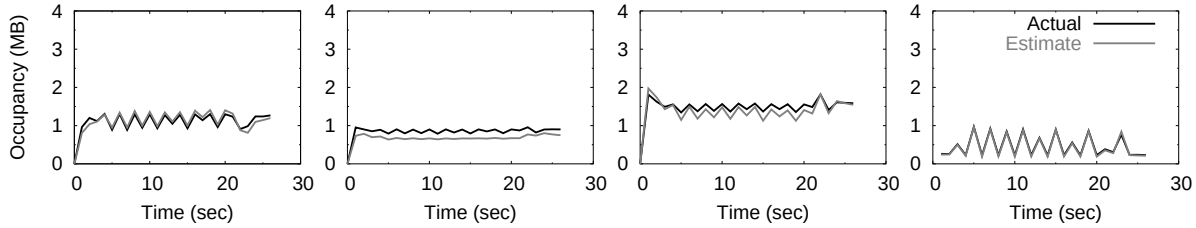
Proof. The proof is presented in the full paper [1]. □

The linear model described above consists of an inexpensive computation that requires only the ability to measure per-core and per-CMP cache misses, which is provided by most modern processor architectures.

1.1 Set-Associative Caches

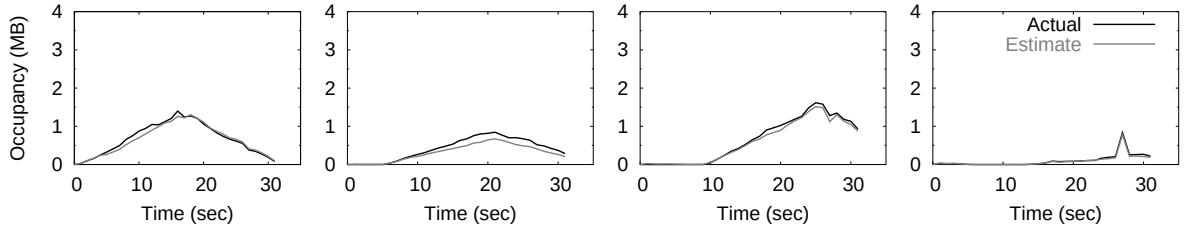
So far, our analysis has assumed that each line of the cache is equally likely to be accessed. Over the lifetime of a large set of threads, this is a reasonable assumption. However, commodity CMPs feature n-way set associative caches, and line replacement policies based on schemes such as *least recently used* (LRU). We modified the linear model to additionally incorporate cache *hit* information, thereby reflecting line reuse probabilities due to LRU ef-

PACT'10, September 11–15, 2010, Vienna, Austria.
ACM 978-1-4503-0178-7/10/09.



(a) `xalancbmk` (b) `twolf00` (c) `art00` (d) `wupwise00`

Figure 1: Co-runner occupancies for quad-core system.



(a) `xalancbmk` (b) `twolf00` (c) `art00` (d) `wupwise00`

Figure 2: Occupancy estimation for 4 out of 10 workloads in an over-committed quad-core system.

fects. As with miss counts, hit counts are available via performance counters on most modern processors. Consequently, the occupancy equation can be rewritten as

$$E' = E(1 - m_o p_l) + (C - E)m_l p_o \quad (1)$$

where p_l is the probability that a miss falls on a line belonging to τ_l, and p_o is the probability that a miss falls on a line belonging to τ_o. In the theorem described earlier each line is equally likely to be replaced, meaning $p_l = p_o = 1/C$. Considering LRU effects, we calculate

$$r_l = (h_l + m_l)/E \quad (2)$$
$$r_o = (h_o + m_o)/(C - E) \quad (3)$$

to quantify the frequency of reuse of the cache lines of τ_l and τ_o, respectively, since we are unable to precisely know which line is the least recently used. h_l and h_o represent the number of cache hits experienced by τ_l and τ_o, respectively, in the measurement interval. Considering the probability that a miss evicts a line belonging to a thread is inversely proportional to its reuse frequency, we assume the following relationship:

$$p_o/p_l = r_l/r_o \quad (4)$$

Furthermore, since a miss must fall on some line in the cache with probability 1:

$$p_l E + p_o(C - E) = 1 \quad (5)$$

Solving Equations 4 and 5, we obtain:

$$p_o = r_l/[r_o E + r_l(C - E)] \quad (6)$$
$$p_l = r_o/[r_o E + r_l(C - E)] \quad (7)$$

The values of p_o and p_l from Equations 6 and 7 can be used in Equation 1 to obtain the hit-adjusted occupancy estimation model, which approximates LRU effects.

1.2 Experiments

We evaluated the cache estimation models on Intel's CMPSched$im simulator, which supports binary execution and co-scheduling of multiple workloads. We configured the simulator to use a 3 GHz clock frequency, with private per-core 32 KB 4-way set-associative L1 caches, and a shared 4 MB 16-way set-associative L2 cache. All caches used a 64-byte line size and pseudo-LRU policy.

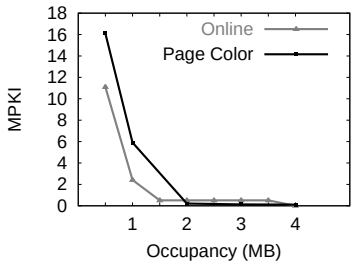


Figure 3: MRC for `twolf00` (misses per kilo-instruction).

Performance counters measuring L2 misses and hits were sampled once per millisecond, after which the occupancy estimates were updated for each software thread. Since cache occupancies exhibit rapid changes at this time scale, we averaged occupancies over 100 millisecond intervals.

Figure 1 shows results for four different co-running benchmarks from the SPEC CPU2000 and CPU2006 suites in a quad-core configuration. Likewise, Figure 2 shows the same benchmark results for an over-committed system, which includes six other threads (not shown) competing for the four cores. Threads are scheduled in round-robin order with a quantum of 100 milliseconds. Even when threads are descheduled in a system that is over-committed, our estimates closely track actual occupancies. Finally, Figure 3 shows a sample MRC generated using our online technique implemented in VMware's ESX hypervisor, compared to offline page-coloring runs. Further details, along with more occupancy estimation and MRC results, are in the full paper [1].

2. REFERENCES

[1] R. West, P. Zaroo, C. A. Waldspurger, and X. Zhang. Online cache modeling for commodity multicore processors. Technical Report VMware-TR-2010-002 (also BUCS-TR-2010-015), VMware, Inc./Boston University, July 2010.