A Factorial Performance Evaluation for Hierarchical Memory Systems

Xian-He Sun[†*] Dongmei He[†]

[†]Department of Computer Science
Louisiana State University
Baton Rouge, LA 70803-4020
{sun,dhe}@bit.csc.lsu.edu

Kirk W. Cameron[†‡] Yong Luo[‡]

[‡]Mail Stop B256
Los Alamos National Laboratory
Los Alamos, New Mexico 87545
{kirk,yongl}@lanl.gov

Abstract

In this study, we introduce an evaluation methodology for advanced memory systems. This methodology is based on statistical factorial analysis. It is two fold: it first determines the impact of memory systems and application programs toward overall performance; it also identifies the bottleneck in a memory hierarchy and provides cost/performance comparisons via scalability analysis. Different memory systems can be compared in terms of mean performance or scalability over a range of codes and problem sizes. Experimental testing has been performed on Department of Energy's Accelerated Strategic Computing Initiative (ASCI) machines and benchmarks available at the Los Alamos National Laboratory to validate this newly proposed methodology. Experimental and analytical results show this methodology is an effective tool for memory system evaluation and design.

1 Introduction

Various advanced memory systems have been developed to manage the increasingly wide disparity between central processing unit (CPU) speed and data access speed. Performance evaluation of these hierarchical memory systems, however, is very challenging. There are a variety of means by which the impact of memory latency on computer performance can be diminished by the computer architecture, as described in [1]. In addition, the performance and optimization requirements vary with problem sizes.

Based on statistical factorial analysis and performance scalability analysis, in this paper we propose a methodology for examining the effectiveness of both hardware and software memory latency hiding techniques of a memory system. This methodology consists of four levels of evaluation. For a set of codes and a set of machines, we first determine the effect of code, machine, and code-machine interaction on performance respectively. If a main or interaction effect exists, then, in the second level of evaluation, the code and/or machine is classified based on certain criteria to determine the cause of the effect. The first two levels of evaluation are designed to detect the characteristics of codes and their influence on different memory systems. They are based on average performances over the ranges of problem size interested in. The last two levels of evaluation determine the performance variation when problem sizes scale up and are based on scalability analysis which is a new approach for memory system evaluation. Level three evaluation is the scalability evaluation of the underlying memory system for a given code. Level four evaluation conducts a more detailed examination on the component contribution of the memory system toward the final scalability. The combination of the four levels of evaluation makes the proposed methodology adaptive, effective, and more appropriate for advanced memory systems than existing methodologies.

The Silicon Graphics Inc. (SGI) Origin2000 system and a previous SGI architecture, the PowerChallenge system, have been used as the test-bed to illustrate the newly proposed methodology. The single-processor performance (in terms of cycles per instruction,*cpi*) of the two machines are compared and analyzed. Evaluation results given by this methodology are confirmed by measured results and by a previously-reported performance model. The comparison of these two machines is of particular interest because they both use the same compute node, a 200-MHz MIPS R10000 processor [2] [3], but the memory subsystems of the two architectures are vastly different. Although improvement in the Origin memory network has important consequences for

* This author was supported in part by NSF under grant ASC-9720215, by LSU under 1998 COR award, and by Louisiana Education Quality Support Fund.

system-wide, multiprocessor performance, Origin single-node performance benefits as well.

We use a benchmark set consisting of applications from the Los Alamos portion of the ASCI workload. This set has five codes. They are HEAT, SWEEP, DSWEEP, HYDRO, and HYDROT. We give a brief introduction of the method and present some of the implementation results in the following two sections. Interested readers should refer to [4] for detailed information.

2 A Methodology for Hierarchical Memory Systems

We have developed a hierarchical evaluation methodology for advanced memory systems. This methodology consists of four levels of evaluation. All of the four levels of evaluation are based on two-factor factorial and regression statistical methods as given in [4, 5].

Level One : Main Effect
Level one evaluation uses a two-factor factorial experiment to find the effects of code and machine. Using the two factors code and machine, it detects the overall effect of code, machine, and their interaction on the final performance. The dependent variable for the two-factor factorial design is *cpi*. The random samples for each of the code-machine level combinations are chosen from different problem sizes within the interested problem size range. So, the effective comparison is based on the mean performance over different problem sizes. If code effect exists, we conclude that the codes have different memory reference patterns which diverge memory access time. When machine effect exists the memory system difference on the machines does make a difference in performance. Finally, when code-machine interaction effects exist the memory system difference has a different impact on different memory reference patterns. Notice that all these effects are overall effects of codes and machines. Any of the effects that exist deserve further investigation to identify the source or sources.

Level Two: Code/Machine Classification
Level one evaluation detects the overall effect of code, machine, and their interaction on performance. When these effects exist, we would like to know the contribution of each code/machine toward the effects and to identify the outstanding code/machine for more detailed study. The key technique to single out outstanding contributors is to find the relative performance of a code/machine with that of others. Statistical classification methods provide a means to group code/machine based on their relative performance.

The Contrast method and Post Hoc comparisons [4] are classical statistical methods for classification. We have used the contrast method and all the four Post Hoc methods in our study. These methods have different classification criteria. If two machines belong to the same category, then statistically they are the same, for the set of codes and under the interested range of problem sizes. If two codes belong to two different categories, then they have different memory reference/computation patterns. A good general purpose machine should not deliver a wide cpi distribution among codes.

Level Three: Scalability Comparison
The basic statistical method for memory scalability evaluation is the regression method given in [4]. The two factors are problem size and machine. The regression method does not measure data scalability directly, for which a formal quantitative definition of scalability is required. Instead, it gives a statistical relative comparison of two or more machines for a given code. Problem size increase may change the performance of a code-machine combination. This change varies with code, machine, and code-machine combination. It forms the base of scalability comparison. Using cpi as the measurement, with the same code on two different machines, if the interaction of the two variations is negative then the second machine has a better scalability; if the interaction of the two variations is zero then the two machines have the same scalability; otherwise, the first machine has a better scalability.

Level Four: Memory Hierarchy
The performance of a code may vary with problem size and the variation is different over different memory architectures. The last step of our evaluation methodology is designed to locate memory components which cause the variation. Level four evaluation compares the performance variation of primary components of the underlying memory systems. Combined with the level two evaluation, this evaluation determines the ability of each memory component in handling different memory reference patterns and suggests possible improvements at the component level.

The basic statistical method used in level four evaluation is the same as that of level three evaluation, except for the dependent variables. The actual design of level four evaluation varies with the underlying memory structure. The memory hierarchy of SGI PowerChallenge and Origin2000 has four primary components: $L1$ cache, $L2$ cache, *outstanding cache misses*, and main memory. $L1$, $L2$ hit ratio can be derived using hardware counters provided on-board the SGI microprocessor. For this reason we choose $L1$ and $L2$ as the dependent variables.

3 Evaluation of SGI PowerChallenge and Origin2000

In our experimental testing, the two machines, PowerChallenge and Origin2000, are assigned machine level 1 and level 2, respectively. The five codes, HEAT, HYDRO, SWEEP, DSWEEP, and HYDROT, are assigned a level value of 1, 2, 3, 4, 5, respectively. We have used the SAS solving environment [6] throughout the experimental evaluation.

The problem sizes used in the experiment range from N=50 to memory/time constraints. The corresponding range for the codes are: HEAT = [50, 100], HYDRO = [50, 300], SWEEP = [50, 200], DSWEEP = [50, 200], HYDROT = [50, 300]. All the experimental data are measured from single node sequential executions using SGI hardware performance counters.

3.1 Main and Interaction Effects

The relationship between code and machine is first investigated. To catch the mean relationship over the interested range of problem sizes, replicate measurements have been taken for different problem sizes for a given experimental unit. The GLM procedure of SAS is used to carry the two-factor factorial experiment for level one evaluation. Table 1 shows results from GLM.

Table 1. Mean Effects Table

Dependent	Variable:	cpi		
	Sum of	Mean		
Source	Squares	Square	F Value	$Pr > F$
Model	112.541	12.5046	27.44	0.0001
Error	46.9437	0.4558		
Total	159.4847			
R-Square	C.V.	Root MSE	cpi Mean	
0.7057	34.6445	0.6751	1.9487	
Source	Type I SS	MS	F Value	$Pr > F$
Machine	14.3956	14.3956	31.59	0.0001
Code	93.179	23.2947	51.11	0.0001
M*C	4.9664	1.2416	2.72	0.0334

Table 1 is the mean effects table of the factorial experiment. It consists of two sectors separated by the double-line. The upper table is for overall effect and the lower table is for individual effects. Look at row four of Table 1. The F value is 27.44 and the probability of F $(Pr > F)$ is 0.0001. The probability of F is less than 0.05. The hypothesis of overall-effect does not exist is rejected. This means that code or machine effects exist. The lower table is a continuation of the upper table to locate the potential effects. Look at row two of the lower table. The probability of F is $0.0001 < 0.05$, which suggests that machine main effect exists. The same conclusion can be drawn for code. For machine and code interaction, the probability of F is 0.0334, which is again smaller than 0.05. Interaction effect for code and machine also exists. Evaluation should be continued to understand these effects.

3.2 Scalability Comparison

Using the regression method discussed in Section 2, we have conducted scalability comparisons on all of the five codes over the two machines. Recall that this third step in our methodology compares the data scalabilities of a given code on different machines whereas the level two evaluation grouped codes based on their average performance over the range of problem sizes. As we discussed in the previous section, a better memory system should lead to a smaller cpi, and a more scalable memory system should have a smaller cpi increase, or no cpi increase at all as problem size scales up. The procedure PROG REG of SAS is used for the scalability comparison. The response variable is *cpi*. Table 2 is generated by PROG REG for the scalability comparison of HEAT over problem size range [50,100].

Table 2. Scalability Comparison of HEAT

	Parameter	Standard	
Variable	Estimate	Error	$Prob > \|T\|$
INTERCEP	2.4532	0.0507	0.0001
CODE	0.0776	0.0160	0.0001
MEMORY	-0.4683	0.0506	0.0001
INTAC	0.0795	0.016	0.0001

In Table 2, the "INTAC" stands for INTerACtion effect. Recall that the probability to test whether an interaction is zero is 0.05. At the 0.0001 level (see last column of Table 2), the hypothesis of zero effect has been rejected. The interaction effect exists. The parameter estimate of "INTAC" is 0.0795, which means that the performance difference of the two machines decreases with problem size. PowerChallenge is more scalable than Origin2000 over the range of problem sizes. This reduction in difference is very reasonable. When problem size increases into main memory, the advantage of having a larger L2 cache fades away. The performances of the two machines, therefore, become closer. Different codes have different memory access/computing ratio and have different memory reference patterns. Some codes have good locality, some do not. Some memory reference patterns can take advantage of the underlying memory

support, some cannot. These factors and others give codes different scalabilities on different memory systems. While the resulting table is not shown, HYDRO has an INTAC probability level of 0.0111 indicating interaction effects exist for HYDRO. Unlike HEAT, for HYDRO, the parameter estimate is $-0.050885 < 0$, which means that the performance difference between the two machines increases with problem size. Origin2000 has a better scalability than PowerChallenge for HYDRO. The scalability improvement may be due to Origin2000's larger L2 cache or hardware support in handling cache misses or faster memory access time. The results of code SWEEP, DSWEEP and HYDROT are different. The probabilities for rejecting zero interaction effects for these codes are larger than 0.05. Our no-effect hypotheses stands. The more advanced memory system of Origin2000 does not improve the performance difference of these three codes when problem sizes scale up. The relative performances over the two machines remain unchanged.

Table 3 lists results generated by PROG REG for scalability analysis of SWEEP. From Table 3, the probability level of interaction effect is 0.2216, which is greater than 0.05. Therefore, SWEEP has the same scalability on the two machines. For DSWEEP and HYDROT, the probability level of interaction effect is 0.3002 and 0.2799 respectively.

Table 3. Scalability Comparison of SWEEP

	Parameter	Standard	
Variable	Estimate	Error	$Prob > \|T\|$
INTERCEP	1.6135	0.0265	0.0001
CODE	0.0494	0.0097	0.0003
MEMORY	-0.3901	0.0265	0.0001
INTAC	0.0125	0.0097	0.2216

3.3 Evaluation of Memory Components

The memory systems of the SGI machines consist of four primary components: L1 cache, L2 cache, outstanding cache misses, and main memory. In the level four evaluation we examine the role of the four components in scalability variation. The same regression method used in scalability study is used here. We use SAS procedure PROC REG to evaluate the relative performance of L1 and L2 cache independently. The response variable is the cache hit ratio of L1 and L2 accordingly. The cache hit ratios of L1 and L2 are independent of each other and can be used as independent variables. Outstanding cache misses cannot be measured. However, based on the scalability comparison given in the previous section, its role in performance variation can be estimated when the variations of L1 and L2 hit ratio are known.

Table 4 is the L2 hit-ratio analysis table for HYDRO. As given in Table 4, the null hypothesis of interaction is accepted. The hit ratio differences of HYDRO remain the same for the SGI machines when problem size scales up. As analyzed in Section 3.2, HYDRO-Origin2000 has a better scalability than HYDRO-PowerChallenge. This scalability increase is not due to the larger L2 cache of Origin2000 as shown by the cache hit ratios across machines. It is due to the outstanding cache misses ability and faster main memory access time supported by Origin2000.

Table 4. L2 Hit-Ratio Comparison for HYDRO

	Parameter	Standard	
Variable	Estimate	Error	$Prob > \|T\|$
INTERCEP	0.9116	0.0094	0.0001
CODE	-0.0115	0.0021	0.0001
MEMORY	0.0463	0.0094	0.0001
INTAC	0.0039	0.0021	0.0771

Table 5 and 6 are the analysis table for L2 cache comparison for SWEEP and DSWEEP respectively. By Table 5, interaction effect exists for SWEEP and the effect is negative. The L2 hit ratio of SWEEP on Origin2000 becomes relatively smaller compared with that of PowerChallenge when problem size scales up. Since SWEEP has the same scalability on these two machines, the main memory contribution and/or the outstanding cache miss ratio must be improved on Origin2000 when problem size scales up. The outstanding cache-miss principle and faster main memory access also work well for SWEEP when problem size is large.

Table 5. L2 Hit-Ratio Comparison for SWEEP

	Parameter	Standard	
Variable	Estimate	Error	$Prob > \|T\|$
INTERCEP	0.8262	0.003	0.0001
CODE	-0.0132	0.0011	0.0001
MEMORY	0.0425	0.003	0.0001
INTAC	-0.0038	0.0011	0.0047

Like HYDRO, DSWEEP maintains a constant L2 hit-ratio difference on the two machines. DSWEEP has the same scalability on the two machines. When L1, L2 hit-ratio difference remain unchanged, the difference of main memory contribution toward the final performance is also unchanged [4]. Therefore, we can conclude that DSWEEP's

outstanding cache-miss ratio does not vary with problem size.

Table 6. L2 Hit-Ratio Comparison for DSWEEP

	Parameter	Standard	
Variable	Estimate	Error	$Prob > \|T\|$
INTERCEP	0.81	0.008	0.0001
CODE	-0.0328	0.004	0.0001
MEMORY	0.0779	0.008	0.0001
INTAC	0.0022	0.004	0.5966

Finally, Table 7 lists the L2 hit-ratio comparison for HYDROT. HYDROT has the same effect as SWEEP. Its L2 hit-ratio difference remains the same and has the same scalability on the two machines, as given in the previous section. Like SWEEP, HYDROT's outstanding cache-miss ratio does not change with problem size.

Table 7. L2 Hit-Ratio Comparison for HYDROT

	Parameter	Standard	
Variable	Estimate	Error	$Prob > \|T\|$
INTERCEP	0.919	0.0028	0.0001
CODE	-0.006	0.0006	0.0001
MEMORY	0.0237	0.0028	0.0001
INTAC	-0.0027	0.0006	0.0002

The four-level evaluation methodology proposed in Section 2 has been applied to analyze the performance of two ASCI machines and five benchmarks. It is interesting to note, that, despite the fact that all the codes had a better performance on Origin2000, by level three evaluation these codes have different relative performance variations over the two machines when problem size scales up. When problem size becomes large, the performance difference of HEAT on these two machines becomes smaller; the performance difference of HYDRO on these two machines becomes larger; while the differences of the other three codes remain unchanged. Obtaining the variation in relative performance is important for benchmarking and other performance comparisons. For instance, the scalability analysis shows that the relative performance of HEAT and HYDRO are more likely to vary with problem size than the other three codes. A more detailed evaluation, the level four evaluation, has found the causes of the scalability difference over the codes. In addition to a larger L2 cache capacity, the four outstandings for cache misses and the faster main memory access supported by Origin2000 have played an important role in performance improvement. This is especially true for HYDRO and SWEEP.

4 Conclusions

We have proposed a hierarchical statistic methodology for memory system evaluation. Unlike many existing statistic and stochastic methods, the newly proposed methodology is not designed to determine parameters of a preassumed performance model. Instead, it is built on the approach of relative performance comparison, which is one of the most important concerns in architectural design and algorithmic development, and is set to reach a balance of simplicity and effectiveness. The methodology compares the relative impact of codes, machines, codes and machines, and components of machines toward the final performance. It also compares the relative performance variation when problem sizes scale up, in terms of scalability. It is a post evaluation methodology. This newly proposed method can be used collectively with existing empirical and analytical models for quantitatively assessing the contribution of low-level system components toward the final performance.

References

[1] D. Burger, J. Goodman, and A. Kagi, "The declining effectiveness of dynamic caching for general-purpose microprocessors." Tech. Report CS-TR-95-1261, Jan. 1995.

[2] MIPS Technologies, Inc., "R10000 microprocessor product overview." MIPS Product Preview, 1995.

[3] K. Yeager, "The MIPS R10000 superscalar microprocessor," *IEEE Micro*, pp. 28–40, Apr. 1996.

[4] X.-H. Sun, D. He, K. Cameron, and Y. Luo, "An adaptive statistical methodology for advanced memory systems evaluation." Los Alamos national laboratory Unclassified technical Report (LAUR), No. 98-4176, 1998.

[5] R. Jain, *The Art of Computer System Performance Analysis*. John Wiley & Sons, 1991.

[6] SAS Institute Inc., *SAS User's Guide*. SAS Institute Inc., 1996.