2-level Cache vs Single-level Cache

(Research Essay #3)

Seok Bum Ko

kosby@ele.uri.edu

ELE548 Computer Architecture

9 April, 1999

1. INTRODUCTION

The performance gap between processors and memory leads the architect to this question: Should I make the cache faster to keep pace with the speed of CPUs, or make the cache larger to overcome the widening gap between the CPU and main memory?
By adding another level of cache between the original cache and memory, the first-level cache can be small enough to match the clock cycle time of the fast CPU, while the second-level cache can be large enough to capture many accesses that would go to main memory, thereby lessening the effective miss penalty.
A secondary cache is bigger than the primary cache (and is usually in the same chip as the CPU) and fits between the primary cache and the main memory (RAM). The secondary cache is faster than the main memory, but slower than the primary cache memory. The main functionality of the secondary cache is that, it reduces the miss penalty of the primary cache, by prefetching the data from the main memory. And the blocks in the secondary cache are replaced whena miss occurs in the secondary cache based on any one of the replacement algorithms like the LRU, FIFO or the RANDOM replacement policy. The position of a secondary cache in the memory hierarchy is shown in a simplified manner in Figure 1.

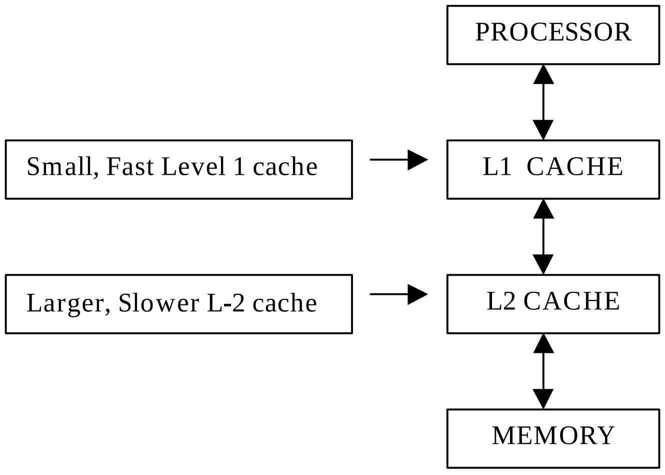


Figure 1. Memory Hierarchy

2. SIMULATION METHODOLOGY

The simulator dineroIV was used, which is available at WARTS (Wisconsin Architectural Research Tools) to determine the performance of the caches. This simulator evaluates only one uniprocessor cache at a time but produces more performance metrics (e.g., traffic to and from memory) and allows various cache design options to be varied which was the main reason and the motivation for it's choice as the simulator.
dineroIV is a trace-driven cache simulator that supports sub-block placement. Simulation results are determined by the input trace and the cache parameters. A trace is a finite sequence

of memory references usually obtained by the interpretive execution of a program or a set of programs. Trace input is read in din format. Cache parameters, e.g. block size and associativity, are set with the command line options. dineroIV uses the priority stack method of memory hierarchy simulation to increase flexibility and improve simulator performance in highly associative caches. This simulator lets one to simulate either a unified cache (mixed, data and instructions cached together) or separate instruction and data caches.
dineroIV differs from most other cache simulators because it supports sub-block placement (also known as sector placement) in which address tags are still associated with cache blocks but data is transferred to and from the cache in smaller sub-blocks. This organization is useful for on-chip microprocessor caches which have to load data on cache misses over a limited number of pins. Sub-block placement allows a cache to have small sub-blocks for fast data transfer and large blocks to associate with the address tags.
dineroIV reads the trace input in the din format from stdin. A din record is a two-tuple label address. Each line of the trace file consists one din record. The rest of the line is ignored so that any comments could be included in the trace file. In the two-tuple label address record the label gives the access type of a reference :

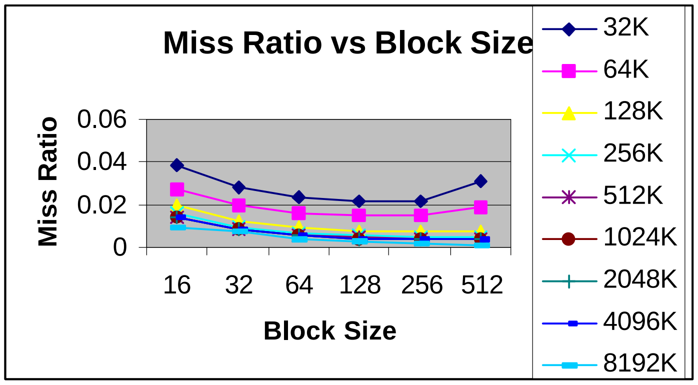
0 read data
1 write data
2 instruction fetch
3 escape record (treated as unknown access type)
4 escape record (causes cache flush)
And the address is a hexadecimal byte-address between 0 and ffffffff inclusively.

Trace-driven simulation is used to evaluate the cache performance because the simulations are repeatable and it allows to vary the cache design parameters and above all it does not require the access to or the existence of the architecture being studied. The objective is to compare the performance of a cache that has a level-2 cache with a cache that does not have a level-2 cache by varying the various parameters of the caches such as the cache size, block size, associativity, replacement policy, write policy and the fetch policy.
Input traces: Spice.din, tex.din, cc1.din: traces for GCC, TeX and Spice
The objective is to compare the performance of a cache that has a level-2 cache with a cache that does not have a level-2 cache by varying the various parameters of the caches such as the cache size, block size, associativity, replacement policy.
Parameters: cache size, block size, associativity, replacement algorithm, write policy
Performance metrics: average memory access time, hit ratio

3. SIMULATION RESULTS

Table 1. Miss rate versus block size

Block Size	Cache Size								
	32K	64K	128K	256K	512K	1024K	2048K	4096K	8192K
16	0.0381	0.0271	0.0195	0.0158	0.0142	0.0136	0.0136	0.0136	0.0091
32	0.0284	0.0197	0.0126	0.0098	0.0087	0.0082	0.0082	0.0082	0.0075
64	0.0236	0.0162	0.0092	0.0067	0.0060	0.0055	0.0055	0.0055	0.0041
128	**0.0213**	0.0148	0.0076	0.0053	0.0047	0.0042	0.0042	0.0042	0.0025
256	0.0215	**0.0147**	**0.0071**	**0.0045**	**0.0040**	**0.0035**	**0.0035**	**0.0035**	0.0015
512	0.0311	0.0191	0.0078	0.0049	0.0042	0.0035	0.0035	0.0035	**0.0009**



Assume the memory system takes 40 clock cycles of overhead and then delivers 16 bytes every 2 clock , cycles, and so on. Thus, it can supply 16 bytes in 42 clock cycles, 32 bits in 44 clock cycles, and so on.

Block Size	Clock Cycles(=Miss Penalty)
16	42
32	44
64	48
128	56
256	72
512	104

Table 2. Average Memory Access Time versus block size

Block Size	Cache Size								
	32K	64K	128K	256K	512K	1024K	2048K	4096K	8192K
16	2.6002	2.2382	1.819	1.6636	1.5964	1.5712	1.5712	1.5712	1.3822
32	2.2496	1.8668	1.5544	1.4312	1.3828	1.3608	1.3608	1.3608	1.33
64	**2.1328**	**1.7776**	1.4416	1.3216	1.288	1.264	1.264	1.264	1.1968
128	2.1928	1.8288	**1.4256**	**1.2968**	**1.2632**	**1.2352**	**1.2352**	**1.2352**	1.14
256	2.5480	2.0584	1.5112	1.324	1.288	1.252	1.252	1.252	**1.108**
512	4.2344	2.9864	1.8112	1.5096	1.4368	1.364	1.364	1.364	1.0936

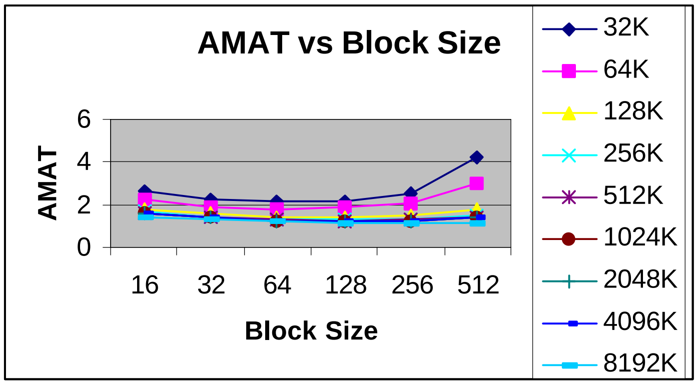
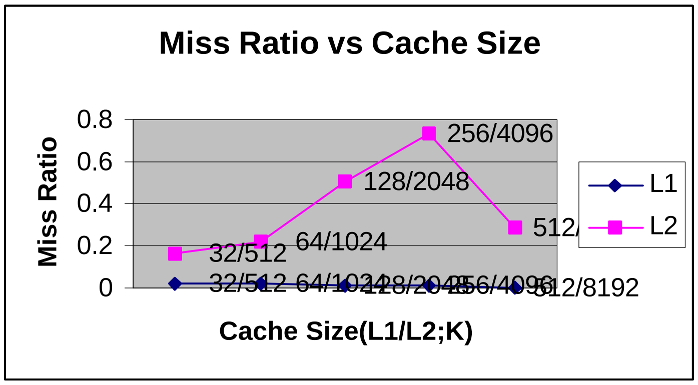


Table 3. Miss ratio versus Cache size

Cache Size	L1	L2	
		L1	L2
32K / 512K; 64 / 128	0.0236	0.0236	0.1647
64K / 1024K; 64 / 128	0.0162	0.0162	0.2204
128K / 2048K; 128 / 128	0.0076	0.0076	0.5038
256K / 4096K; 128 / 128	0.0053	0.0053	0.7291
512K / 8192K; 128 / 256	0.0047	0.0047	0.2850



$AMAT = t_{l1} + p_{l1}(t_{l2}+p_{l2}*t_{mem})$

t_{l1} : level-1 cache hit time – 1 cycle
p_{l1} : level-1 cache miss rate
t_{l2} : level-2 cache hit time – 6 cycles
p_{l2} : level-2 cache local miss rate
t_{mem}: memory access time – 48 for 64 bytes, 56 for 128 bytes and 72 for 256 bytes.

In the case of 256K in L1 cache, 2-level cache's AMAT is calculated as follows:
1 + 0.0053(6+0.7291*56) = 1.2482 clock cycles

Table 4. Average Memory Access Time versus cache size

Cache Size(L1 / L2)	L1	L2
32K / 512K; 64 / 128	2.1328	1.3593
64K / 1024K; 64 / 128	1.7776	1.2971
128K / 2048K; 128 / 128	1.4256	1.2600
256K / 4096K; 128 / 128	1.2968	**1.2482**
512K / 8192K; 128 / 256	1.2632	1.1246

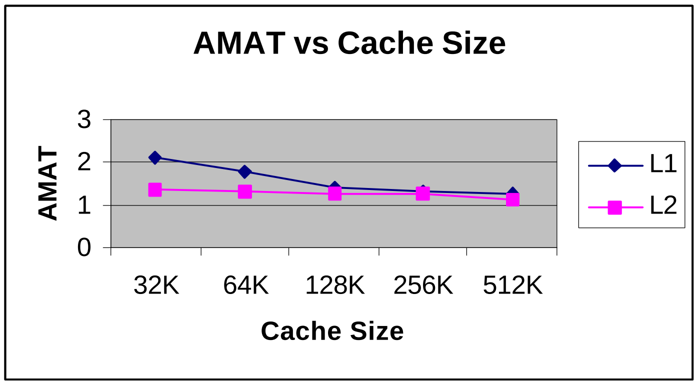


Table 5. Miss Ratio versus Set Associativity

L2 Cache Size	L1 Miss Ratio	Associativity			
		1-way	2-way	4-way	8-way
512K	0.0236	0.1647	0.0978	0.0936	0.0917
1024K	0.1621	0.2204	0.1352	0.1338	0.1337
2048K	0.0076	0.5038	**0.2919**	0.2919	0.2919
4096K	0.0053	0.7291	**0.4211**	0.4211	0.4211
8192K	0.0047	0.2850	**0.2850**	0.2850	0.2850

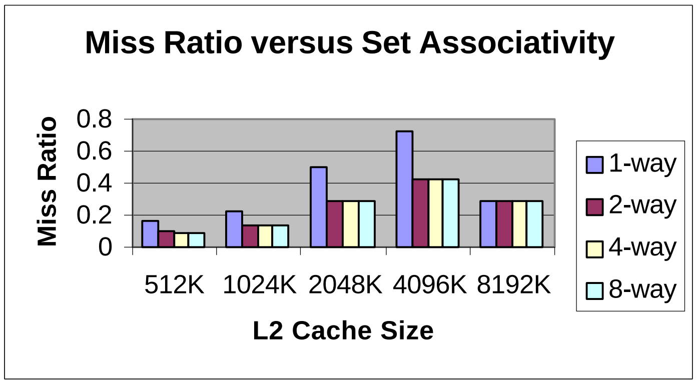
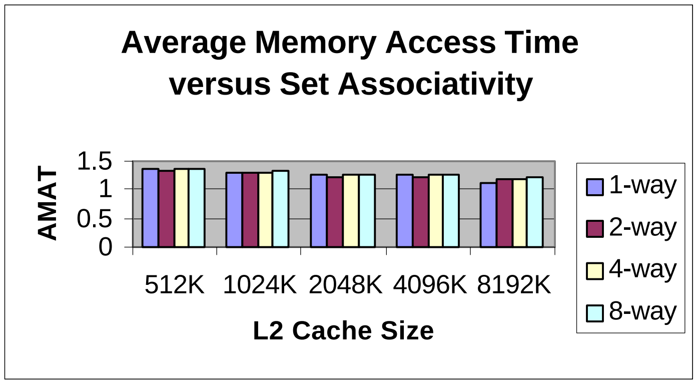


Table 6. Average Memory Access Time(Set-Associativity)

L2 Cache Size	Associativity			
	1-way	2-way	4-way	8-way
512K	1.36	1.33	**1.35**	1.37
1024K	1.30	1.28	**1.30**	1.32
2048K	1.26	*1.23*	**1.25**	1.27
4096K	1.25	1.22	**1.24**	1.26
8192K	1.12	**1.18**	1.20	1.22



4. CONCLUSIONS

A comparison of the performance between a cache with a level-2 cache and a cache without a level-2 cache is made based on trace driven simulation studies using selected input data on the cache simulator dineroIV by varying the different parameters of the cache design. It was found that the performance of a cache with a level-2 cache was better than that of a cache without a level-2 cache.

5. REFERENCES

1. Computer architecture: a quantitative approach by John Hennessy and David Patterson
2. Architectural choices for multi-level cache hierarchies by J.L Baer and W.H Wang
3. The Cache memory book by Jim Handy