Application Performance on the MIT Alewife Multiprocessor

Frederic T. Chong and John D. Kubiatowicz*
ftchong@lcs.mit.edu, kubitron@lcs.mit.edu

MIT Laboratory for Computer Science

This paper reports on the performance of several applications on the Alewife machine, focusing on emerging applications and evolving architectural mechanisms. We show that low-latency miss-handling mechanisms for both local and remote accesses, such as those in Alewife, coupled with careful data placement in the application make these emerging applications viable candidates for shared-memory parallel processing. In fact, we discover that efficient shared memory is an excellent communication mechanism for fine-grain applications (even in the absence of data re-use), that have long been considered message-passing applications. Not surprisingly, we find that Alewife mechanisms perform well on traditional coarse-grain applications. We confirm that hardware support for limited sharing is adequate for a broad range of applications, even on large numbers of processors. We also observe that modeling local cache miss behavior is important for machines, such as Alewife, where remote misses have become more competitive. To account for the effect of local misses, we introduce two novel performance metrics, which provide more revealing results than previously proposed metrics. We conclude that the fine-grained applications can take advantage of Alewife's high integration and efficiency to achieve a new level of performance on scalable shared memory machines.

Table 1 lists the applications, a short description of the problem each of the programs solves, and the input parameters. MP3D, BARNES, LOCUS, CHOL, and WATER are from the SPLASH suite [SWG92]. APPBT and MG are part of the NAS parallel benchmarks [Bai94]. The rest of the applications are engineering-type kernels from the University of Rochester, MIT, and Berkeley.

Remote cache misses from a processor to another processor's cache or memory are an integral part of every multiprocessor application study. It turns out that local cache misses are also very important on the Alewife machine because remote misses are only about five times as expensive as a local miss. A local misses takes 10 cycles, a clean read miss to remote memory takes 38 cycles and a three-party miss takes 63 cycles.

Figure 1 presents our cache hit statistics. An important point is that remote misses are not indicative of application performance in themselves. In particular, note that many applications exhibit increasing local cache hit ratios as the number of processors increases. This is because the total available cache increases while we hold the data size constant. On multiprocessors with low remote to local miss ratios, this local cache effect has a significant impact on performance. To capture the effect of local misses, the bottom graph in Figure 1 presents the hit ratio for a weighted total of memory references, computed by:

$$\frac{(\text{remote hits} + (\text{local hits})/5)}{(\text{remote accesses} + (\text{local accesses})/5)}.$$

In essence, this formula assumes that five local misses are equivalent to a remote miss, and accounts for the overhead of local misses in order to compute the overall effect of cache misses. We find this metric to be more indicative of application performance. This metric, which we call *weighted miss ratio*, is more indicative of application performance than local and remote miss ratios in isolation or combined without weighting.

Overall, we find that EM3D, ICCG, and MP3D have lower hit ratios than the other applications. This partly explains the lower utilization on these applications. However, low hit ratios only lead to poor processor utilization when there is little actual computation between memory references. Our full paper analyzes the amount of computation in between cache misses. In combination with cache hit ratios, these two metrics allow us to determine the effective granularity of the applications.

The full paper and related documents are available from http://www.ai.mit.edu/people/ftchong/

References

[Bai94] D. Bailey et al. The NAS Parallel Benchmarks. Technical Report RNR-94-007, NASA Ames Research Center, March 1994.

[SWG92] Jaswinder Pal Singh, Wolf-Dietrich Weber, and Anoop Gupta. SPLASH: Stanford parallel applications for shared-memory. *Computer Architecture News*, 20(1):5–44, March 1992.

*Joint work with Beng-Hong Lim (IBM Watson) and Ricardo Bianchini (UFRJ Brazil). This work is advised by Anant Agarwal and is funded in part by ARPA contract N00014-94-1-0985 and in part by NSF grant MIP-9504399. Address: Room 613, 545 Technology Square, Cambridge MA 02139.

Program	Description	Input Data
MP3D	Simulates rarefied fluid flow	18000 particles, 6 iterations
MMP3D	Modified MP3D	18000 particles, 6 iterations
BARNES	Simulates movement of bodies under gravitational forces	16K bodies, 4 iters
LOCUS	Routes of wires in a standard cell circuit	3817 wires
CHOL	Cholesky factorization of a sparse matrix	BCSSTK15 (order 3948, 56934 floats)
WATER	Simulates movement of water molecules	125 molecules, 10 iterations
APPBT	Solves multiple independent systems of equations	$20 \times 20 \times 20$ floats
MG	3D Poisson solver using multi-grid techniques	56x56x56 floats
EM3D	Electromagnetic wave propagation through 3D objects	20000 nodes, 20% remote neighbors
GAUSS	Unblocked Gaussian elimination	512x512 floats
FFT	1D Fast Fourier Transform	80000 floats
CGRID	Straightforward 2D successive over-relaxation	512x512 floats, 50 iters
MSORT	Sorts a list of integers	64K integers
ICCG	Preconditioned conjugate gradient sparse solver	BCSSTK18 (order 11948, 149090 doubles)

Table 1: Applications and Kernels

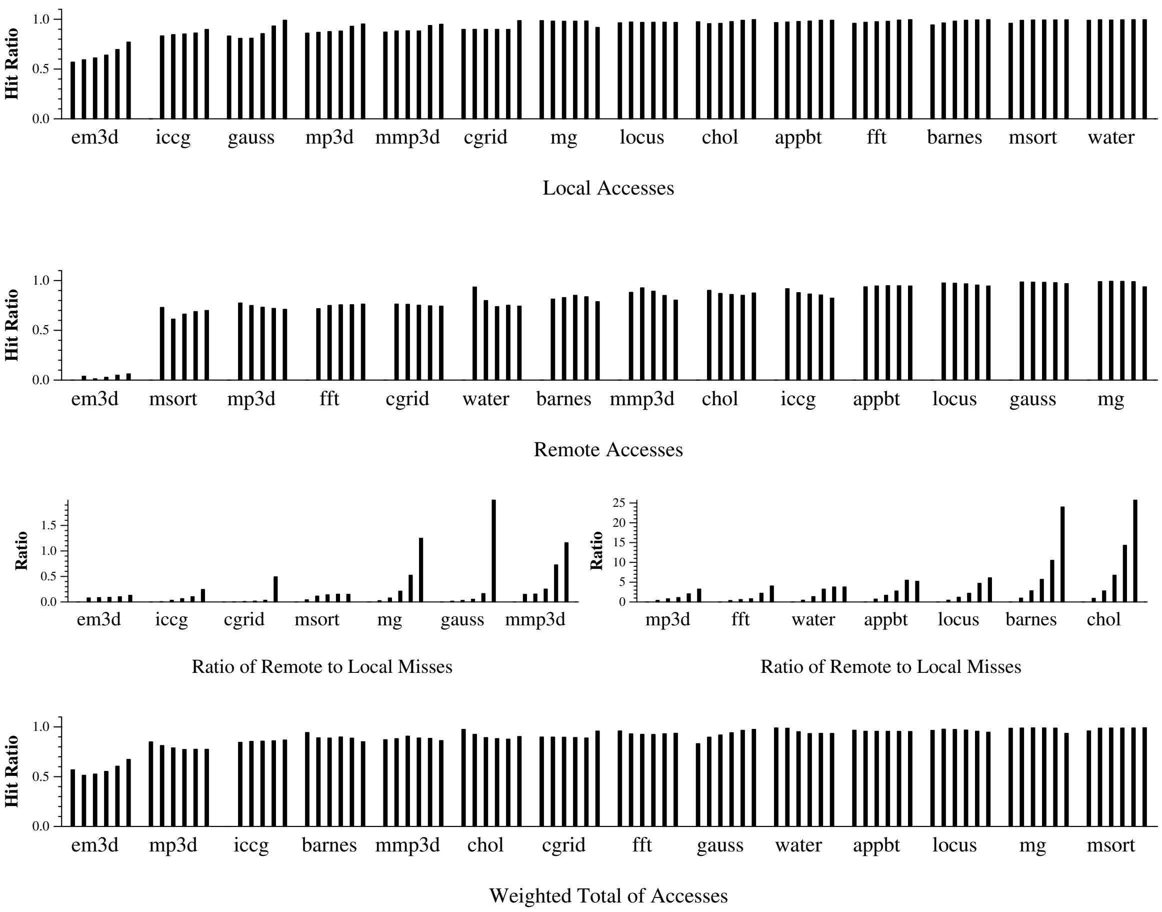

Figure 1: Cache hit ratios of applications sorted by average. Bars are for 1, 2, 4, 8, 16, and 32 processors for each application. ICCG is missing a 1-processor bar because its dataset does not fit on a single Alewife processor.