Performance Impact of Basic Cache Configuration Parameters Using SimpleScalar

Alan Devkota, Student ID: 2215320
Department of Electrical and Computer Engineering
University of Houston
Houston, TX, USA
adevkota2@uh.edu

***Abstract*—This paper investigates the performance impact of several basic cache configuration parameters, such as the L1, L2, and TLB cache size, associativity, and block size using the SimpleScalar "sim-outorder" model and the SPEC 2000 benchmark suite. The results generated illustrate the relationship between Miss Rate and modifications in cache size, associativity, and block size. The results also reveal the impact of the multi-level cache design as well as the efficacy of the TLB cache in enhancing data locality.**

***Index Terms*—SimpleScalar, SPEC2000, Sim-outorder, Cache Design.**

I. Introduction

SImpleScalar is an open-source microarchitecture simulator application developed by Todd Austin while he was a Ph.D. student at the University of Wisconsin Madison [1]. The SimpleScalar toolkit mimics a virtual computer system featuring processing units, caches, and memory hierarchies. It allows users to develop modeling applications that simulate real programs running on contemporary processors and platforms. Moreover, the toolbox includes a range of sample simulators: from a functional simulator to a sophisticated and dynamically scheduled processor model that supports non-blocking caches, speculative execution, and state-of-the-art branch prediction. It also includes simulators, statistical analysis tools, and infrastructures for debugging and verifying software. The simulator takes binary-formatted cache design parameters as input. It outputs performance statistics such as Miss Rate, Cycles per Instruction (CPI), Instruction per Cycle (IPC), and so on as results [2]. Sim-outorder allows for detailed modeling of microarchitectural design. The out-of-order microprocessor with all its bells and whistles, such as branch prediction, caches, and external memory, is modeled in detail by this tool. This simulator is highly parameterized and can emulate machines of varying numbers of execution units.

This paper explores the performance of the cache by using several basic cache configuration parameters. The presented results imply that, for L1 and L2 caches, the cache size increase lowers the Miss Rate. The multi-level cache design reduces the Miss Rate for the L1 cache. Additionally, an increase in associativity lowers the Miss Rate. CPI and IPC for a fixed benchmark are constant across all configurations; however, they differ with various benchmarks. Moreover, the TLB cache results demonstrate that the TLB instruction and data caches offer good data locality and hence aid in lowering latency.

II. Methodology

The cache parameters corresponding to the Instruction cache and data cache in the configuration files were varied to evaluate the cache performance of L1, L2, and TLB caches. Ten different settings with were run on the simulator using the default parameters, except for the parameters under study. The modifications were either applied to the parameters for the instruction cache or the data cache, or sometimes both.

A. Configurations

The configuration code is of form:
<name>:<nsets>:<bsize>:<assoc>:<repl>
Where each field has the following meaning:
<name> - cache name, which must be unique
<nsets> - number of sets in the cache
<bsize> - block size (page size in case of TLB)
<assoc> - associativity of the cache
<repl> - replacement policy (l | f | r), where l = LRU, f = FIFO, r = random replacement.

For example, il1:512:32:1:l code represents the L1 Instruction cache, with 512 sets, a block size of 32, associativity of 1, and a Least Recently Used (LRU) page replacement policy. The configurations used in the experiment are listed below:

1) Instruction cache size (8KB, 32 KB, and 64KB, block size and associativity follow the default configuration)
2) Data cache size (2KB, 32KB, and 128KB, all 4-way set associative, block size follows the default configuration)
3) Data cache associativity (1-way, 2-way, 8-way, all 32KB, block size follows the default configuration)
4) L2 cache size (128KB, 512KB, and 1MB, block size and associativity follow the default configuration)
5) L1 and L2 block size (32B, 64B, 128B, and 256B, cache size and associativity follow the default configuration)
6) L1 instruction and data cache size and associativity (64KB, 128KB, 512KB, 2-way, 4-way, 8-way)
7) L2 cache size and associativity (128KB, 512KB, and 1MB, 2-way, and 4-way)
8) TLB instruction cache size (64KB, 128KB, and 256KB)
9) TLB data cache size (64KB, 128KB, and 256KB)
10) TLB instruction and data associativity (2-way and 4-way)

B. Benchmark

The cache performance can be tested using the configuration files with variations in the number of sets, block size, associativity, and replacement policy. Then, a shell script is run on each configuration on nine different SPEC2000 benchmarks. The benchmarks used in this experiment are bzip, crafty, eon, gcc, mcf, mesa, swim, galgel and wupwise.

III. Cache Memory Performance and Metrics

A. Types of Caches

- L1 Cache: Level 1 (L1) is a cache located inside the CPU. This cache stores the most recent data. When the data is needed again, the microprocessor first checks this cache rather than the main memory or Level 2 cache. It uses the principle of Locality of Reference, which states that a place recently visited by the CPU has a higher likelihood of being needed again.
- L2 Cache: Level 2 (L2) is a cache located on a separate chip next to the CPU. This cache stores recently used data not found in the L1 cache. Some CPUs have a level 3 (L3) cache chip in addition to their integrated L1 and L2 caches.
- Instruction Cache Vs Data Cache: The Instruction or I-cache stores just instructions, whereas the Data or D-cache stores only data. It is possible to differentiate between instruction and data access patterns by employing the instruction and data cache. The programs generally have more temporal and spatial locality than the data they process and only occasionally require write accesses.
- Unified Cache Vs Split Cache: A unified cache stores both instructions and data. On the other hand, a split cache is composed of an I-cache and a D-cache, two associated but mostly separate units. We can also configure this type of cache to handle two separate units in various ways.

B. Miss Rate

Miss rate is the fraction of cache accesses that result in a miss. It is equal to the number of misses in a cache accesses divided by the total number of accesses. The Miss rate varies with the cache size, associativity, and block size. A cache hit occurs when a cache request a file, and the cache can satisfy that request, whereas a cache miss occurs when the cache does not contain the requested content.

C. Load/ Store Instruction

Load and store instructions are the operations that impact both the CPU and the memory. During execution, both load and stores must first wait for an ALU or address unit to compute their addresses. The load instruction can then access the data cache to get the desired memory data and places them in a register. Typically, the load instruction completes by writing the fetched data into the designated architectural register. Stores follow a different execution pattern. Stores must wait for their operands to be available after receiving their generated addresses. A store, unlike other instructions, is deemed complete when operands become available. Now let us consider a re-order buffer (ROB) is in use. When the ROB indicates that the store will be executed next in the sequence, the memory address and data to be saved are passed to the cache, and a cache store operation is launched.

D. Cycles Per Instruction (CPI)

CPI is a metric that indicates the average number of CPU cycles required to execute an instruction and is thus an indicator of how much system delay affected the running program. Since CPI is a ratio, it will be affected by changes in the number of CPU cycles an application takes or changes in the number of instructions executed. The CPI is always less than one (CPI 1). By getting close to 1, one can attain better CPI values.

E. Instruction Per Cycles (IPC)

IPC represents the average number of instructions executed for each clock cycle. IPC is calculated by dividing the number of machine-level instructions by the number of CPU clock cycles required to complete the instructions on the actual hardware. A high IPC with a high frequency will provide optimal cache performance.

F. Translation Lookaside Buffer

TLB enables quick address translation by utilizing data locality. TLBs help to minimize additional memory access by storing address translations, decreasing the need for memory to do a second access to translate the data and thereby lowering latency.

IV. Experimental Results

The cache parameters corresponding to the Instruction cache and data cache in the configuration files were varied to form 10 different setups. Then, a shell script was used to run each configuration on each SPEC2000 benchmarks. First, 100 million instructions were skipped to ensure proper warming up of the caches. Then, each benchmark was run for 100 million instructions to acquire the results.

A. Cache Size Variation

Figures 1, 2, and 3 depict the average Miss Rates for various cache sizes in L1 instruction caches, L1 data caches, and L2 caches, respectively. It is observed that the Miss Rate reduces when the cache size is increased while keeping the associativity and block size constant in the case of L1 instruction and data caches. The same holds for the L2 cache.

Figure 4 compares L2 cache Miss Rates with L1 instruction and data cache Miss Rates. The average Miss Rate of 512KB L1 instruction and data caches is lower than that of the L2 cache, regardless of L2 cache size. We can observe that multi-level cache design in the memory architecture reduces the miss rate in the L1 cache. This effect can be seen by comparing the miss rate of L1 caches in Figure 1 and Figure 2.

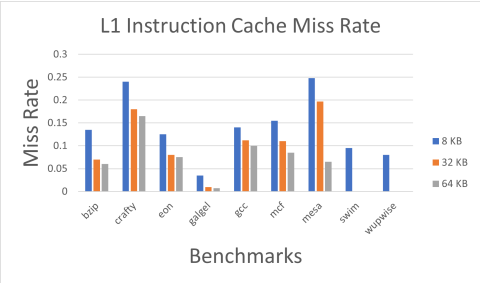

Fig. 1. Instruction Cache L1 Miss Rate by changing the cache size with constant block size of 32B and 1-way associativity.

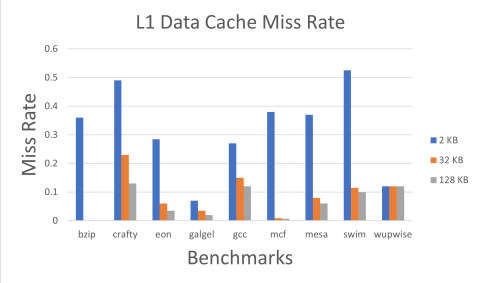

Fig. 2. Data Cache L1 Miss Rate by changing the cache size with constant block size of 32B and 1-way associativity.

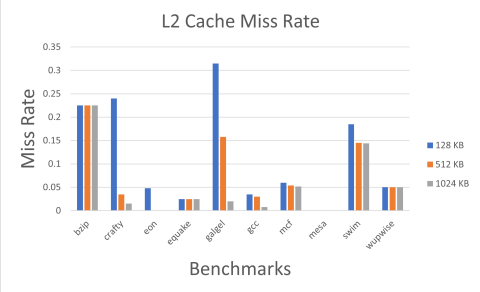

Fig. 3. Miss Rate for L2 cache by changing the cache size with constant block size of 64B and 4-way associativity.

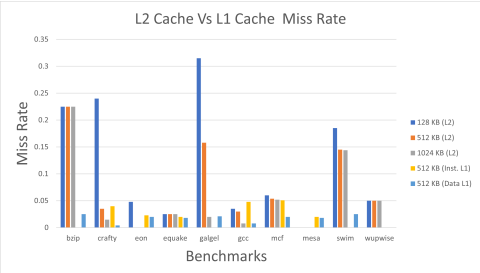

Fig. 4. L2 Cache Vs L1 Cache Miss Rate by changing the L2 cache size and keeping block size of 64B and 4-way associativity (L1 cache size is 512 KB).

B. Associativity Variation

Figure 5 depicts the results achieved for an L1 data cache with varying associativity while keeping cache and block sizes constant. Miss Rate drops considerably when associativity is increased from 2-way to 4-way. However, increasing associativity to 8-way does not reduce the Miss Rate. The result suggests that associativity only has a limited effect on Miss Rate. Any further increase in associativity does not influence the Miss Rate.

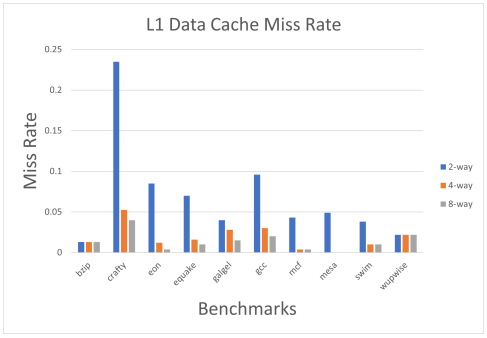

Fig. 5. Miss Rate for L1 data cache by changing the associativity with constant cache size of 32KB and block size of 32B.

C. Block Size Variation

Figure 6 depicts the results achieved for the L1 instruction cache with modifications in block size while maintaining associativity and cache size. It is observed that the Miss Rate decreases with increasing block size. Some benchmarks, however, yielded no results for these settings.

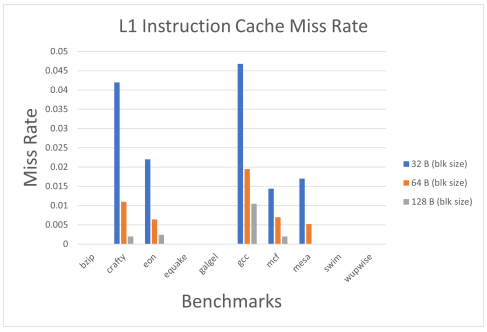

Fig. 6. Miss Rate for L1 Instruction Cache with change in block size setting constant cache size of 512KB and associativity as 1-way.

D. Instruction Per Cycles

The simulation results indicate that, for a particular benchmark, the ICP is the same across all setups (L1, L2, and TLB caches). However, for a given configuration, ICP varies with respect to the benchmark in Figure 7. The change in ICP is related to the difference in Miss Rate for each benchmark.

E. Cycles Per Instruction

The simulation results demonstrate that, for a particular benchmark, CPI is the same across all setups (L1, L2, and TLB caches). However, for a given configuration, CPI varies with respect to the benchmark in Figure 8.

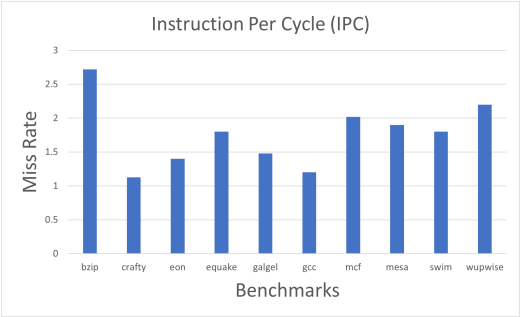

Fig. 7. IPC value for every Benchmarks

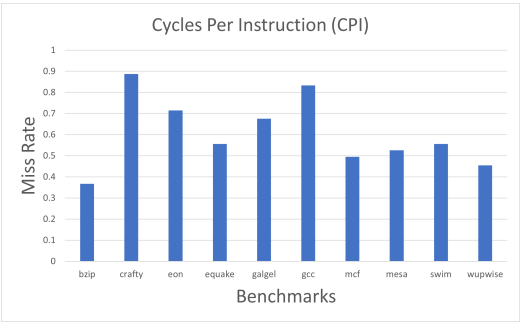

Fig. 8. CPI value for every Benchmarks

F. Translation Lookaside Buffer

Figures 9 and 10 show that the Hit Rate (number of hits/ number of accesses) is 100 % for all setups on all benchmarks and is unaffected by cache size. The number of misses in the results is insignificant. Result demonstrates that the TLB cache may provide a higher data locality and increase system performance. It is also seen that the number of accesses and hits for the TLB instruction cache is higher than that of the TLB data cache. Therefore, we can say that the system does more instruction address translations than data address translations.

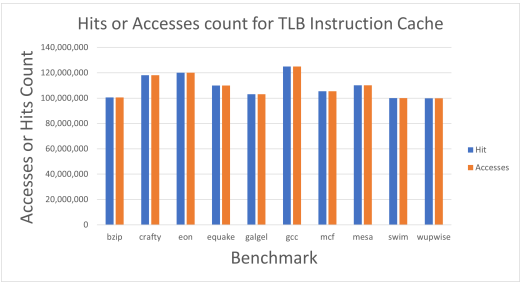

Fig. 9. Hit Count and Accesses for TLB Instruction Cache

G. Load/ Store Instructions

Figure 11 shows that the load/store instructions are maximum in the eon benchmark while benchmark galgel has the minimum value.

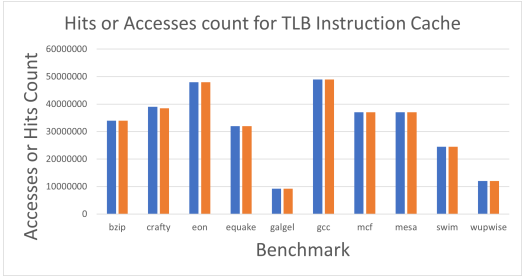

Fig. 10. Hit Count and Accesses for TLB Data Cache

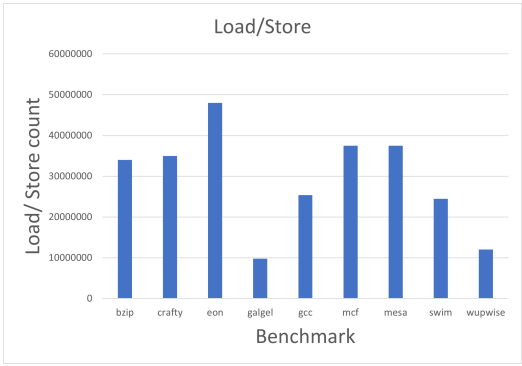

Fig. 11. Load/ Store Count in benchmarks.

V. Conclusion

The performance impact of several basic cache configuration parameters, such as the L1, L2, and TLB cache size, associativity, and block size have been evaluated using the SimpleScalar ”sim-outorder” model and the SPEC 2000 benchmark suite. The results illustrate the relationship between Miss Rate and modifications in cache size, associativity, and block size. The results also reveal the impact of the multi-level cache design as well as the efficacy of the TLB cache in enhancing data locality.

Acknowledgment

The authors would like to thank Dr. Xin Fu.

References

[1] D. Burger and T. M. Austin, “The simplescalar tool set, version 2.0,” *ACM SIGARCH computer architecture news*, vol. 25, no. 3, pp. 13–25, 1997.

[2] J. L. Hennessy and D. A. Patterson, *Computer architecture: a quantitative approach*. Elsevier, 2011.