Lattice QCD with Commodity Hardware and Software

D. Holmgren, P. Mackenzie, D. Petravick, R. Rechenmacher and J. Simone

Fermi National Accelerator Laboratory, Batavia, IL, USA

Abstract

Large scale QCD Monte Carlo calculations have typically been performed on either commercial supercomputers or specially built massively parallel computers such as Fermilab's ACPMAPS. Commodity computer systems offer impressive floating point performance-to-cost ratios which exceed those of commercial supercomputers. As high performance networking components approach commodity pricing, it becomes reasonable to assemble a massively parallel supercomputer from commodity parts. We describe the work and progress to date of a collaboration working on this problem.

Keywords: Lattice QCD, Commodity Computing, Massively Parallel Computers

1 Introduction

Lattice QCD is a method of studying Quantum Chromodynamics (QCD) numerically. A vigorous lattice QCD research program is a necessary adjunct to experimental high energy physics research. Results from lattice calculations are needed to extract parameters of the Standard Model of particle physics from measurements obtained from experiment. The next generation of experiments such as Babar, CLEO III and those at the Fermilab Tevatron will demand more ambitious lattice calculations with theoretical errors at the level of those experiments. For example, Fermilab's CDF expects to measure x_s at the percent level. To connect this to CP violation requires a similarly precise QCD matrix element. This will require an estimated 10^2 to 10^3 fold increase in lattice computation.

Large scale QCD calculations have typically been performed on either commercial supercomputers or on specially built massively parallel computers such as Fermilab's ACPMAPS [1]. The next generation of lattice calculations will require a new generation of computer. Commodity computer systems offer impressive floating point performance-to-cost ratios which exceed those of commercial supercomputers. As high performance networking components achieve commodity pricing and achieve greater bandwidths and lower latencies, it becomes appealing to assemble a massively parallel computer from commodity hardware.

This concept is under investigation by a collaboration of high energy theorists and computing professionals representing Fermilab's Theoretical Physics [2] and Distributed Systems Projects [3] Groups, the MILC [4] lattice collaboration and the Cornell Theory Group [5].

2 Hardware

We have constructed a prototype cluster consisting of nine nodes[1]. Eight of the systems are based upon dual 500-MHz Intel Pentium-III processors and Intel L440-GX+ ("Lancewood") mother-

[1]Work supported by the U.S. Department of Energy under contract No. DE-AC02-76CH03000.

boards. The ninth system is based upon a 600 MHz AMD Athlon processor and an FIC SD-11 motherboard. All systems have 128 MB of 100 MHz SDRAM memory.

The Athlon-based system, and seven of the Pentium-III systems, are interconnected via a Myrinet network [6] consisting of M2L-PCI64A-2 PCI interface cards and an M2F-SW8 eight-port switch. The nodes are further interconnected via a fast ethernet network. The nodes of the cluster run RedHat 6.0 Linux. The Myrinet interfaces communicate via GM version 1.1.1 from Myricom. MPICH [7] version 1.1.2 is used as the communications and synchronization API.

We believe that by using commodity components that we can construct computing clusters which are easily upgradeable as more cost effective hardware becomes available. Such upgrades could even be based upon processors from the various non-x86 families. This flexibility arises because internode communications are based upon PCI interfaces, and the fact that Linux provides a common operating system and development environment for all commodity hardware platforms.

3 Lattice QCD Tests

For the tests described below, we have selected the MILC collaboration's Wilson quark propagator solver. In the large-scale computations we envision, solving for quark propagators represents well over 90% of the effort. Thus this solver is a good estimator of performance. The solver is written within the MILC parallel framework. It is portable and tolerant to communications latencies, with good overlap of communications and computation on parallel configurations. Communications are implemented through standard MPI-1.2 calls. The code iteratively solves for propagators using a sparse-matrix bi-conjugate gradient algorithm and incomplete LU preconditioning.

3.1 Single System Performance

Computer systems employ a memory cache hierarchy to match relatively slow main memory to high clock speed cpus. System performance will depend upon data size and layout in memory. We have tested Wilson solver performance for a range of problem sizes on our test cluster, as well as on an Alpha-based and on a Xeon-based system as listed in Tab. I. All systems, except the Athlon, have 500 MHz processors. The systems have a variety of cache designs.

All systems ran Linux 2.2-series kernels. On each computer the code was compiled with the gcc compiler. Each test was run on a single cpu (single process) in multiuser environments with light to moderate system loads. For these one cpu tests, communications routines in the solver code were replaced by functions that return memory references, hence, "communications" involve zero data copies.

Results are shown in Fig. 1. Performance is measured by the number of kilo-sites updated per second. Problem size is indicated by data size in megabytes. The numbers above the horizontal axis show problem size in terms of hypercube volume. QCD computations are peformed on a four-dimensional space time grid. The problem size thus scales like L^4 where L is the number of grid sites in each dimension of the hypercube.

The figure shows that for problem sizes $\lesssim 0.5$ MB the Alpha has nearly twice the performance of a PIII at the same clock speed. Scaling the 600 MHz Athlon performance by $5/6$, we expect an Athlon to outperform a PIII of the same clock speed by about 17%.

Cache organization appears to be an important indicator of perfomance for data sizes less than about 10 MB. The Alpha's larger 4 MB cache appears to give it about a 40% performance advantage over the PIII for a 3 MB data size. We note a linear decrease in performance rather than a rapid drop as the problem size exceeds cache size. We take this as evidence that our application is reusing data already in cache.

Lattices of physical interest have $L \gtrsim 10$ and in excess of 20 MB of data. The figure shows

cpu	clock (MHz)	cache speed	cache size (MB)
PIII	500	$\times 0.5$	0.5
PIII-Xeon	500	$\times 1$	1.0
Athlon	600	$\times 0.5$	0.5
Alpha 21264	500	$\times 0.5$	4.0

Table I: Features of systems tested in Fig. 2.

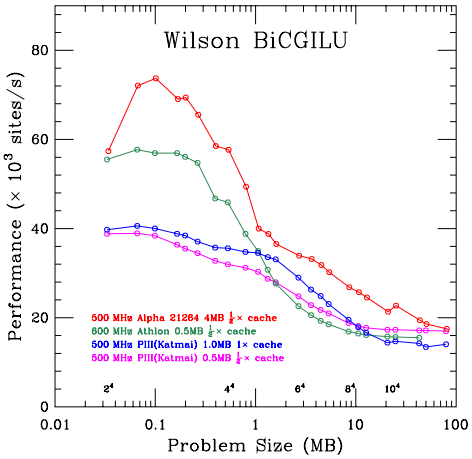

Figure 1: Single computer performance *vs* problem size on four commodity systems with different cpus.

that single processor performance for such data sizes is entirely limited by the bandwidth to main memory, irrespective of cpu speed or cache design. For the Wilson solver and large data sizes, main memory bandwidth is the most important indicator of performance.

System performance for cacheable data sizes is an important parameter when computations are parallelized. Domain decomposition is used to distribute lattice sites among computer memory spaces. For example, a $10 \times 10 \times 10 \times 10$ lattice may be partitioned on ten computers into $10 \times 10 \times 10 \times 1$ slices. There will be a speedup when a significant part of a sublattice fits in cache. Cache speedup leads to important constraints on communications requirements in order to hide latencies.

3.2 Parallel Performance

We examined parallel performance of the Wilson solver as a function of the numbers of cpus while keeping the total lattice size fixed. We show runs for $6^3 \times 12$ and $12^3 \times 24$ lattices. The smaller lattice shows the effect of cache speedup as data is distributed among more cpus. According to our single cpu performance measurements, the $12^3 \times 24$ lattice is large enough that there should be no significant cache speedup even when distributed over all available cpus.

A site update in the solver depends only on data from neighboring sites. Thus, when a lattice is distributed among cpus, an update to a site on the surface of a sublattice requires data held in another computer's memory. For optimal lattice partitioning, the relative number of surface sites decreases as $1/\ell$ for large ℓ where ℓ is the dimension of the sublattice. Hence, it is less demanding to hide communication latencies when sublattices are large.

We produced test results (Fig. 2) for both Myrinet and ethernet, distributing parallel tasks in two ways: allowing only one process per node or allowing two parallel processes per node. For

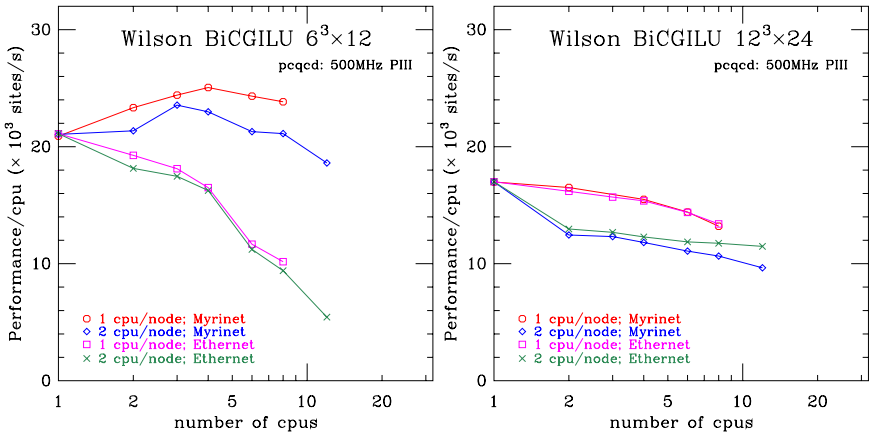

Figure 2: Performance/cpu *vs* number of cpus *vs* communications medium for two lattice sizes.

the $6^3 \times 12$ lattice we observe an increasing performance difference between Myrinet and ethernet as the lattice is distributed over more cpus. Communications performance is more demanding for small sublattices, and Myrinet latencies and bandwidth are better than those obtained with TCP over ethernet. Note that the Myrinet per cpu performance rises to greater than the performance on a single cpu. This speedup is the effect of larger portions of the lattice fitting into cache. Comparing the two Myrinet curves we see a significant performance difference between distributing one task per node and two tasks per node. We attribute this affect to a combination of contention for memory and for the Myrinet interface, and task scheduling inefficiencies. Instrumentation of our code will allow us examine this issue in greater detail.

For the $12^3 \times 24$ lattice we observe little performance difference between Myrinet and ethernet for runs involving up to twelve cpus. TCP over ethernet, even with its significant software latencies, was sufficiently performant for these runs. Based on our $6^3 \times 12$ results, we expect to see a larger difference between ethernet and Myrinet when the $12^3 \times 24$ lattice is distributed over a larger number of nodes.

4 Planned Work

Planned future work includes testing Myrinet hardware on Alpha and perhaps G3-based systems, as well as characterizing the performance of clusters with multiple layers of Myrinet switches. Assuming encouraging results, we hope to consturct a much larger cluster, on the order of 1000 nodes. Towards this end we are continuously investigating and developing the tools and techniques required to manage and monitor extremely large clusters.

References

1 http://www-isd.fnal.gov/acpmaps/acpmaps.html
2 http://www-theory.fnal.gov/
3 http://www-isd.fnal.gov/dsp/
4 http://physics.indiana.edu/~sg/milc.html
5 http://w4.lns.cornell.edu/public/theory/
6 http://www.myri.com/
7 http://www.mcs.anl.gov/mpi/mpich/