Comparing Commodity SMP System Software with a Matrix Multiplication Benchmark

Georgios Tsilikas and Martin Fleury
Electronic Systems Engineering Dept., University of Essex
e-mail: {gtsili,fleum}@essex.ac.uk}

Abstract

Commodity symmetric multiprocessors (SMPs), though originally intended for transaction processing, because of their availability, are now used for numerical analysis applications in SMP clusters. Dense matrix multiplication is a suitable benchmark as it exposes memory access issues. Various block-based algorithms are compared across Windows 2000 and Linux variants with `gcc` *compiler versions. Differences in compilers may explain differential performance between algorithms.*

1 Introduction

Choice of commodity symmetric multiprocessor (SMP) operating system (o.s.) and compiler has an influence on the performance of numerical analysis code. Therefore, issues of code portability between o.s. also play a part in choice of software. To compare the SMP versions of Linux and Windows 2000, we selected matrix multiplication (MM) as a benchmark. Dense MM is a fundamental linear algebra kernel, which can be cast as the basis of all level-3 BLAS routines [5] [7]. There are many MM algorithms; [11] contains a useful review, though it does not cover self-modify code libraries such as ATLAS (see below).

The intention of our study is not to benchmark optimal implementations of MM on a uniprocessor, but to compare the effect of the multiprocessor software environment on the performance of algorithms. However, our study is indirectly related to these optimizing approaches, as it brings choice of o.s. and compiler into the picture as a further complicating factor.

By 'commodity multiprocessor' we mean Intel architecture (IA) SMPs [13], based on quad Pentium III Xeon processors. These SMPs were originally intended as commercial database servers, being optimized for I/O access with dual PCI busses. As an Intel 'quad' unit commonly forms a single 'blade' within a multi-blade rack-mounted machine, economies of scale have been made in their production, making this architecture also attractive to researchers. As a result, quad SMP nodes now form the nodes of SMP clusters, where they certainly are used for numerical analysis applications. However, the AMD multiprocessor architecture, which incorporates per processor dedicated front-side busses (FSB), rather than the shared FSB in the IA, is outside the scope of this paper.

MM was used as a demonstrator for the Cilk language [2], originally developed for shared-memory machines programmed through multi-threading. Associated with Cilk are a set of 'cache-oblivious' algorithms [6], including one with a recursive 'divide-and-conquer' structure for generalized matrix multiplication. This algorithm is an adaptation of an earlier demonstration algorithm for square matrices [3], which resembles the standard recursive block algorithm for MM. We implemented the Cilk algorithm under Linux systems and as an approximation to it, a recursive block algorithm under Windows 2000. As a base-line for comparison, we used a simple-minded or naïve implementation of MM, multiplying row by column, and then allocating a set of rows to each thread.

Cilk is highly optimized for the `gcc` compiler running on Unix and Unix-like o.s., and does not readily transfer to Windows style o.s. or its `cl` compiler. Underlying Cilk are POSIX threads (Pthreads) , which are weakly supported in the Windows NT o.s. [12], and, hence, Cilk will not run in this environment. The common integrated development environment in a Windows environment is Visual C++, with the `cl` being the underlying compiler. To run `gcc` under Windows 2000 requires a software emulator such as `CygWin` or `MinGWI` which introduces a further complicating factor. The Intel C++ compiler works across both Linux and Windows 2000. However, the Intel compiler, which is not open source, requires a license for Windows 2000, though not for non-commercial use on Linux systems.

A simple block MM algorithm is also possible. We ported this block algorithm from Linux to the Windows 2000 o.s. and `Win32` threads, allowing performance comparisons using this intermediary program with the Cilk al-

Proceedings of the 14th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP'06)
1066-6192/06 $20.00 © 2006 IEEE

gorithm across operating systems and compilers.

The Cilk approach was subsequently extended in the PHiPAC and later ATLAS project [14] to automatically tune a matrix multiplication kernel at run-time. ATLAS's "Automated Empirical Optimization of Software" (AEOS), combines a code database of hand-tuned routines with exploration of parameter space, through source code generation. The ATLAS libraries includes a thread-safe extension to multiprocessors.

Performance comparisons were on a Dell PowerEdge 6400, varying the o.s. between Windows 2000 (server version) and Linux. Windows 2000 has a similar software architecture to Windows NT [12], upon which it builds. The PowerEdge 6400 with quad Pentium III Xeons with CPU clock speed of 700 MHz has 16 KB level-one (L1) cache (4-way associativity, line size 32 B) and 1 MB level-two (L2) cache (8-way associativity, line size 32 B) and 1 GB global main memory. The front-side bus is clocked at 100 MHz.The tool `lmbench` [8] recorded L1, L1/L2 and main memory latencies as respectively 4.3, 8.8, and 143.5 ns. In the case of Linux, both the kernel and the compiler version were varied. The Linux o.s. kernel [4] has been revised and provides reduced latency thread scheduling. The `gcc` compiler has also been revised with a not insignificant improvement in performance. Tests were also made using this improved Linux software environment.

2 Results

Figure 1 shows tests for a range of square matrix size to double precision running under Linux Red Hat v. 7.2, kernel version 2.4.9-31smp. A vector-to-vector organization of matrix layout was used, whereas the Cilk matrix layout was a one-dimensioned array. The vector-to-vector layout, whereby a vector of 'C++' pointers, point to a set of dynamically-created matrix rows, is well known for its flexibility The plots are annotated with the number of threads, and though the underlying Linux o.s. is responsible for thread placement, it is safe to assume that threads are placed on a per-processor basis. The non-Cilk algorithms were implemented without calls to the portability library, and a recursive single processor version of the Cilk algorithm was directly derived from Cilk by removing Cilk directives.

A value of $c = 96$, or equivalently a block size of 32, was found to give good overall performance on the Dell machine for the Cilk algorithm. Similarly, a value of 32 was by found by experiment to be a good setting for the block size in the block algorithm. Note carefully that as doubles take up four bytes of storage, the figure of 32 does not directly correspond to the cache line size of either L1 or L2 caches on these machines. Figure 2 shows results for the two algorithms for larger sized matrices using all four processors. The block algorithm plots have well-defined minima at 32, whereas the Cilk plots for larger matrices are relatively flat.

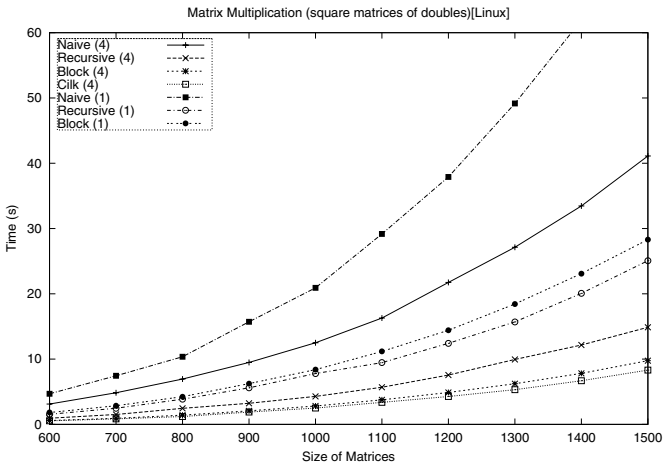

Figure 1. MM on the Dell PowerEdge 6400 running Linux and using the `gcc` compiler

However, below 32, the Cilk performance rapidly declines illustrating the need to choose a block size, rather than select any suitable low value.

The `gcc v. 2.96` compiler optimization settings did have a significant effect for algorithms run under our portability library, either level two or three being selected depending on algorithm. However, the Cilk implementation did not benefit from more aggressive optimization settings.

Under Linux, good speed-ups are achieved by Cilk followed closely by the simple block algorithm. It is clear that the naïve algorithm is not at all competitive, as even the multithreaded version is outperformed by the sequential versions of both the Cilk and block algorithms.

A further comparison was made by running the same tests but taking timings at more frequent intervals, Figs. 3. There are performance drop spikes, which are very severe in their effect on the naïve algorithm. The effect of the cache-oblivious algorithms, block, recursive or Cilk, is to smooth out the oscillations seen in the naïve algorithm plots, also making performance more predictable. The same can be said about increasing the number of threads (processors) from one to four. Similar trends were found in later tests under Windows 2000, and these no doubt also reflect the underlying organization of the cache system. Other tests (not reported herein) using hardware counter instrumentation confirmed that level 2 cache access has the main impact on access times.

Windows 2000 Pro (Service Pack 5) was also installed on the Dell PowerEdge 6400. The results of taking timings is seen in Fig. 4. The measurements show that for larger-sized matrices the performance of the Dell machine is improved

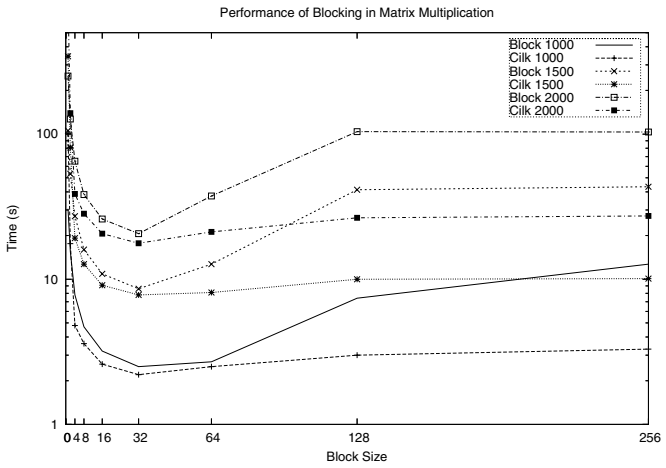

Figure 2. MM block size timings for small matrices on the Dell PowerEdge 6400 running Linux and the `gcc` compiler for the Cilk and block algorithms

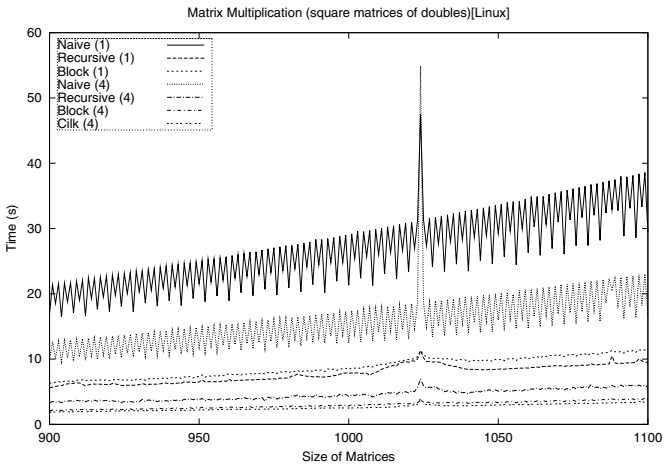

Figure 3. Fine-grained timings of MM on the Dell PowerEdge 6400 running Linux using the `gcc` compiler

for the block algorithm. However, the recursive algorithm fares less well dropping from about 15 s to about 27 s.

It is difficult to be sure whether the drop in performance for the recursive algorithm is attributable to operating system or compiler. Windows 2000, though organized as components, is also highly layered, with system calls passing through a Win32 subsystem, through the `NTDLL.DLL` system interface, to memory manager procedures, then the kernel and the Hardware Abstraction Layer (HAL) until the hardware is addressed. Linux's monolithic kernel appears more direct, with fewer levels of indirection. However, there is a differential effect between the recursive algorithm and the block algorithm, which suggests that differences between the compilers are the principal influences occurring for this benchmark and not differences between o.s. This confirms product comparison surveys, which suggest that the code produced by `gcc` was very different from that by `cl`. Using various sets of tests, a consensus view seems to be that `cl` compares favorably to `gcc` but clearly this is dependent on many factors.

Comparative timings when the same algorithms were run on the Dell machine, but with different versions of the Linux kernel. Red Hat Linux version 7.2 supports version 2.4 of the kernel, whereas the Debian unstable (DU) software release is able to support version 2.6.4 of the kernel. There are also some differences in compiler technology. Version 2.96 of the `gcc` compiler is a specialized version from Red Hat, and, therefore, the nearest equivalent under Debian is version 2.95. Cilk would not compile under version 3.3 of the compiler. Figure 5's comparison between the block algorithm with change of kernel demonstrates a consistent improvement in performance, all measurements being taken with four threads (processors). However, there is a greater improvement from changing the compiler version with the same kernel, version 2.6. Therefore, again the compiler is the main determinant of improved performance.

The Cilk implementation remains good in all circumstances. However, timings for Cilk under different kernel versions were virtually identical, which may be attributable to the Cilk run-time load balancing component. In this instance, the Cilk load balancer may over-ride kernel thread-scheduling.

In general, a comparison of the times for block/Cilk algorithms on the Dell machine with the two operating systems, shows equivalent performance. Speed-up on the Dell machine, Fig. 6 with timings taken for 1, 2, 3 and 4 processors, shows the Cilk and block algorithms gaining little beyond three processors, whereas the other algorithms gain from more processors, but equally the performance is weaker. Marginally best performance arises from the Windows 2000 Pro (W2K) block algorithm version.

3 Conclusions

Cilk is an example of a multiprocessor programming language that is dependent on the presence of POSIX threads and on the compiler type, which rules out the common operating systems for commodity multiprocessors, namely Windows NT or 2000 Pro. However, Cilk's performance on a matrix multiplication (MM) benchmark is comparable with a block recursive algorithm for MM, which could be ported between operating systems. A non-recursive block algo-

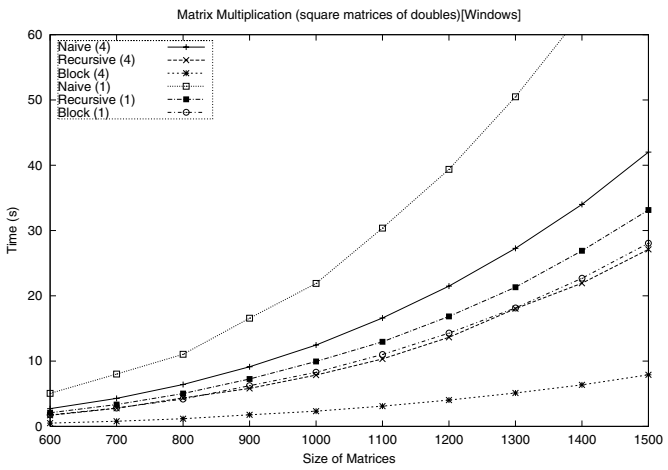

Figure 4. MM on the Dell PowerEdge 6400 running Windows Pro 2000 using the `cl` compiler

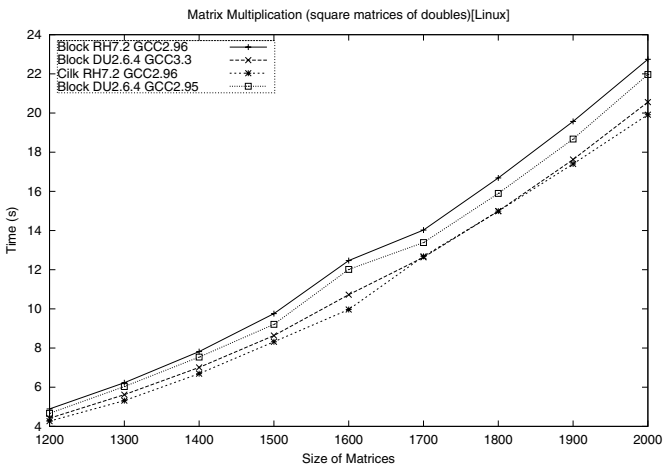

Figure 5. MM on the Dell PowerEdge 6400 running different versions of the Linux kernel and the `gcc` compiler

rithm was also found to be competitive. It became possible to make direct and indirect comparisons between the multiprocessor o.s. and compilers. The tests imply that the layered organization of Windows 2000 did not lead in itself to weak performance for the recursive block algorithm as the non-recursive block algorithm's performance improved on the same machine. This suggests that differences between compilers are the principal factors at play and future work would remove compiler dependency by use of Intel's compiler.

References

[1] R. Blumofe, C. F. Joerg, B. C. Kuszmaul, C. E. Leiserson, K. H. Randall, and Y. Zhou. Cilk: An eficient multithreaded runtime system. In *5th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 207–216, 1995.

[2] R. D. Blumofe, M. Frigo, C. F. Joerg, C. E. Leiserson, and K. H. Randall. Dag-consistent distributed shared-memory. In *10th International Parallel Processing Symposium*, pages 297–308, 1996.

[3] D. P. Bovet and M. Cesati. *Understanding the Linux Kernel*. O'Reilly, Beijing, 2002.

[4] J. J. Dongarra, J. D. Croz, I. S. Duff, and S. Hammarling. A set of level 3 basic linear algebra subprograms. *ACM Transactions on Mathematical Software*, 16(1):1–17, 1990.

[5] M. Frigo, C. E. Leiserson, H. Prokop, and S. Ramachandran. Cache-oblivious algorithms. In *40th Annual Symposium on Foundations of Computer Science*, pages 285–297, 1999.

[6] P. Kagström, P. Ling, and C. Van Loan. GEMM-based level 3 BLAS: High performance model implementations and performance evaluation benchmark. *ACM Transactions on Mathematical Software*, 24(3):268–302, 1998.

[7] L. W. McVoy and C. Staelin. lmbench: Portable tools for performance analysis. In *USENIX Annual Technical Conference*, pages 279–294, 1996.

[8] S. Sen, S. Chatterjee, and N. Dumir. Towards a theory of cache-efficient algorithms. *Journal of the ACM*, 49(6):828–858, 2002.

[9] D. A. Solomon. *Inside Windows NT*. Microsoft Press, Redmond, WA, 1998.

[10] M. Weitz. 4-way servers. *Windows NT Magazine*, pages 141–143, 2000.

[11] R. C. Whaley, A. Petitet, and J. J. Dongarra. Automated empirical optimization of software and the ATLAS project. *Parallel Computing*, 27:3–35, 2001.

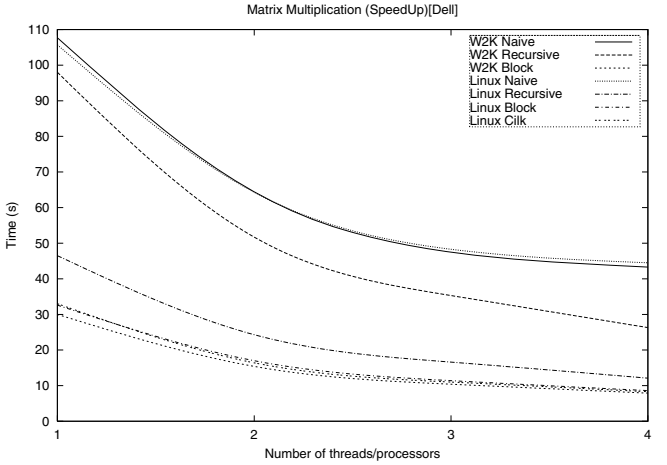

Figure 6. MM speed-up under Linux and Windows 2000 Pro on the Dell PowerEdge 6400