LA-UR- 98-1130

Title: An Empirical Hierarchical Memory Model Based on Hardware Performance Counters

CONF-980747--

Author(s): Olaf M. Lubeck
Yong Luo
Harvey Wasserman
Federico Bassetti

Submitted to: Int'l. Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA '98)
Las Vegas, Nevada
July 13–16, 1998

Los Alamos
National Laboratory

An Empirical Hierarchical Memory Model Based on Hardware Performance Counters

Olaf M. Lubeck
Yong Luo
Harvey Wasserman
Federico Bassetti
oml@lanl.gov, yongl@lanl.gov, hjw@lanl.gov, fede@lanl.gov
Mail Stop B256
Los Alamos National Laboratory
Los Alamos, NM 87544

ABSTRACT
In this paper, we characterize application performance with a "memory-centric" view. Using a simple strategy and performance data measured by on-chip hardware performance coutners, we model the performance of a simple memory hierarchy and infer the contribution of each level in the memory system to an application's overall cycles per instruction (*cpi*). We account for the overlap of processor execution with memory accesses - a key parameter not directly measurable on most systems. We infer the separate contributions of three major architecture features in the memory subsystem of the Origin 2000: cache size, outstanding loads-under-miss, and memory latency.

Keywords: performance evaluation, cache, memory subsystem, computer architecture, and microprocessor

I. Introduction

The performance and scalability of high performance scientific applications on large-scale parallel machines are more dependent on the hierarchical memory subsystems of these machines than the peak instruction rate of the processors employed [1-2]. A few attempts [13, 14] have been tried to characterize the memory system performance impact on the total runtime. Many architecture improvements such as out-of-order execution and more outstanding misses are widely studied by simulations. However, studying the performance impact of memory subsystem and processor architecture improvements based on real applications on production machines is rarely attempted.

In this paper, we model real application performance with a "memory-centric" view. The applications and their realistic problem sizes are a representative part of the Los Alamos National Laboratory (LANL) computational physics workload and most have been designed with referential locality in mind. Using overall average effect strategy and empirical performance data from hardware performance counters, we infer the contribution of each level in the memory system to the application's overall cycles per instruction (*cpi*). We account for the overlap of processor execution with memory accesses - a key performance parameter that is not directly measurable on most systems.

Performance data on the application codes are obtained on the latest Origin 2000 systems and Power Challenge machines from SGI. This paper discusses only single node executions. The machines provide a unique performance evaluation opportunity since the architectures employ identical R10K processors but differ significantly in the design of the memory subsystems so that performance studies due solely to the memory architecture are possible. The same executables are used on both machines to eliminate software difference. In particular, there are three major memory architecture differences: 1) secondary cache size, 2) latencies to the main memory, and 3) number of outstanding cache misses. Thus, we are able to infer the separate contribution of each of these on the performance of the application benchmarks.

The following sections of this paper describe: the parts of the machine architecture relevant to this work, small descriptions of the codes from the Los Alamos National Lab computational physics workload, the model and empirical methodology, validation of the model, results, analysis and major conclusions.

II. Origin 2000 and PowerChallenge: Architecture Descriptions

The PowerChallenge is an SMP architecture that employs a central bus to interconnect memories and processors [3]. The bus bandwidth (1.2 Gbytes/sec) does not scale with more processors. Cache coherence is maintained through a snoopy bus protocol which broadcasts cache information to all processors connected to the bus. The Origin 2000, on the other hand, is a distributed shared memory (DSM) architecture which uses a switch interconnect that improves scalability by providing interconnect bandwidth proportional to the number of processors and memory modules [4]. Coherence is maintained by a distributed directory-based scheme. The processing elements of both the Origin 2000 and PowerChallenge systems use a 200MHz MIPS R10000 microprocessor. The processor is a 4-way super-scalar architecture which implements a number of innovations to reduce pipeline stalls due to data starvation and control flow [5]. For example, instructions are initially decoded in-order, but are executed out-of-order. Also, speculative instruction fetch is employed after branches. Register renaming minimizes data dependencies between floating-point and fixed-point unit instructions. The two programmable performance counters track a number of events [6] and were a necessity for this study.

While the processing elements of the PowerChallenge and Origin 2000 systems are identical, there are major differences in the memory architecture and corresponding performance of the two systems. The PowerChallenge is an UMA architecture with a memory latency of 205 clocks (1025 ns). Latencies to the memory modules of the Origin 2000 system, on the other hand, depend on the network distance from the issuing processor to the destination memory node. Accesses issued to local memory take about 80 clocks (400 ns) while latencies to remote nodes are the local memory time plus 33 clocks for an off-node reference plus 22 clock periods (CP; 110 ns) for each network router traversed. In the case of a 32 processor machine, the maximum distance is 4 routers, so that the longest memory access is about 201 clocks (1005 ns) which is close to the uniform latency of the PowerChallenge. This unique feature of Origin 2000 systems provides us a good opportunity to adjust the memory access latency by placing memory and execution thread on different nodes.

In addition, improvements in the number of outstanding cache misses that can be queued by the memory system were made. Even though the R10000 processor is able to sustain four outstanding primary cache misses, external queues in the memory system of the PowerChallenge limited the actual number to less than two. In the Origin 2000, the full capability of four outstanding misses is possible. The L2 cache sizes of these two systems are also different. A processor of PowerChallenge can be equipped up to 2MB L2 cache while a CPU of Origin 2000 system always has a L2 cache of 4MB.

III. LANL Benchmark Code Information

Four applications which form the building blocks for many nuclear physics simulations were used in this study. Previously, a performance comparison of the Origin and PowerChallenge architectures has been done using the codes [7].

a. Code Descriptions

SWEEP3D is a three dimensional solver for the time independent, neutral particle transport equation on an orthogonal mesh [8]. In SWEEP3D, the main part of the computation consists of a "balance" loop in which particle flux out of a cell in three Cartesian directions is updated based on the fluxes into that cell and on other quantities such as local sources, cross section data, and geometric factors. The cell-to-cell flux dependence, i.e., a given cell cannot be computed until all of its upstream neighbors have been computed, implies a recursive or wavefront structure. The specific version used in these tests was a scalar-optimized "line-sweep" version [8] that involves separately nested, quadrant, angle, and spatial-dimension loops. In contrast with vectorized plane-sweep versions of SWEEP3D, there are no gather/scatter operations and memory traffic is significantly reduced through "scalarization" of some array quantities. Because of these features, L1 cache reuse on SWEEP3D is fairly high (the hit rate is about 85%). A problem size of N implies N^3 grid points.

HYDRO is a two-dimensional explicit Lagrangian hydrodynamics code based on an algorithm by W. D. Schulz [9]. HYDRO is representative of a large class of codes in use at the Laboratory. The code is 100% vectorizable. An important characteristic of the code is that most arrays are accessed with a stride equal to the length of one dimension of the grid. HYDRO-T is a version of HYDRO in which most of the arrays have been transposed so that access is now largely unit-stride. A problem size of N implies N^2 grid points.

HEAT solves the implicit diffusion PDE using a conjugate gradient solver for a single timestep. The code was written originally for the CRAY T3D using SHMEM. The key aspect of HEAT is that its grid structure and data access methods are designed to support one type of adaptive mesh refinement (AMR) mechanism, although the benchmark code as supplied does not currently handle anything other than a single-level AMR grid (i.e. the coarse, regular level-1 grid only). A problem size of N implies N^3 grid points.

NEUT is a Monte-Carlo particle transport code. It solves the same problem as SWEEP3D but uses a statistical solution of the transport equation. Particles are individually tracked through a three dimensional mesh where they have some probability of colliding with cell material. The output from the particle tracking is a spatial flux discretized over the mesh. Vector (or data parallel) versions of this type of code exist which track particle ensembles rather than individual ones. A problem size of N implies N^3 grid points and 10 particles per grid point.

Based on the performance data collected through R10000 hardware performance counters, we calculated TLB hit ratio and branch prediction hit ratio. The calculation shows that MIPS R10000 processor can do a good job of speculative branch prediction. All four benchmark codes (HEAT, HYDRO, HYDRO-T and SWEEP) have branch prediction hit ratios over 99%. This means that over 99% of speculated branch predictions are taken in real executions. TLB hit ratios for all these codes are higher than 98%. This high TLB hit ratio implies that the impact of TLB misses can be ignored for these data sets.

IV. Model Description

The analysis in the following sections uses a simplified mean value parameterization [11] to separate CPU execution time from stall time due to memory loads/stores. Figure 1 is a pictorial description of the times in the model.

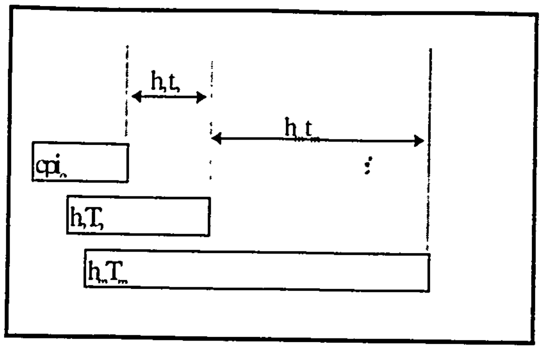

Figure 1. Relationship of modeled times

The model projects the overall *cpi* of an application as a function of CPU execution time and average memory access times:

$$cpi = cpi_0 + \sum_{i=2}^{nlevels} h_i * t_i \qquad (1)$$

where cpi_o is defined to be the *cpi* of the application assuming that all memory accesses are from an infinite L1 cache and take 1 CP (i.e., the i=1 term is included in cpi_o), and h_i and t_i are, correspondingly, the hits per instruction and average non-overlapped access times for the ith level in the memory hierarchy. The second term of Eq. 1 is also referred to as cpi_{stall}.

If no overlap of CPU execution and memory accesses occur, every memory access to the ith level incurs the full round-trip latency, which we denote as T_i. We define (following Larson [12]) a measure of the overlap of memory accesses with computation as m_0, where

$$cpi = cpi_0 + (1-m_0) \sum_{i=2}^{nlevels} h_i * T_i \qquad (2)$$

and, from Eq 1, m_0 is one minus the ratio of the average memory access time to the maximum memory access time:

$$m_0 = 1 - \frac{\sum_{i=2}^{nlevels} h_i * t_i}{\sum_{i=2}^{nlevels} h_i * T_i} \qquad (3)$$

We note here that the separation of computational time from memory access time in this model implies that the two can be treated independently (i.e., that cpi_0 is constant).

V. Measurements and Validation

a. Measurements

The model described in the previous section provides the foundation for an analysis of the Origin 2000's architectural features on application performance. The first key issue is determination of the amount of memory access time that is overlapped by computation. Although this overlap is not directly measurable using the R10000 performance counters, we can infer the overlap for an individual application by fitting empirical performance data obtained from its execution using different problem sizes.

R10000 performance counters supply measurements of the total execution cycles and total graduated instructions. The ratio of these two measurements gives the overall cpi of the application. The maximum latencies, T_i, are measured with LMBENCH [10] and are found to be consistent with numbers published by SGI. The hit ratios (coming from the same application executing on different problem sizes) are also directly measurable and the unknowns in Equation 1 become the average times, t_i, and cpi_0. The value of cpi_0 can be obtained by measuring the *cpi* of a problem that fits entirely in the L1 cache. The remaining unknowns are inferred from the measured data by a least squares fit constrained such that

$$0 <= t_i <= T_i,$$

Table 1 shows the model parameters for each of the LANL benchmark codes determined from a data set of executions on the 1-MB L2 PowerChallenge. The least square fit generally has errors that are less than 6%.

	t_2	t_m	cpi_o
HEAT	2	128	0.74
HYDRO	3	117	0.89
HYDRO-T	0	69	.9
SWEEP	11	134	.88
NEUT	2.2	205	.77

Table 1. Model parameters for each code (Power Challenge)

b. Validation

Validation of the inferred model parameters is accomplished using the model to predict performance on a different machine configuration. Original data from a PowerChallenge with a 1-MB secondary cache is used to determine the unknown model parameters which are then used to predict the performance of each code on a 2-MB PowerChallenge. Figure 2 shows that the fit is extremely close.

VI. Results and Analysis

a. Analysis of stall time due to memory accesses.

Table 2 compares the memory access times, t_i, for the benchmark codes on the Power Challenge and the Origin 2000. In general, L2 cache accesses are mostly overlapped with computation (low values of t_2). Additionally, the observed values of t_m suggest that about one-half of the main memory latency is hidden on both the Power Challenge and Origin. The exception is SWEEP where the value of 11cps for t_2 indicates that accesses to the secondary cache are not overlapped. The reason that SWEEP stands out may be due to loop-carried dependencies in the inner loops. These dependencies present less prefetch opportunities for the compiler and result in less overlap of processor execution with memory accesses. We believe that the model parameters for NEUT may be inaccurate. There is so little time associated with the memory accesses for NEUT (due to ~99% L1 cache hit ratio) that small

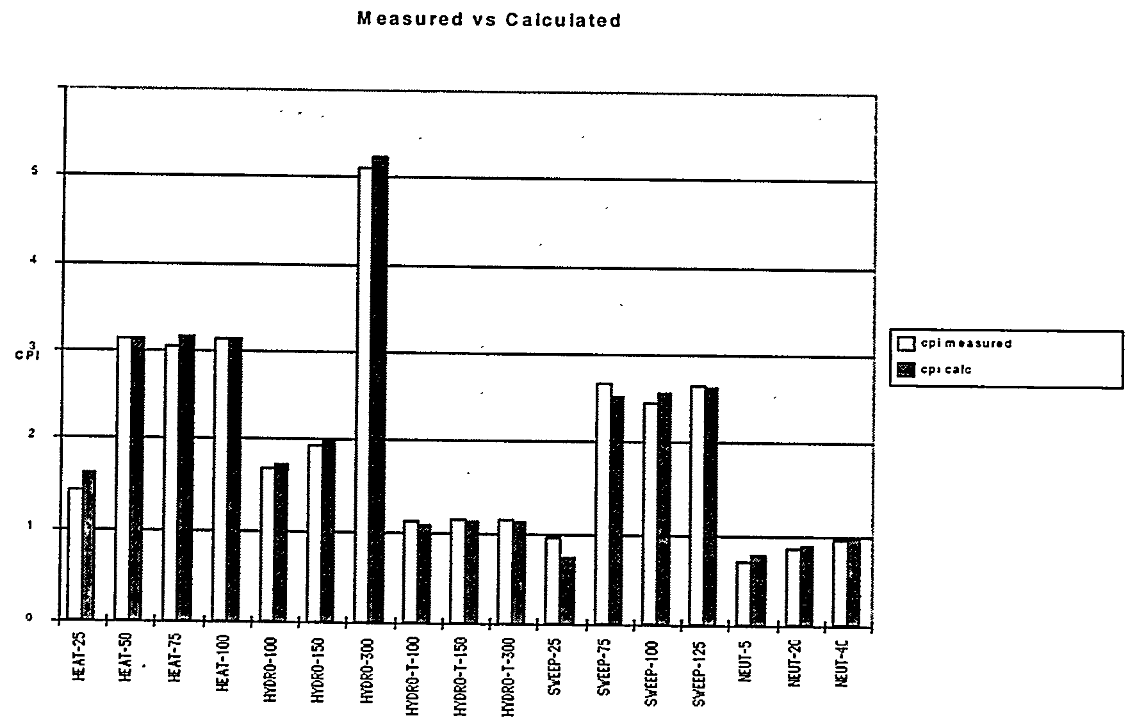

Figure 2. Model fit for codes with varying problem size

absolute least square errors can result in large relative changes to the parameters.

	t_2 PC	t_m PC	t_2 Origin	t_m Origin
HEAT	2	128	0	60
HYDRO	3	117	2.4	50
HYDRO-T	0	69	0	11
SWEEP	11	145	11	43

Table 2. Memory Access times, t_i

Figures 3 and 4 show graphs of cpi_{stall} relative to the overall cpi for both machines on each code. The second half of each figure shows the corresponding overlap parameter, m_0. A number of general observations are apparent from the graphs. The overall cpi on the Origin is typically less than that of the Power Challenge by factors of up to three (see also Luo, et al. [7]). The percentage of cpi represented by stall time on the Origin can be less than 40%, while, on the Power Challenge, it can be as large as 80%. Two codes, HYDRO-T and NEUT, exhibit high locality of reference and cpu stalls due to memory accesses are less than 10% of the total time. A study of the algorithms/implementations of these codes would lead one to expect this. NEUT has a modest number of scalar variables per particle that are used many times before another particle is computed (high temporal locality). HYDRO-T is a 2D code and was re-coded from the original HYDRO so that inner loops have stride-1 vectorizable loops (high spatial locality). The success of the transposition is readily seen by comparing the two versions in the figures. Memory overlap parameters are higher on the Origin than on the PowerChallenge which is indicative of the better latency hiding capability of the Origin.

Two extreme cases standout: HYDRO-T with very high overlap, and SWEEP, with very low overlap. The high spatial locality of HYDRO-T means that there is a great deal of parallelism between CPU and L1, L2 or main memory accesses. Additionally, on the Origin 2000, major portions of this 2-D algorithm fit entirely in the 4-MB L2 cache. In contrast, SWEEP shows much less overlap on either the Power Challenge or the Origin. This is consistent with the information in Table 2 in which we attributed to loop-carried dependencies. The results for NEUT, where the Power Challenge shows high overlap and the Origin shows very low overlap, are again

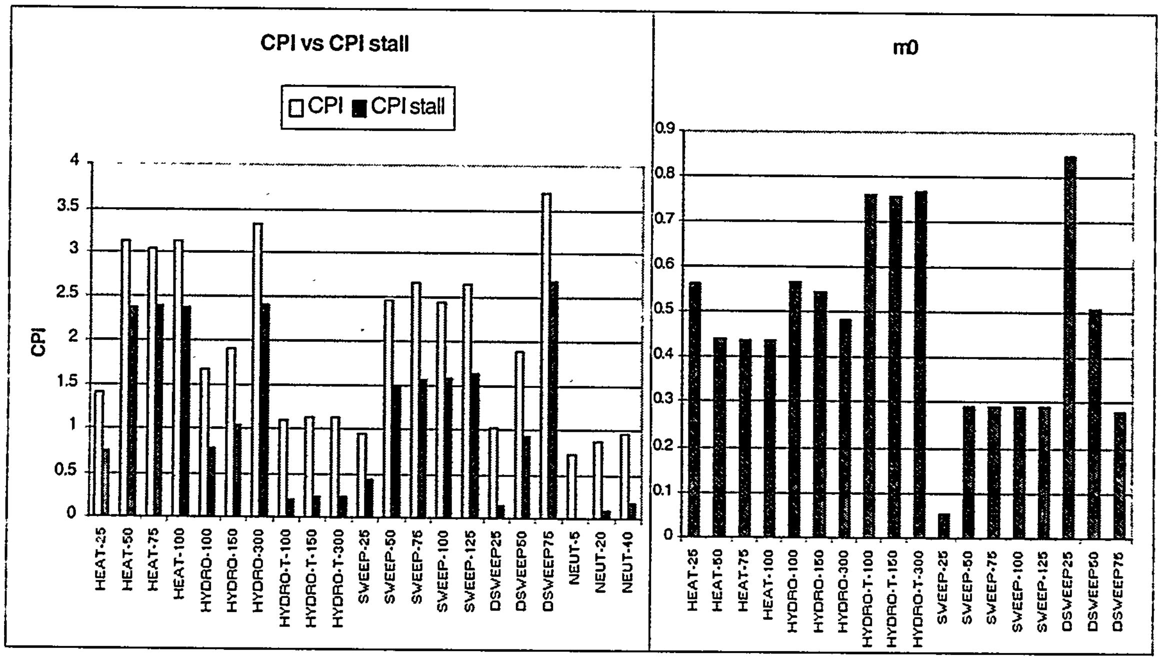

Figure 3. Memory stall & overlap parameters (Power Challenge)

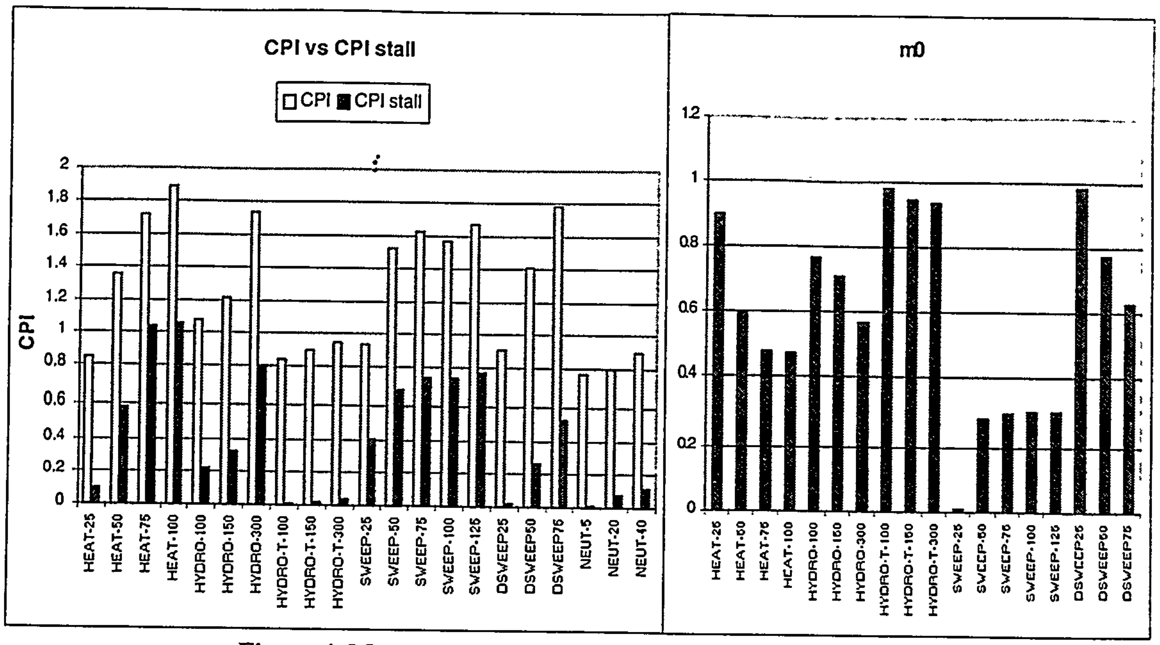

Figure 4. Memory stall & overlap parameters (Origin2000)

due to the large parameter changes associated with the least-squares fit mentioned above.

b. Separate contributions to the stall time.

As described in Section II, we can perform an experiment in which we systematically varied T_m, the latency to main memory seen by an executing thread, by placing the thread and its associated data on two different nodes of the Origin.

Using these experimental measurements and other empirical data on the two machines, we can infer the separated contribution of cache size, memory latency and number of outstanding misses to the improved *cpi* of the Origin over the Power Challenge. Let F be a measure of this overall improvement:

$$F = cpi^{PC} / cpi^{O} \qquad (5)$$

We wish to find the contributing factors, f_c, f_o, and f_m (corresponding to cache size, outstanding misses and memory latency, respectively) such that:

$$F = f_c * f_o * f_m . \qquad (6)$$

These factors can be defined as follows:

$$f_c = \frac{h_2^{PC} t_2^{PC} + h_m^{PC} t_m^{PC} + cpi_0}{h_2^{O} t_2^{PC} + h_m^{O} t_m^{PC} + cpi_0} \qquad (7)$$

$$f_O = \frac{h_2^{O} t_2^{PC} + h_m^{O} t_m^{PC} + cpi_0}{cpi^{O*}} \qquad (8)$$

$$f_m = \frac{cpi^{O*}}{h_2^{O} t_2^{O} + h_m^{O} t_m^{O} + cpi_0} . \qquad (9)$$

The denominator in f_c can be viewed as the *cpi* of a virtual machine whose characteristics are identical to those of the Power Challenge but with L2 cache size equal to that of the Origin (4MB L2). The larger cache size simply changes the hit ratios, h_i^{PC}, to h_i^{O}. Similarly, the denominator in f_o represents the *cpi* of a virtual machine identical to the Origin except for a memory latency equal to that of the Power Challenge. The cpi^{O*} for this machine is directly measured on the Origin by placing memory on a remote node so as to increase memory access latency to the same level of PowerChallenge. The

Code	f_c	f_o	f_m	F_{calc}	F_{obs}
HEAT50	1.47	1.40	1.13	2.34	2.37
HEAT75	1.02	1.58	1.09	1.76	1.80
HEAT100	1.00	1.54	1.12	1.73	1.68
HYDRO100	1.41	1.06	1.02	1.52	1.53
HYDRO150	1.34	1.08	1.09	1.59	1.47
HYDRO300	1.28	1.16	1.31	1.94	1.67
HYDRO-T100	1.16	1.05	0.99	1.21	1.28
HYDRO-T150	1.09	1.10	1.03	1.23	1.25
HYDRO-T300	1.01	1.12	1.08	1.22	1.21
SWEEP50	1.05	1.28	1.13	1.52	1.60
SWEEP75	1.00	1.18	1.27	1.50	1.63
SWEEP100	1.00	1.42	1.06	1.52	1.55

Table 3. Observed and calculated performance on the Origin2000

quantity, f_c, then, is the ratio of the actual Power Challenge to a Power Challenge with the Origin's cache. The quantity, f_o, is the ratio of this "larger cache" Power Challenge to an Origin with larger memory latency. Finally, the quantity, f_m, is the ratio of this "large latency" Origin to the real Origin. The separate factors satisfy the relationship in Eq.6.

Each of these factors is listed in Table 3, along with the calculated and observed values, F, for the codes. The calculated and observed speedups are in good agreement. With the exception of HYDRO and a small HEAT problem, the values of f_c are 1.0-1.1 indicating that the effect of a larger L2 cache is negligible. The values of f_m are also quite small (most showing10% or less improvement). The overall improvement for over half of benchmark codes comes from the increased number of outstanding misses on the Origin. About 75% of the total improvement of the larger HEAT problems and 50% to 80% of SWEEP come from this feature.

VII. Conclusions

This paper describes a methodology using a simple memory model with empirical parameters that accounts for the overlap of single processor execution with memory accesses. This method is applied to real applications using performance counter data available on actual machines. In general, this model quantifies the amount of overall time that is spent on memory accesses for each application. On the Power Challenge, the memory access time can be as large as 80% of the overall execution time. On the Origin 2000, the memory access time is less than 40%. Using this model, we discover that the increased number of outstanding misses in the Origin 2000 is a major contributing factor to the performance improvement in two out of four codes. The effect of cache size on the performance of these codes is generally much less important except for a code with poor cache reuse (HYDRO). Currently, the methodology is an excellent diagnostic tool that can provide information about the actual time that an application spends in memory accesses. This model can be eventually incorporated into a performance tool. The methodology can also be applied to any architecture with the necessary hardware counter information. Further work in our group is currently underway to model other microprocessors systems such as the IBM RS/6000 and Intel Pentium Pro. Future work will attempt to enhance the predictive capability of the model.

Acknowledgements

We thank Jim Schwarzmeier (SGI/Cray) who reviewed the paper and contributed many thoughtful suggestions for improvement and future work.

References

1. Wulf, W. A. and McKee, S. A. "Hitting the Memory Wall: Implications of the Obvious," Computer Architecture News, Association for Computing Machinery, March, 1995.

2. Burger, D. C., Goodman, J. R., and Kagi, A., "The Declining Effectiveness of Dynamic Caching for General-Purpose Microprocessors," Univ. Wisconsin Computer Sciences Tech. Report CS-TR-95-1261, Jan. 1995, and references therein.

3. Galles, M. and Williams, E., "Performance Optimizations, Implementation, and Verification of the SGI Challenge Multiprocessor," Silicon Graphics Computer Systems, ," Silicon Graphics Computer Systems, Mountain View, CA web paper http://www.sgi.com/Technology/challenge_paper.html.

4. Laudon, J. and Lenowski, D., "The SGI Origin: A ccNUMA Highly Scalable Server," Proc. Compcon Spring.1997, IEEE Computer Society, Los Alamitos, California.

5. (a) MIPS Technologies, Inc., "R10000 Microprocessor Product Overview," MIPS Product Preview, 1995. (b) Yeager, K. C., "The MIPS R10000 Superscalar Microprocessor," IEEE Micro, April, 1996, pp 28-40.

6. Zagha, M., Larson, B., Turner, S., and Itzkowitz, M., "Performance Analysis Using the MIPS R10000 Performance Counters," Proc. Supercomputing '96, IEEE Computer Society, Los Alamitos, California, 1996.

7. Luo, Y., Lubeck, O.M., and Wasserman, H. J., "Preliminary Performance Study of the SGI Origin2000," Los Alamos National Laboratory Unclassified Release LA-UR-97-334, 1997.

8. Koch, K. R., Baker, R. S. and Alcouffe, R. E., "Solution of the First-Order Form of the 3-D Discrete Ordinates Equation on a Massively Parallel Processor," Trans. of the Amer. Nuc. Soc., 65, 198, 1992.

9. W. D. Schulz, "Two-Dimensional Lagrangian Hydrodynamic Difference Equations," Methods in Computational Phys. Vol 3, p1, 1964.

10. McVoy, L. and Staelin, C., "lmbench: Portable Tools for Performance Analysis," http://reality.sgi.com/lm_engr/lmbench.

11. Vernon, M.V, Lazowska, E. D., and Zahorjan, J., "An Accurate and Efficient Performance Analysis Technique for Multiprocessor Snooping Cache-consistency Protocols," in Proc. 15th Annu. Symp. Comput. Architecture, Honolulu, HI, June, 1988, pp 308-315.

12. Larson, B., Silicon Graphics Computer Systems, private communication, January, 1997.

13. Bhandarkar, D. and Cvetanovic, Z., "Performance Characterization of the Alpha 21164 Microprocessor Using TP and SPEC Workloads," Proc. Second. Int. Sypm. on High-Perf. Comp. Arch., IEEE Computer Society Press, Los Alamitos Ca., 1996.

14. Bhandarkar, D. and Ding, J., "Performance Characterization of the Pentium Pro Processor, " Proc. Third. Int. Sypm. on High-Perf. Comp. Arch., IEEE Computer Society Press, Los Alamitos Ca., pp 288-297, 1997.