M98000349

CONF-980214--

LA-UR 97-3462

Los Alamos National Laboratory is operated by the University of California for the United States Department of Energy under contract W-7405-ENG-36

19980401 031

TITLE: **DEVELOPMENT AND VALIDATION OF A HIERARCHICAL MEMORY MODEL INCORPORATING CPU- AND MEMORY-OPERATION OVERLAP**

AUTHOR(S): OLAF M. LUBECK
YONG LUO
HARVEY J. WASSERMAN
FEDERICO BASSETTI

SUBMITTED TO: *Fourth International Symposium on High-Performance Computing Architecture, Las Vegas, Nevada, February 1-4, 1998*

Los Alamos

Los Alamos National Laboratory
Los Alamos New Mexico 87545

DISCLAIMER

Development and Validation of a Hierarchical Memory Model Incorporating CPU- and Memory-Operation Overlap

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 87545

ABSTRACT

Distributed shared memory architectures (DSM's) such as the Origin 2000 are being implemented which extend the concept of single-processor cache hierarchies across an entire physically-distributed multi-processor machine. The scalability of a DSM machine is inherently tied to memory hierarchy performance, including such issues as latency hiding techniques in the architecture, global cache-coherence protocols, memory consistency models and, of course, the inherent locality of reference in algorithms of interest. In this paper, we characterize application performance with a "memory-centric" view. Using a simple mean value analysis (MVA) strategy and empirical performance data, 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 parameter which is not directly measurable on the Origin 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, 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]. The view that anticipates real application performance will grow directly proportional to increases in processor speed is simplistic, even for single-processor systems.

Distributed shared memory architectures (DSM's) are being implemented which extend the concept of single-processor cache hierarchies across an entire physically-distributed multi-processor machine. Machines currently available to the Department of Energy's Accelerated Strategic Computing Initiative (ASCI), such as the Silicon Graphics, Inc. (SGI) Origin 2000, can be configured with 128 processors in a single DSM. Potential systems at Los Alamos National Laboratory may have 1000s of processors in a single cache-coherent shared address space. Scalability of these machines is inherently tied to memory hierarchy performance. This includes such issues as latency hiding techniques in

the architecture, global cache-coherence protocols, memory consistency models and, of course, the inherent locality of reference in algorithms of interest.

In this paper, we characterize application performance with a "memory-centric" view. The applications and realistic problem sizes are a representative part of the ASCI workload at LANL and most have been designed with referential locality in mind. Instruction-level simulation of even small problem sets would require at least 12-36 hours and we thus resort to experimental techniques and modeling to understand the effect of changes in major architectural parameters . Using a simple mean value analysis (MVA) strategy and empirical performance data, 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 parameter which is not directly measurable on the Origin systems.

Performance data on the ASCI codes are obtained on the latest Origin 2000 and two different configurations of Power Challenge machines from SGI. This paper discusses only single thread 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. In particular, the major memory parameters that we are able to vary independently among these machines are: 1) secondary cache size, 2) latencies to the main memory, and 3) number of outstanding loads-under-cache-misses. Thus, we are able to infer the separate contribution of each of these on the performance of the ASCI benchmarks. For example, two Power Challenges allow us to study cache size without changing any other architectural features. We also investigate the effects of different memory latencies (holding the other two parameters constant) by using memory placement directives on the Origin to locate an executing thread and its corresponding data set on different remote nodes at variable distances. One of these processor-remote memory pairs on the Origin matches the uniform memory latency of the Power Challenge allowing us to investigate the effect of different loads-under-misses holding memory latency constant.

The following sections of this paper describe: the parts of the machine architecture relevant to this work, small descriptions of the codes from the ASCI workload, the model and empirical methodology, validation of the model using a combination of measurement and simulation, 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. Figure 1 shows a network view of the machine. Each router in the hypercube topology connects two nodes to the network. Each node contains two processing elements and one local memory unit. A 128 processor system, for example, consists of a fifth-degree hypercube with 4 processors per router.

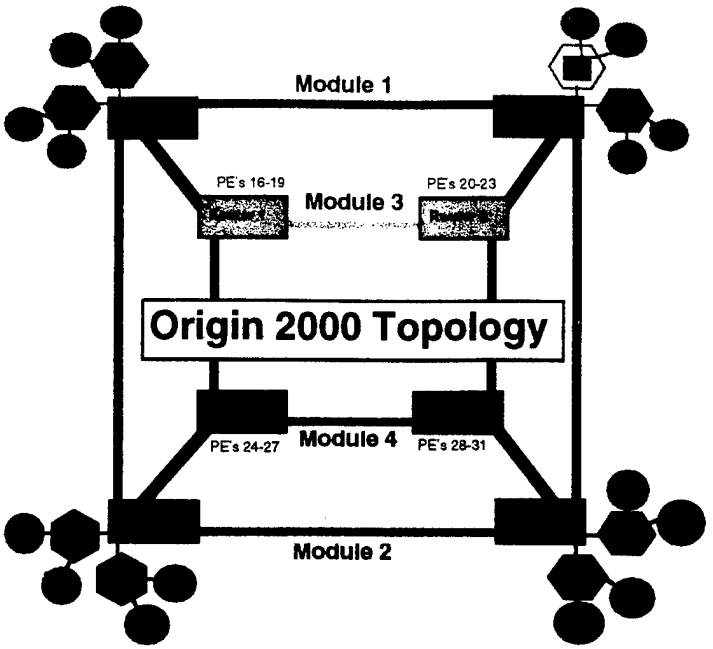

Figure 1. Origin 2000 Topology for a 32-processor system

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. Logical destination register numbers are mapped to the 64 integer and 64 floating point physical registers during execution. The two programmable performance counters track a number of events [6] and were a necessity for this study. The most common instructions typically have one- or two-clock latencies. The MIPS processor has been optimized for 64-bit floating point arithmetic, and integer multiply and divide operations take longer than their corresponding floating point instructions. Floating point load/store instruction latencies are 3 clocks while integer load/store latencies are 2 clocks. Both L1 and L2 caches are two-way set associative. The L1 line size is 32 bytes while the L2 line size is 128 bytes. The peak bandwidth is 16 bytes/clock.

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 a UMA architecture with a 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.

In addition, improvements in the number of outstanding loads 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.

III. ASCI Benchmark Code Information

Four applications which form the building blocks for many ASCI 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]. The first-order form of the transport equation is solved by sweeping through the spatial mesh along discrete directions (ordinates). 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[Koch] 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.

LMBENCH [McVoy & Staelin] is a micro-benchmark suite designed to measure latency and bandwidth of various levels of memory hierarchy. One LMBENCH kernel (mem_rd_latency) is adapted in our test code. With the –O3 optimization of SGI compilers (C & FORTRAN 77), the kernel part of this code can generate a series of (1000) load instructions. Each of these load instructions uses the result of the previous load as one of its operands, so every load instruction is dependent on the previous one. This code can be

used as a good validation tool for our model since its kernel has a known cpi_0 (see below), which is the pipeline latency of load instructions (the pipeline latency for integer load instructions is 2 cycles and 3 cycles for floating point loads).

b. Performance Characteristics

In this section we present some single-processor characteristics of the benchmark codes as obtained from performance counters on the Origin 2000. Table 1 shows two derived characteristics (averaged over all problem sizes) for all five codes. Note that the maximum MFLOPS observed may, in some cases, be obtained from unreasonably-small problem sizes relative to actual ASCI production runs; the data are presented here merely as a reference for the normalized Mflop curves in Figures 2-6.

Table 1 Code Characteristics

	HEAT	HYDRO	HYDRO-T	SWEEP	NEUT
Mem/Flops	2.59	1.34	1.33	1.49	1.95
Branch Rate (%)	5.40	3.77	3.56	4.91	8.46
Max. MFLOPS (Onyx)	21.9	14.1	36.0	51.0	44.6
Max. MFLOPS (Origin)	35.2	37.8	46.2	44.3	49.0

Mem/FLOPS is the ratio of memory references to floating point instructions and reflects the density of load/store instructions in a code. The results show the number of accesses is related to FLOPS by a small constant (greater than one) and the growth rate of both memory accesses and FLOPS is $O(n)$. The branch rate is the percentage of branches in all graduated instructions. HEAT has the highest Mem/Flops and NEUT has the highest branch rate among these benchmark codes. In HEAT, the high Mem/FLOPS ratio is due to gather/scatter memory accesses in the code. In NEUT, the high branch rate is indicative of the Monte Carlo method employed.

Detailed performance characteristic data for these codes were collected on a 1-MB L2 Power Challenge system and a 4-MB L2 Origin2000 system. Performance data as a function of problem size for the Power Challenge and Origin are illustrated in Figures 2 through 6. MFLOPS curves are normalized such that the maximum rate for each code is one.

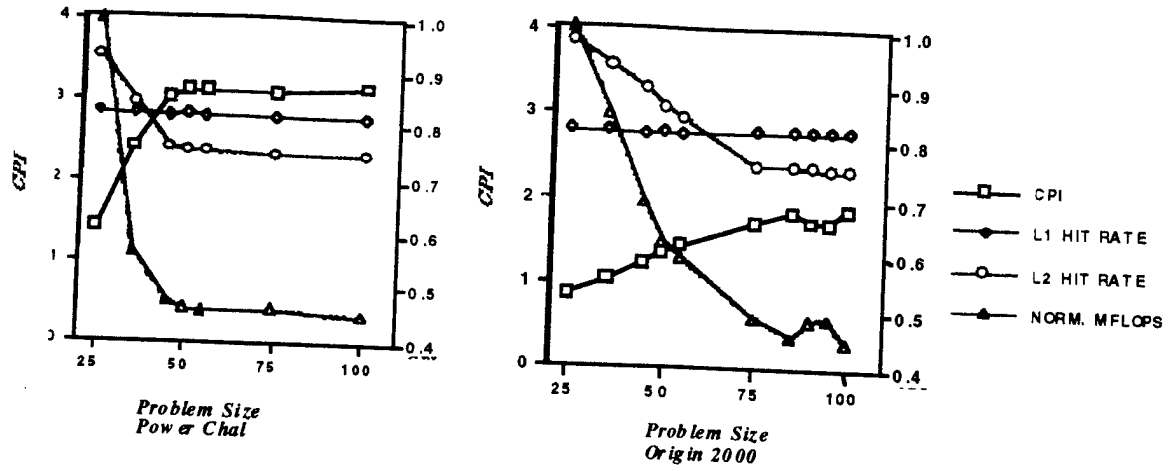

Figure 2. Performance of HEAT as a Function of Linear Problem Size. The right axis shows cache hit rates and normalized MFLOPS.

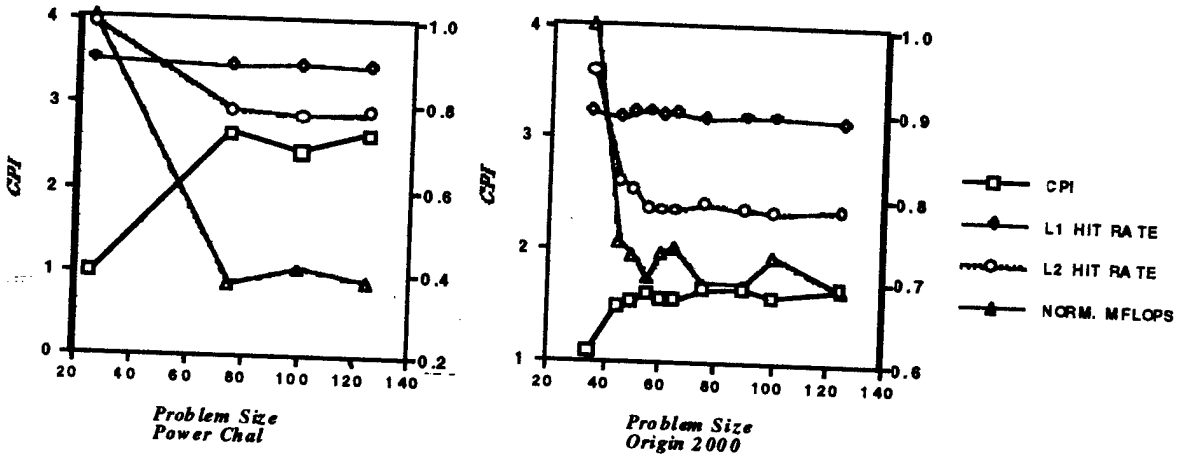

Figure 3. Performance of SWEEP as a Function of Linear Problem Size

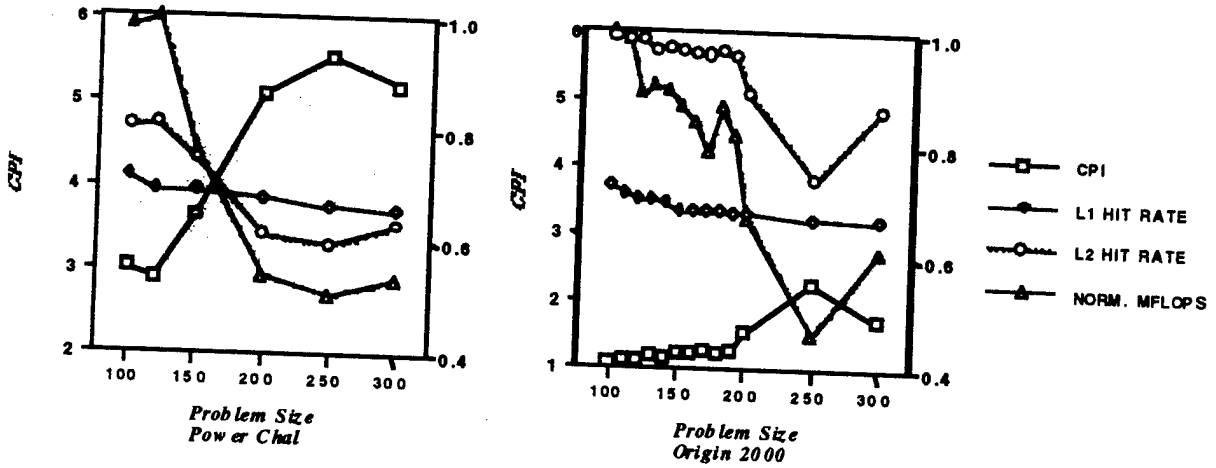

Figure 4. Performance of HYDRO as a Function of Linear Problem Size.

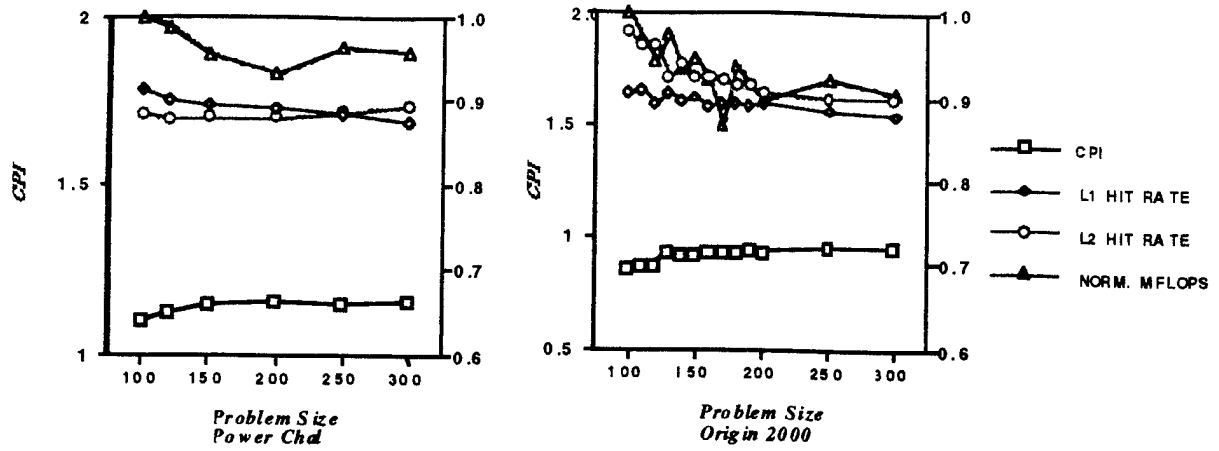

Figure 5. Performance of HYDRO-T as a Function of Linear Problem Size.

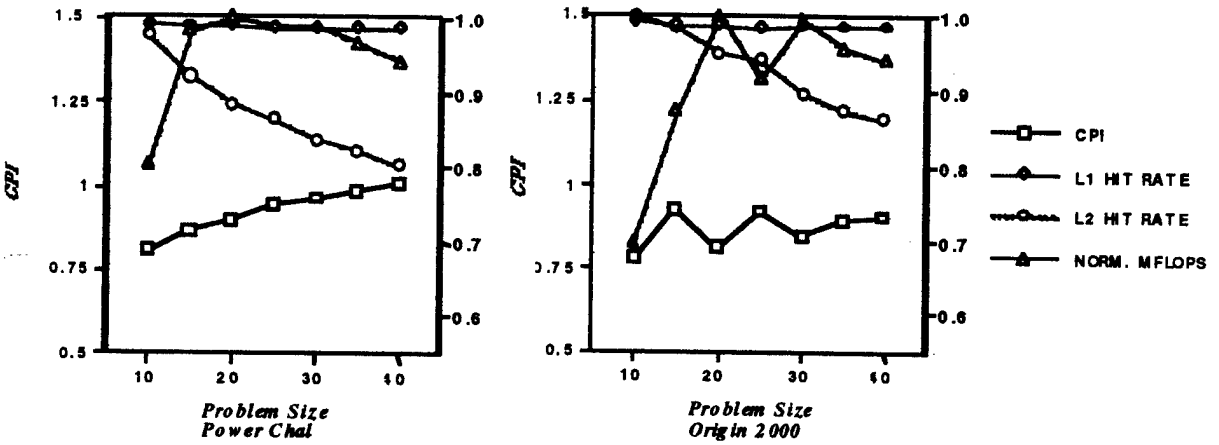

Figure 6. Performance of NEUT as a Function of Linear Problem Size.

The codes' overall cpi curves are generally the inverse of their corresponding MFLOPS curves; that is, an increasing cpi corresponds to a decreasing MFLOPS at nearly the same slope and vice versa. The cpi of three of the codes (HEAT, HYDRO and SWEEP) is strongly dependent on problem size.

The above figures show that normalized MFLOPS curves (except for HYDRO-T) follow the tendencies of the L2_hit curves. On the Power Challenge system, a drop in L2_hit rate causes much more impact to MFLOPS than it does on the Origin system. This is due to lower memory latency (both actual and effective) on the Origin2000 system. Although not shown in the figures, 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 7 is a pictorial description of the times in the model.

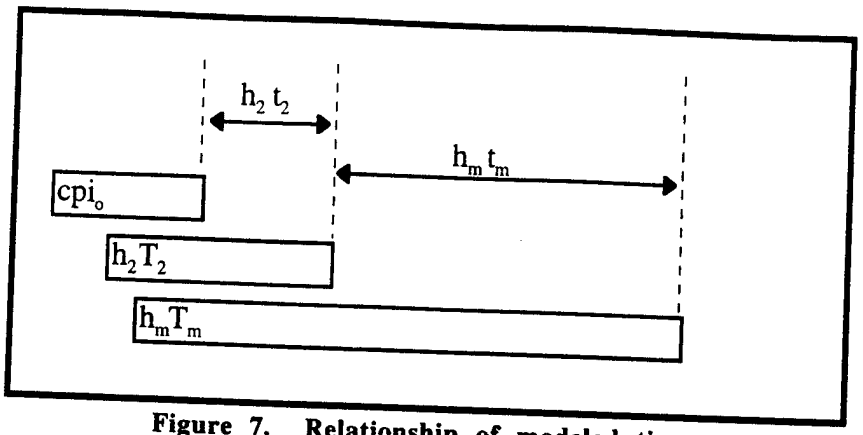

Figure 7. Relationship of modeled times

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). In fact, the out-of-order execution of the R10000 processor means that different dynamic instruction sequences will be seen for different size problems. The assumption that this effect is small is tested with an R10000 simulator in a later section.

The effect of increasing the round-trip memory latency to $T_m + dT_m$ is depicted in Fig. 8. Once the latency hiding ability of the architecture on a particular code has been exhausted, any additional main memory latency will simply add to the non-overlapped time t_m. In this case, the new cpi (from Eq. 1, where the sum is over the L2 cache and main memory) will be:

$$cpi' = cpi_0 + h_2 t_2 + h_m (t_m + dT_m) \qquad (4)$$

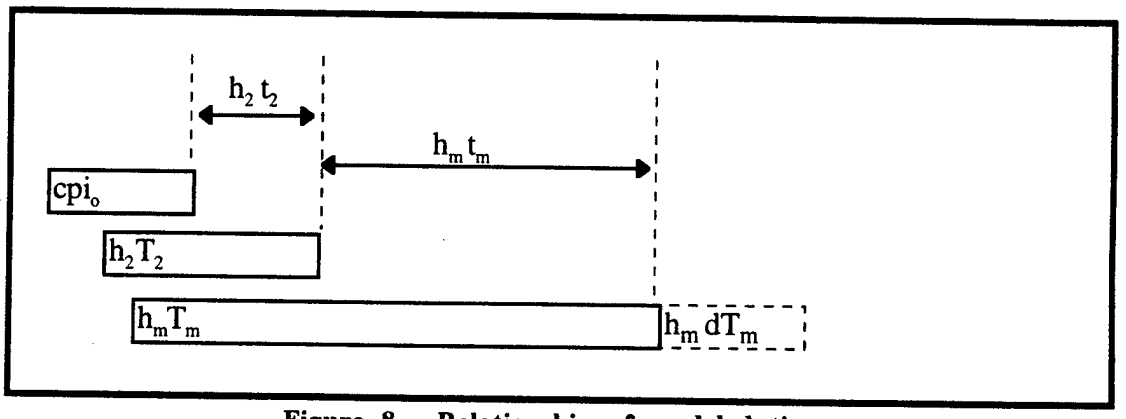

Figure 8. Relationship of modeled times

This equation predicts a linear relationship between dT_m and cpi' of slope h_m. If any additional memory latency incurred by dT_m can be hidden, this will serve to decrease the slope predicted by Eq. 4. That is, h_m is an upperbound for the increase in time due to memory latency. This analysis will be used and verified in a later section.

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 on 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 hit ratios are also directly measurable and the unknowns in Equation 1 become the average times, t_i, and the infinite-L1 computation time, cpi_0. These are inferred from the measured data by a least squares fit constrained such that

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

and

$$cpi_0 >= .25 \text{ (up to 4 issues per cycle).}$$

Table 2 shows the model parameters for each of the ASCI codes determined from a dataset of executions on the 1-MB L2 PowerChallenge. The empirical fit generally has errors that are less than 6%. The maximum latencies, T_i, are measured with LMBENCH (see Table 2) and are found to be consistent with published numbers by SGI perfex cost table.

	t_2	t_m	cpi_0
HEAT	0	121	1.0
HYDRO	.5	118	1.1
HYDRO-T	0	56	.9
SWEEP	11	205	.5
NEUT	0	205	.8
LMBENCH	11	205	4.8

Table 2. 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 9 shows that the fit is extremely close.

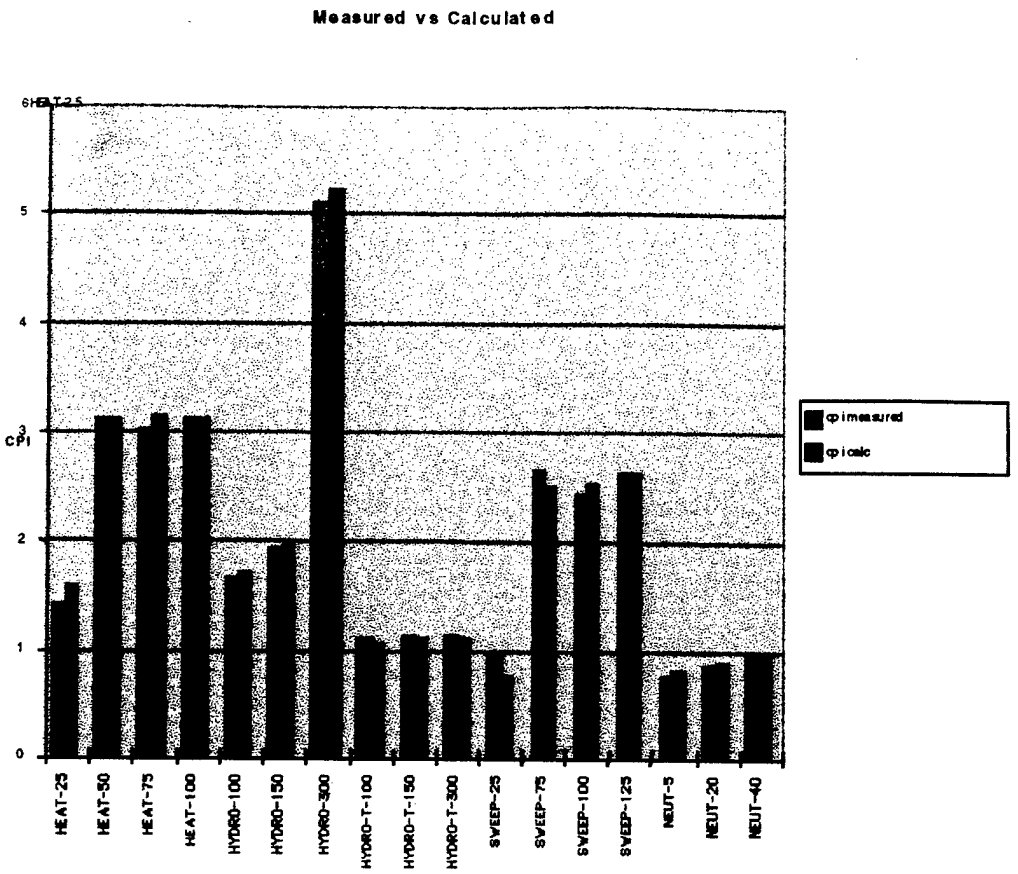

Figure 9. Model Fit for ASCI codes with varying problem sizes.

In addition, confidence in the methodology is further tested with an independent measurement of cpi_o using an R10000 simulator made available from SGI [13]. We also executed problem sizes designed to fit entirely in the L1 cache. Table 3 shows the data from these measurement/simulations compared with the model cpi_o. Most model parameters and measurements are in good agreement. However, for HEAT and SWEEP, cpi_o is not consistent across the three independent measurements. We note, however, that the values of cpi obtained from L1-cached runs are consistent with at least one other measurement, and since it is a directly-measured parameter, we will use these measured values of cpi_o in the analysis below, fitting the remaining two parameters, t_2 and t_m.

	Model cpi_o	Simulated cpi_o	L1-cached cpi
HEAT	1.0	.59	.92
HYDRO	1.1	.8	.89
HYDRO-T	.9	—	.9
SWEEP	.5	.94	.88
NEUT	.8	.76	.77

Table 3. Model, simulated, and L1-cached cpi.

As discussed above, we might expect cpi_0 to vary with problem size since instruction execution is dynamic on the R10000. Using the simulator, we found this effect to be small as can be seen from the data in Table 4.

	10	20	30	50	100
HYDRO	.80			.84	.85
SWEEP		.94	.94	.97	

Table 4. Simulated cpi_0 for two codes as a function of problem size.

VI. Results and Analysis

a. Analysis of stall time due to memory accesses.

Table 5 compares the memory access times, t_i, for the ASCI codes on the Power Challenge and the Origin 2000. In general, L2 cache accesses are completely 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_m indicate that accesses to the secondary cache are not overlapped. The reason that SWEEP stands out may be due to loop-carried dependences in the inner loops. These dependences 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 high cache-hit ratios; see Figure 6) that small absolute least square errors can result in large relative changes to the parameters.

	t_2 Power Chal	t_m Power Chal		t_2 Origin 2000	t_m Origin 2000
HEAT	0	125		0	50
HYDRO	3	120		2.4	53
HYDRO-T	0	72		0	11
SWEEP	11	145		11	43
NEUT	0	183		7.7	80
LMBENCH	11	205		11	80

Table 5. Memory Access times, t_i , for the Power Challenge and Origin 2000

Figures 10 and 11 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 totat 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 Power Challenge which is indicative of the better latency hiding capability of the Origin .

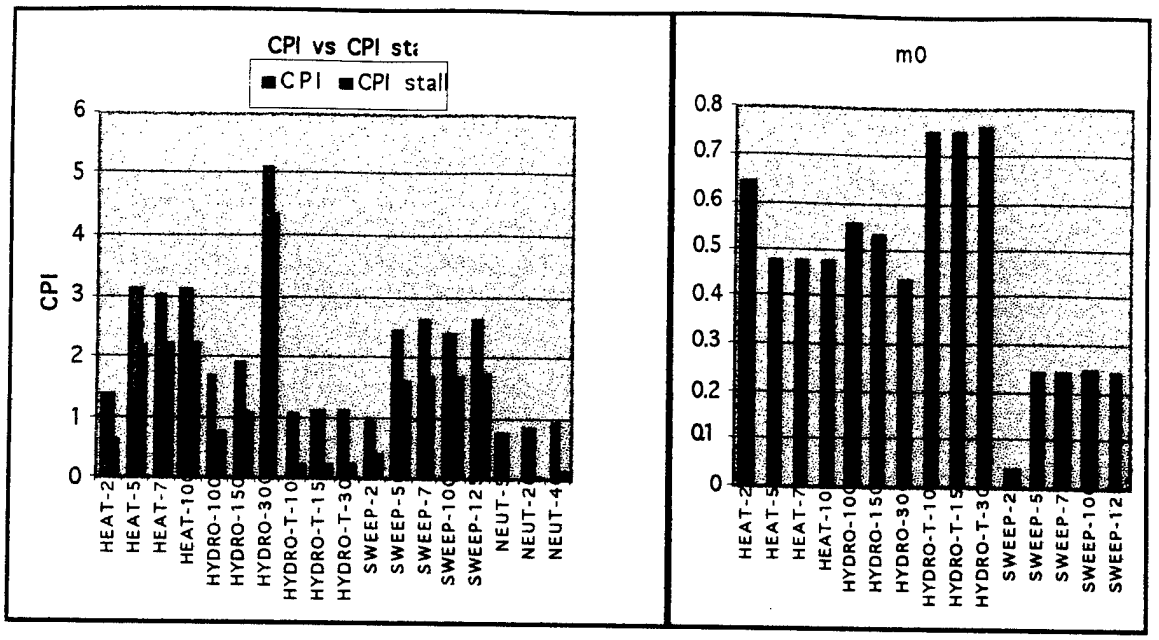

Figure 10. Memory stall & overlap parameters (Power Challenge)

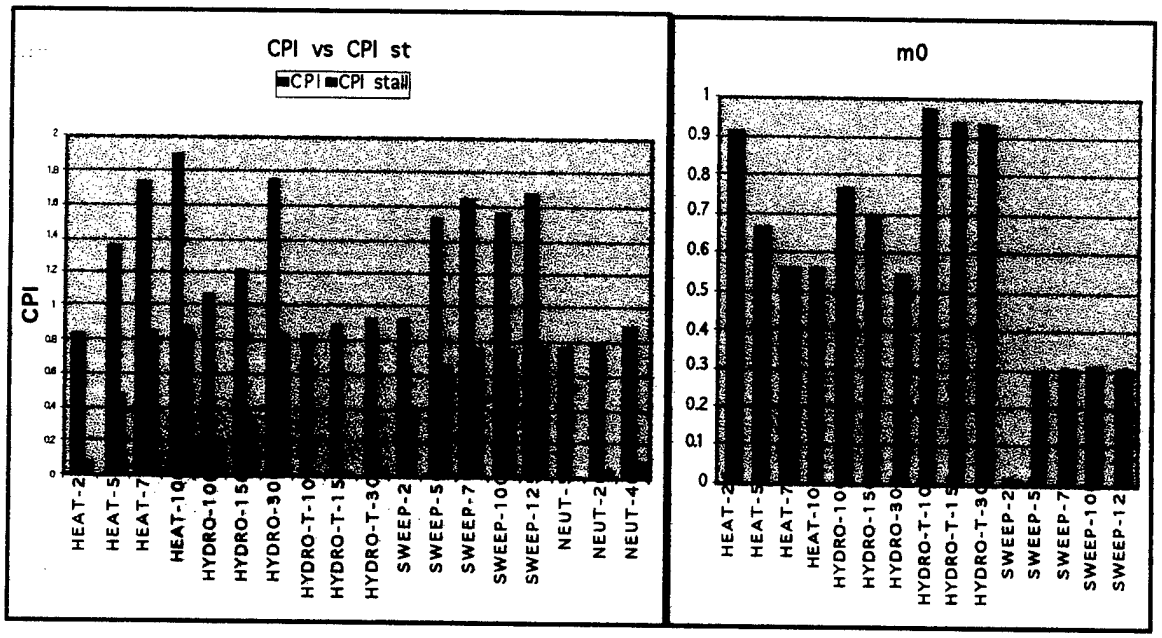

Figure 11. Memory stall & overlap parameters (Origin2000)

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 L1, L2 and 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 4 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 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 I, we performed 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. Figures 12-15 display the measurements for four codes (HEAT, HYDRO, HYDRO-T and SWEEP) showing the effect of memory latency on the measured cpi. A linear dependency observed in agreement with Eq. 4, where the slope is bounded by h_m(see discussion in Section IV).

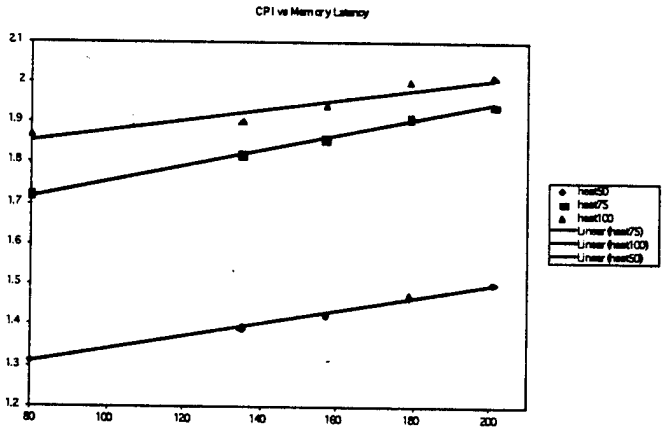

Figure 12. Observed HEAT CPI vs. Memory latency (Origin 2000)

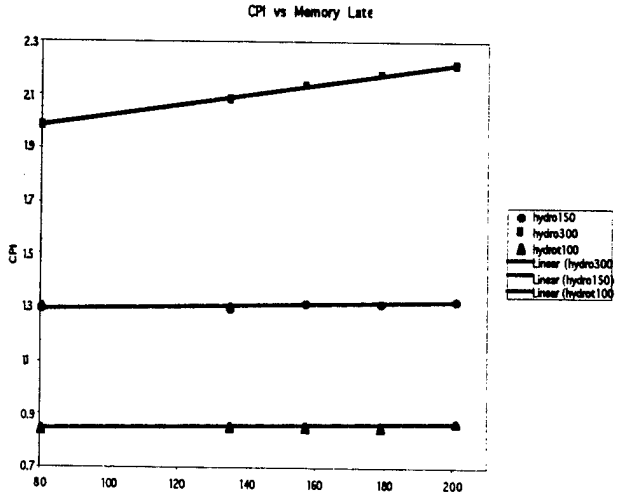

Figure 13. Observed HYDRO CPI vs. Memory latency (Origin 2000)

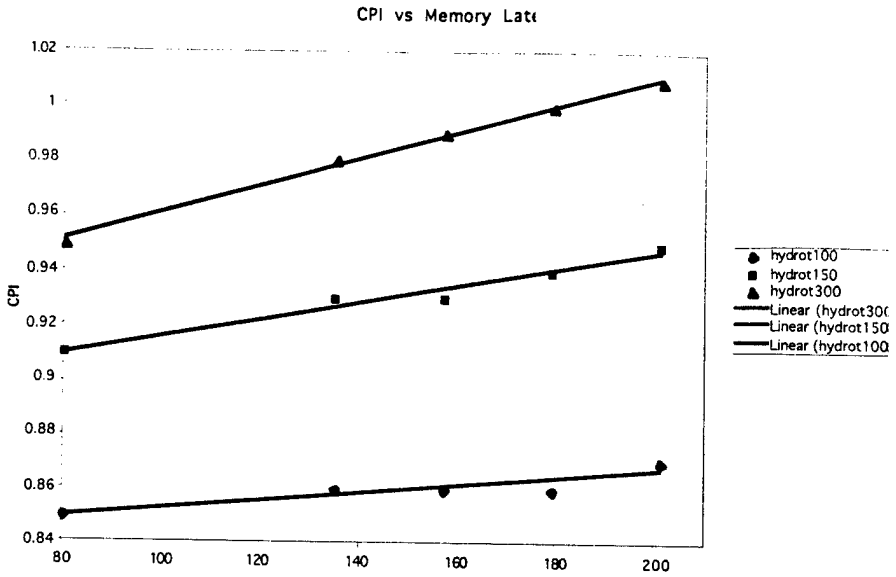

Figure 14. Observed HYDRO-T CPI vs. Memory latency (Origin 2000)

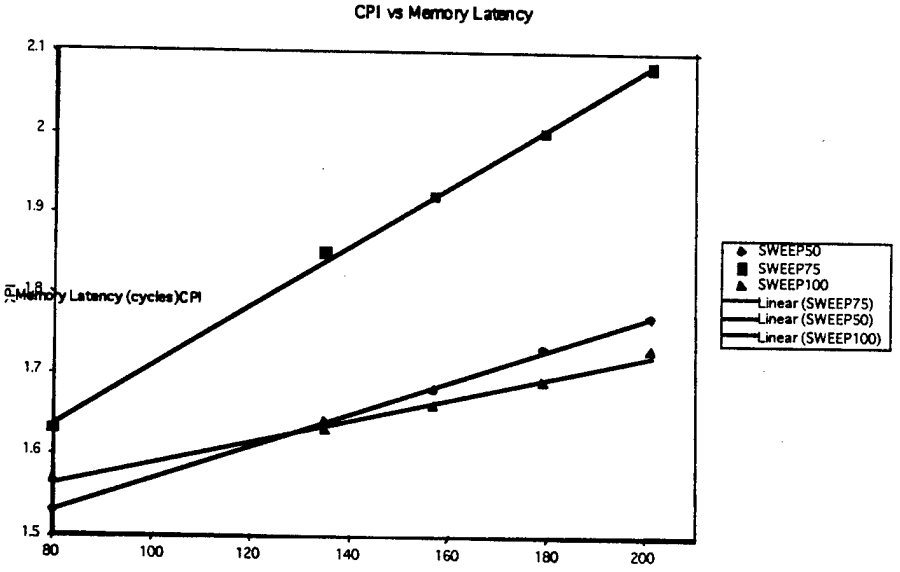

Figure 15. Observed SWEEP CPI vs. Memory latency (Origin 2000)

Using these measurements and other empirical data on the two machines, we can infer the separate the 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, outstanding misses and memory latency, respectively) such that:

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

These factors can be defined as follows:

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

$$f_o = \frac{h_2^{O} t_2^{PC} + h_m^{O} t_m^{PC} + cpi_0}{h_2^{O} t_2^{O} + h_m^{O} t_m^{O^*} + cpi_0} \quad (8)$$

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

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). The larger cache size simply changes the hit ratios, h_i^{PC}, to h_i^{O}. Similarly, the denominator in f_o represents a virtual machine identical to the Origin except with a memory latency equal to that of the Power Challenge. The quantity, $t_m^{O^*}$, is the non-overlapped memory access time on this kind of virtual Origin, which has full Power Challenge memory latency. The cpi for this machine is measured as in Figures 12-15 (when the memory latency is around 201 cycles). The quantity, f_c, then, is the ratio of the actual Power Challenge to a Power Challenge with the Origin's cache. The quantitiy, 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 6, along with the calculated and observed values, F, for the codes. The calculated and observed speedups are in good agreement. With the

Code	f_c	f_o	f_m	F_{calc}	F_{obs}
HEAT50	1.46	1.42	1.07	2.22	2.36
HEAT75	1.02	1.59	1.09	1.76	1.80
HEAT100	1.00	1.55	1.12	1.74	1.68
HYDRO100	1.42	1.06	1.02	1.53	1.53
HYDRO150	1.35	1.09	1.09	1.59	1.47
HYDRO300	2.00	1.17	1.28	3.01	2.56
HYDRO-T100	1.17	1.05	0.99	1.22	1.28
HYDRO-T150	1.09	1.10	1.03	1.25	1.25
HYDRO-T300	1.01	1.13	1.08	1.23	1.21
SWEEP50	1.06	1.32	1.13	1.58	1.60
SWEEP75	1.00	1.22	1.27	1.56	1.63
SWEEP100	1.00	1.48	1.06	1.58	1.55

Table 6. Observed and calculated performance on the Origin2000

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

(typically showing10% improvement). Most of the overall improvement 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.

Conclusions

This paper describes an empirical model which allows us to infer the separate contributions of three major architectural features in the memory subsystem of the Origin 2000. The model accounts for the overlap of processor execution with memory accesses. In general, significant amount of overall time is spent on memory accesses. 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%. The major contribution in reducing the memory access time is the increased number of outstanding misses in the Origin 2000. The effect of cache size on the performance of these codes is generally much less important. Currently, the methodology is an excellent diagnostic tool that can provide information about the actual time that an application spends in memory accesses. Future work will attempt to enhance the predictive capability of the model.

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, January, 1995, and references therein.

3. Galles, M. and Williams, E., "Performance Optimizations, Implementation, and Verfication 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 -, 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,"

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. Turner, S., Silicon Graphics Computer Systems, private communication, January, 1997.

Report Number (14) LA-UR--97-3462
CONF-980214--

Publ. Date (11) 199709
Sponsor Code (18) DOE/DP, XF
JC Category (19) UC-705, DOE/ER

DOE