

Memory Hierarchy Model Validation

Yan Solihin, Yong Luo, Kirk Cameron

Los Alamos National Laboratory

Abstract

The memory hierarchy model is a powerful tool that can separate the stall time in a multi-level cache. However, since no tool currently exists that is able to compute this information, it is difficult to validate the model. In this respect, a processor simulator can help. With a simulator, we can directly observe the stall time due to cache misses, and thus obtain cpi_0 .

We successfully validate the memory model with the aid of the simplescalar simulator. The predicted $t2$ and tm are within 10% of the values output by the simulator.

1 Introduction

2 Validation Method

To validate the memory hierarchy model [3, 7] we used and modified the simplescalar processor simulator [1]. First, we made modifications to simulate the MIPS R1000 processor. The modification largely follows that made by Daniel Citron [2]. Second, we inserted instrumentation to count the number of cycles stalled due to data cache misses.

Since the R10000 is a superscalar processor, such that the processor is able to hide some cache miss latencies, we need a special method of calculating this stall time. The method that we use is to calculate the stall cycles at the commit stage is the same method used by the RSIM processor simulator [5]. If during commit the processor cannot commit as many instructions as the commit width, we can observe which instruction cannot be committed. If the instruction is a memory instruction (load or store) and it has caused a cache miss¹, we increase the stall cycle by the number of wasted commit slots:

$$stall = \frac{committed}{commitwidth} \quad (1)$$

For example, if the commit width is 4 and during that cycle 2 instructions preceding a stalled memory instruction can commit, we increase the stall cycle by 2/4. Furthermore, we categorize the stall cycles into stalls due to L1 cache miss ($L1stall$), and stalls due to L2 cache miss ($L2stall$).

Since we also know the number of cycles ($cycles$), graduated instructions ($inst$), misses in the L1 ($L1miss$) and L2 cache ($L2miss$), we can use these to calculate cpi_0 , $t2$, and tm :

$$cpi_0 = \frac{cycles - L1stall - L2stall}{inst} \quad (2)$$

$$t2 = \frac{L1stall}{L1miss - L2miss} \quad (3)$$

¹it is possible a memory instruction cannot be committed even when it hits into the L1 cache, for example due to memory port being busy.

$$tm = \frac{L2stall}{L2miss} \quad (4)$$

There are a few things to note: first, we ignore the instruction cache misses in our calculations. This is not a problem since instruction cache misses account for a very small portion of total cache misses in scientific codes [6]. Second, t_2 and t_m may be larger than isolated cache miss latencies (T_2 and T_m) due to the effect of TLB misses. For example, although a hit on the L2 cache should only take a few cycles, for a TLB miss the penalty may be as high as 100 cycles.

The accuracy of the method is demonstrated by calculating t_2 and t_m of `lmbench` [4] and comparing them again the specified parameters of the simulator. `lmbench` is basically a microbenchmark that accesses array elements in a specified stride. By controlling the size of the array and the stride, we can set `lmbench` to produce a cache miss on every access. Thus, we can predict the cache miss penalty in isolation (T_2 and T_m) and compare these with the actual T_2 and T_m supplied as parameters to the simulator. Table 1 shows the numbers calculated using equation 4 for the simulator output and the actual T_2 and T_m ². Overall, the numbers are very close, with 4.3% error for T_2 and 1.2% error for T_m , which is mostly caused by ignoring instruction misses in the calculation. Thus, t_2 and t_m measurement by the simulator is reasonably accurate for our purposes.

Parameters	Calculated	Actual
t_2	11.35	11.86
t_m	81.68	80.68

Table 1: Accuracy of the validation method.

The model first predicts cpi_0 , then t_2 and t_m , using separate techniques. Our purpose is to validate the t_2 and t_m prediction. To do this, we first use the average value of cpi_0 as a constant cpi_0 as problem size increases to predict t_2 and t_m . Then we compare the predicted values with the values output by the simulator.

3 Validation Results

Three Los Alamos scientific applications: `sweep`, `hydro`, and `hydro-t` are validated. The results are shown in figure 1 and in table 2. The figure and the table show that the predicted t_2 and t_m are very close to the values computed by the simulator. Thus, we have validated the memory hierarchy model.

4 Experiment Condition

To get the most accurate t_2 and t_m prediction, the applications must have sufficiently large data set size ³. In addition, a loop base application needs large number of iterations. The data set size of the applications must overflow the L2 cache so that we get steady values for t_m , while the number of iterations must be large enough for the value of cpi_0 to converge.

²Supplied parameters are $T_2 = 11$, $T_m = 80$. The numbers in Actual are somewhat different because we take into account array access wraparounds and TLB misses

³defined as the size of the working set of the applications, sometimes referred to as problem size.

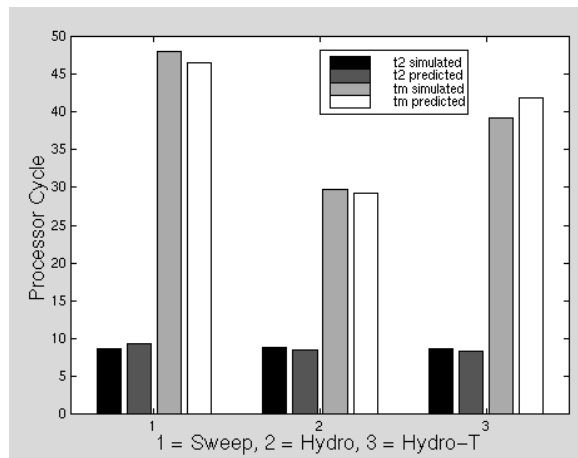


Figure 1: Validation Result

	Sweep		Hydro		Hydro-t	
Parameter	Computed	Predicted	Computed	Predicted	Computed	Predicted
t2	8.5669	9.2890	8.7626	8.3906	8.5918	8.3602
tm	48.0369	46.5925	29.7070	29.3166	39.2415	41.8336

Table 2: t2 and tm prediction

5 Conclusion

We have presented validation of the memory hierarchy model. Using the average value of cpi_0 , we can achieve t2 and tm predictions that are within 10% of the average computed values of t2 and tm output by the processor simulator. We have also validated our t2 and tm computing method using the simulator. We also suggested what conditions are optimal for the accuracy of the model.

References

- [1] Doug Burger and Todd Austin. The simplescalar toolset, version 2.0. Technical Report TR 1342, Computer Science Department, University of Wisconsin-Madison, 1997.
- [2] Daniel Citron. Private communication.
- [3] O. Lubeck, Y.Luo, H. Wasserman, and F. Bassetti. An empirical hierarchical memory model based on hardware performance counters. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, July 1998.
- [4] Larry McVoy. Lmbench: Portable tools for performance analysis. Technical Report unknown, Silicon Graphics, Inc.
- [5] V. Pai, P. Ranganathan, and S. Adve. The Impact of Instruction-Level Parallelism on Multi-processor Performance and Simulation Methodology. In *Proceedings of the Third International Symposium on High-Performance Computer Architecture*, pages 72–83, February 1997.

- [6] Yan Solihin. An application scalability model for distributed shared memory machines. Master's thesis, University of Illinois at Urbana Champaign, 1999.
- [7] H. Wasserman, O. Lubeck, Y. Luo, and F. Basseti. Performance evaluation of the sgi origin2000: A memory-centric characterization of lanl ascii applications. In *Proceedings of Supercomputing 1997*, November 1997.