ICS05
Sponsored by ACM/SIGARCH

The 19th ACM International Conference on Supercomputing
June 20 - 22, 2005
Cambridge, Massachusetts, USA

Scaling Physics and Material Science Applications on a Massively Parallel Blue Gene/L System

George Almasi, Gyan Bhanot, Alan Gara, Manish Gupta, James Sexton, Bob Walkup
IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598
{gheorghe, gyan, alangara, mgupta, sextonjc, walkup}@us.ibm.com

Vasily V. Bulatov, Andrew W. Cook, Bronis R. de Supinski, James N. Glosli,
Jeffrey A. Greenough, Francois Gygi, Alison Kubota, Steve Louis, Thomas E. Spelce,
Frederick H. Streitz, Peter L. Williams, Robert K. Yates
Lawrence Livermore National Laboratory
Livermore, CA 94550
{bulatov, awcook, bronis, glosli1, greenough1, fgygi, kubota, stlouis, spelce1, streitz, plw, kimyates}@llnl.gov

Charles Archer, Jose Moreira
IBM Systems and Technology Group
Rochester, MN
{archerc, jmoreira}@us.ibm.com

Charles Rendleman
Lawrence Berkeley National Laboratory
Berkeley, CA 94720
CARendleman@lbl.gov

ABSTRACT

Blue Gene/L represents a new way to build supercomputers, using a large number of low power processors, together with multiple integrated interconnection networks. Whether real applications can scale to tens of thousands of processors (on a machine like Blue Gene/L) has been an open question. In this paper, we describe early experience with several physics and material science applications on a 32,768 node Blue Gene/L system, which was installed recently at the Lawrence Livermore National Laboratory. Our study shows some problems in the applications and in the current software implementation, but overall, excellent scaling of these applications to 32K nodes on the current Blue Gene/L system. While there is clearly room for improvement, these results represent the first proof point that MPI applications can effectively scale to over ten thousand processors. They also validate the scalability of the hardware and software architecture of Blue Gene/L.

Categories and Subject Descriptors

J.2 [Computer Applications]: Physical Sciences and Engineering – *chemistry, physics.*

General Terms

Measurement, Performance, Experimentation.

Keywords

Blue Gene/L, supercomputers, scalability, MPI, applications.

1. INTRODUCTION

The Blue Gene/L (BG/L) project at IBM, in partnership with Lawrence Livermore National Laboratory (LLNL), has pursued a new approach to building supercomputers [1]. Rather than using clusters of powerful symmetric multiprocessors or vector processors with expensive interconnection networks (the approach taken on the Earth Simulator [2], ASC Purple [3], and the Columbia [4] systems), it uses low power processors and integrated networks. Using low power technology allows a larger packaging density (each BG/L rack has 2048 processors), with modest electrical power consumption and heat dissipation that can be managed with air cooling. BG/L uses system-on-a-chip technology to integrate powerful torus and collective networks, and uses a novel software architecture [5] to support high levels of scalability.

Solving a scientific problem on BG/L typically requires more nodes than conventional supercomputers since BG/L has relatively modest nodes, with 5.6 Gigaflop/s peak performance and 512 MB memory. While promising results have been reported on a 512 node BG/L prototype [6], it has been an open question whether a machine like BG/L can match the computational capabilities of high end supercomputers on real applications.

Another related open question is whether applications of scientific interest can be effectively scaled to tens of thousands of processors. Previous results on performance of MPI applications have been limited to well below ten thousand processors, on systems such as ASCI White, ASCI Q, ASCI Red, and the Earth Simulator, as evidenced by winning entries on the Gordon Bell prizes (awarded annually for achievements in parallel applications performance). Some of the studies on massively parallel systems have reported problems in scaling of applications to thousands of processors due to factors like computational noise [7] or insufficient network bandwidth.

This paper begins to answer the above questions, and describes early experience with several physics and material science applications on a 32,768 node BG/L system. A half-sized version of this system, which occupies less than 400 square feet of floorspace, and consumes about 400 KW in power, was recently rated as the fastest supercomputer in the world (#1 on the November 2004 TOP500 list), with a sustained LINPACK performance of 70.7 Teraflop/s [8]. In contrast, the Earth Simulator, which delivers a LINPACK performance of 35.9 Teraflop/s, occupies an area of about 70,000 square feet, and consumes about 7 MW of power. We show that the applications targeted in this study scale well on the BG/L system, achieving the highest sustained performance ever on the respective applications. These results validate the design of the BG/L system. More importantly, this study establishes the first proof point of MPI applications scaling beyond ten thousand processors, and shows that scientific problems can be effectively solved on systems with tens of thousands of processors. This study also uncovers several opportunities for performance improvements on BG/L through software optimizations.

2. OVERVIEW OF BG/L

We briefly review the key architectural features of BG/L here; further details are available elsewhere [1].

2.1 BG/L Hardware

Each BG/L compute node has two 32-bit embedded PowerPC (PPC) 440 processors. The PPC 440 processor is a low-power superscalar processor with 32 KB each of L1 data and instruction caches. The BG/L nodes support prefetching in hardware, based on detection of sequential data access. The prefetch buffer for each processor holds 64 L1 cache lines (16 128byte L2/L3 cache lines) and is referred to as the L2 cache. Each chip also has a 4 MB L3 cache built from embedded DRAM, and an integrated DDR memory controller. A single BG/L node supports 512 MB memory, with an option to use higher-capacity memory chips. The PPC 440 design does not support hardware cache coherence at the L1 level. However, there are instructions to invalidate a cache line or flush the cache, which can be used to manage coherence in software.

BG/L employs a SIMD-like extension of the PPC floating-point unit, which we refer to as the double floating point unit or DFPU [9]. The DFPU adds a secondary FPU to the primary FPU as a duplicate copy with its own register file. The second FPU is not an independent unit: it is used with a comprehensive set of special parallel instructions. This instruction set includes parallel add, multiply, fused multiply-add, and additional operations to support complex arithmetic. All of the SIMD instructions operate on double-precision floating-point data.

The BG/L ASIC supports five different networks: torus, collective, global interrupts, Ethernet, and JTAG. The main communication network for point-to-point messages is a three-dimensional torus. Each node has six bi-directional links for direct connection with nearest neighbors. The raw hardware bandwidth for each torus link is 2 bits/cycle (175 MB/s at 700 MHz) in each direction. The torus network provides adaptive and deterministic minimal path deadlock-free routing. The collective network implements broadcasts and reductions with a target hardware latency of 1.5 microseconds for a 64K node system. The global interrupts network supports a fast barrier operation, also with a target latency of 1.5 microseconds for a 64K node system. Finally, each BG/L chip supports a serial JTAG network for booting, control and monitoring of the system, and contains a 1Gbit/s Ethernet macro for external connectivity. Only I/O nodes are attached to the Ethernet network, which connects the BG/L core to external file servers and host systems. (BG/L can be configured with an I/O node, architecturally identical to compute node, for every 8 to 64 compute nodes.)

2.2 BG/L Software

Each BG/L compute node has BG/L supports a *single program multiple data* (SPMD) programming model, with message passing via an implementation of the Message Passing Interface (MPI). A BG/L job can be submitted in one of two modes. In *coprocessor mode*, which is the default mode, a single application (MPI) process runs on each compute node – one of the processors of the compute node is used for computation, and the other is used for offloading part of the communication operations. In *virtual processor mode*, two application processes are run on each compute node, one on each of the two processors.

BG/L uses a hierarchical organization of software [5]. User applications run exclusively on compute nodes under the supervision of a simple, minimalist *compute node kernel* (CNK). The I/O nodes run a customized version of Linux. Many system calls (such as *read* and *write*) are not directly executed in the compute node, but are function shipped through the collective network to the “parent” I/O node. The control system is implemented as a collection of processes running in an external computer, called the *service node* for the machine. All of the visible state of BG/L is maintained in a commercial database on the service node.

BG/L provides an operating environment with a very low level of “computational noise” (interference from operating system activity), about two orders of magnitude lower than traditional clusters and the ASCI Q system [10]. It also supports low latency communication (latency to nearest neighbor is about 3.3 microseconds, or 2350 processor cycles), with a low *half-*

bandwidth point (half of asymptotic bandwidth is often achieved at a message size smaller than 1 KB).

3. APPLICATIONS

This section describes the applications we used in this study. We also briefly describe the scientific problems they are addressing.

3.1 Miranda

Miranda is a high order hydrodynamics code for computing fluid instabilities and turbulent mixing. It employs FFTs and band-diagonal matrix solvers for computing spectrally-accurate derivatives, combined with high-order integration methods for time advancement; e.g., fourth-order Runge-Kutta. Fluid properties, i.e., viscosity, diffusivity and thermal conductivity, are computed from kinetic theory. The code contains solvers for both compressible and incompressible flows. It has been used primarily for studying Rayleigh-Taylor (R-T) and Richtmyer-Meshkov (R-M) instabilities, which occur in supernovae and Inertial Confinement Fusion (ICF).

The grid resolution needed to support a sufficiently broad range of dynamical scales for turbulent flow is severe. Hence, virtually all past measurements of R-T and R-M growth rates have been sensitive to grid resolution (i.e., they were not converged). Very large simulations, using well in excess of 1000 grid points in each of three directions, are needed to support the large range of length scales necessary to grow R-T and R-M instabilities to full turbulence. Smaller simulations simply cannot capture true rates of growth and mixing due to initial/boundary effects.

The Miranda code has been optimized for the BG/L torus by distributing the data among Cartesian communicators, which we map to the torus to reduce message time, primarily in MPI_Alltoallv on subcommunicators. All of the code operating in a master-slave CPU manner has been replaced with fully parallel routines, memory overhead has been reduced, and the FFTW library tuned for BG/L has been used.

3.2 Raptor

Raptor is a multi-physics Eulerian Adaptive Mesh Refinement (AMR) code used for applications at LLNL including astrophysics, ICF and shock-driven instabilities and turbulence. Raptor can simulate purely fluid dynamics systems by solving the Eulerian equations (inviscid, non-conducting) or the Navier-Stokes equations (viscous, conducting) using a higher-order Godunov finite difference method. Raptor can also be used to simulate more complex physical systems where the fluids are coupled to the radiation field. A fully implicit treatment is used to solve the radiation-diffusion equation coupled to the matter internal energy.

Raptor is based on the BoxLib and AmrLib general software infrastructure developed and maintained by the Center for Computational Sciences and Engineering at Lawrence Berkeley National Laboratory. BoxLib provides C++ foundation classes for templated data containers and their efficient manipulation. AmrLib adds framework support in C++ that extends BoxLib to efficiently support the demands of block-structured AMR. The entire software system, both base and framework libraries, as well as applications software and physics algorithms, has been optimized for efficient use of modern large-scale parallel computing platforms. Raptor uses a wide variety of communication operations, including point-to-point, reductions, all-to-all, and barrier, and so is a good test of the MPI implementation on BG/L.

Simulations at full scale on BG/L will offer the computational power to gain an order of magnitude more resolution in simulations of three-dimensional shock-driven systems. Two of the systems to be investigated numerically on BG/L are modeled after the research shock tubes at the University of Arizona (low Mach number facility) and the University of Wisconsin-Madison (high Mach number facility) as well as laser driven systems like the National Ignition Facility (ultra-high energy density facility).

3.3 Qbox

Qbox is a C++/MPI implementation of the first-principles molecular dynamics (FPMD) simulation method. It is developed at LLNL and was recently ported to the BG/L platform. Qbox implements FPMD within the Density Functional Theory, pseudopotential, plane-wave framework. It relies on efficient Fast Fourier Transform (FFT) and dense linear algebra libraries. We use a specialized version of the FFT-W library for BG/L developed at the Technical University of Vienna, Austria. Dense linear algebra is implemented using the ScaLAPACK library.

Qbox has been used in the past on other large parallel platforms in many applications including the simulation of liquids and solids in extreme conditions, nanotechnology and solid-state physics. The target BG/L application of Qbox is simulation of the properties of transition metals at high pressure and high temperature.

3.4 ddcMD

ddcMD is a scalable, general purpose code for performing classical molecular dynamics simulations using the highly accurate MGPT potentials. These semi-empirical potentials, which are based on a rigorous expansion of many body terms in the total energy, are needed in order to investigate quantitatively the dynamic behavior of transition metals and actinides under extreme conditions.

To date, accurate atomic scale simulation of materials behavior has been constrained by the maximum size that can be modeled, as un-physically small simulation cell sizes introduce artificial "size effects" into the dynamics. Scientists must either draw inferences from such small simulations, or make sufficient approximations to the underlying physics (i.e., use a "cheaper" potential) to enable larger simulations. By scaling the simulation to tens of thousands of processors, scientists can model dynamic behavior with results independent of system size, while using the most accurate semi-empirical potentials available for the first time. Nearly linear weak scaling is required to develop meso-scale and continuum level models of behavior.

3.5 MDCASK

MDCASK simulates the motion of large collections of individual atoms using the classical laws of Newtonian mechanics and electrostatics. The basic features of the code, as in any "classical" (as opposed to "quantum mechanical") molecular dynamics code, are an algorithm for the integration of the equations of motion, an inter-atomic potential, and boundary conditions and constraints.

A given problem defines initial positions for the atoms (e.g., lattices for crystalline solids). MDCASK calculates the forces on each atom using the inter-atomic potential and atom positions, updates the velocities and then obtains new positions for the atoms based on the new velocities. Each atomic material and spatial configuration type uses an inter-atomic potential, derived from atomic theory and quantum mechanical, "ab initio" calculations. The code repeats this cycle to evolve the system over time. It can use a wide variety of potentials that allow for the simulation of metals, semiconductors, insulators, glasses and other materials.

It is the specifics of atomistic behavior that gives rise to phenomena at the meso- and macroscopic scale that are in turn responsible for the wide range of material properties important for science and industry. Larger computer systems such as BG/L allow us to span the gap between the microscopic scale of individual atoms to the meso-scale, thereby providing critical validation of meso- and macroscopic models of material properties. Also, some phenomena, such as the competition between different possible lattice structures during re-solidification require very large collections of atoms to properly represent. At the largest processor counts, it will be possible to perform full, 3D simulations of hydrodynamic instabilities important to the ICF program without any of the approximations inherent in more commonly used fluid dynamics simulations.

3.6 ParaDiS

ParaDiS (for ***Para**llel **Dis**location **S**imulator*) [12] is a code developed at LLNL for direct computation of plastic strength of materials by tracking simultaneous motion of millions of dislocation lines. Simulations using ParaDiS are closing the computational performance gap long recognized to prevent physicists and materials scientists from understanding the fundamental nature of self-induced strengthening (or hardening) and the origin of intricate patterns that dislocations spontaneously form under mechanical straining. The code is primarily written in C and uses the MPI library for communication among the processors.

ParaDiS relies on a line-tracking model that only considers the defects and not the rest of the material. This reduces degrees of freedom dramatically, but it is a considerable effort to track the constantly evolving topology of the dislocation network. ParaDiS achieves a breakthrough in topology handling by using a minimal set of (irreducible) topological operators. Another challenge is a tendency of dislocation lines to cluster in space and develop highly heterogeneous distributions of degrees of freedom, making it difficult to achieve a good load balance. To maintain scalability, ParaDiS recursively partitions the problem domain and shifts the domain boundaries at regular time intervals.

Because dislocation interaction is long ranged, any two line segments interact with each other. For computational efficiency, all interactions are partitioned into *local* and *remote* contributions, based on proximity of the interacting segments. The *local* interactions are computed explicitly for each local segment pair, while the effect of all *remote* segments in a single cell are lumped together into a super-segment contribution, using a Fast Multipole algorithm. Still, evaluation of forces among dislocation segments typically takes more than 80% of compute time.

Optimizations for BG/L have simply been through the use of compiler options, although significant modifications were made to allow the code to execute with the limited memory available on the nodes of BG/L. No explicit task mapping was done to map the MPI tasks to the BG/L torus.

4. RESULTS

We present the performance of these applications on a 32K node BG/L system at LLNL. Initial runs were performed on a 16K node system at IBM Rochester.

All of these applications were compiled using the IBM XL compilers with the options *–qarch=440 –O3*. Using the highest level optimizations (-O5) generally did not improve performance, and using automatic SIMDization capability of the compiler (with *–qarch=440d*) led to performance degradation in most cases. Some applications use the MASS libraries and/or BLAS routines on BG/L, which do exploit the DFPU.

Due to the low level of computational noise introduced by the software environment, performance results on BG/L are usually reproducible, i.e., multiple runs of an application with the same input show virtually identical elapsed times [10].

Most applications in this study use the default coprocessor mode for execution, which implies a peak performance of 91.75 Teraflop/s for a 32K node system and 45.88 Teraflop/s for a 16K node system. For applications that can work with a 255 MB footprint per MPI process, it is feasible to exploit the virtual node mode that doubles the peak performance of the system.

4.1 Miranda

The initial weak scaling results for Miranda were disappointing. With a grid size of 8 x 8 x 1024 per node, a speedup of 4x was obtained in going from 512 nodes to 16K nodes in coprocessor mode, a factor of 8 below linear speedup. Instrumentation of the MPI library showed that the percentage of communication time was increasing from 34% of execution time on 512 nodes to 79% on 16K nodes, with *alltoall* communication accounting for most of that time. Improvements in the MPI_Alltoallv implementation optimized for the torus network [11] led to dramatic performance improvements, about 6x better performance for a 16K node system. Figure 1 shows recent results on the 32K node system: for a grid size of 12 x 12 x 3072 per node, a speedup of about 45x is obtained in going from 512 to 32K nodes, with communication time remaining flat at about 11% after reaching 2K nodes. Overall on LLNL's 32,768 node system, Miranda has obtained 85% of the theoretical peak speed of the BG/L torus with sustained computation rates of 1.8 Teraflop/s.

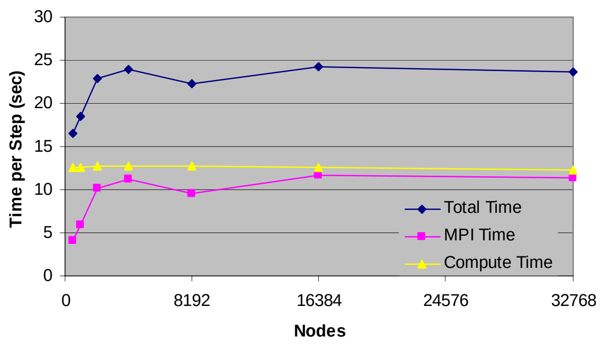

Figure 1: Miranda step timing (problem size proportional to number of BG/L nodes)

Limitations on bisection bandwidth at large scales can be a significant concern with a 3D torus topology. Miranda's scaling results demonstrate that the network bandwidth in BG/L (supported with scalable system software) is sufficient to allow scaling of some applications requiring all-to-all communication to an unprecedented level of 32,768 nodes despite the 3D torus topology.

4.2 Raptor

The experiments with Raptor have used just a single level of data across the computational domain, i.e., no adaptivity has been used. Using a data block size of 32^3 with one block per compute node, we observe a time per step (seconds per step) of 2.67 sec on 2048 processors. This is similar timing to that obtained on the MCR cluster at LLNL using two 2.2GHz Pentium 4 CPU's per node and a Quadrics Elan 3 interconnect.

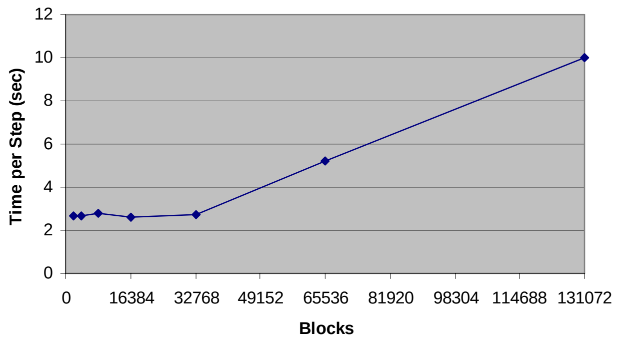

Figure 2: Raptor step timing (32,768 BG/L nodes)

Figure 2 demonstrates results for Raptor on the 32,768 node LLNL system in coprocessor mode for problems with varying number of 32^3 data blocks. Raptor demonstrates nearly perfect scaling as shown in the weak scaling portion of the graph with up to 32,768 blocks, i.e., with one block per node (remaining nodes are essentially idle) since the times are very flat. In other words, the time per step is constant as we scale up the problem size but keep the work per node constant. The remaining portion of the graph, with more than one block per node, shows the data scaling regime where the work per node increases. The data scaling performance is also very good: compared to 32,768 blocks, an efficiency of 91% is achieved with 131,072 blocks. Raptor's computational kernel has achieved 91.2 Mflop/s per node or 3.0 Teraflop/s aggregate performance on the current LLNL 32,768 node system. More importantly, Raptor's AMR capabilities will enable the solution of unprecedented problem sizes.

4.3 Qbox

As a first benchmark of Qbox on BG/L, we computed the electronic structure of a sample of bulk crystalline Molybdenum. A sample of 686 atoms including 8232 electrons was used to establish the strong scaling properties of Qbox on up to 16K nodes on the LLNL BG/L platform. Electronic wavefunctions were expanded on a plane-wave basis with an energy cutoff of 100 Rydberg. Electron-ion interactions were represented by norm-conserving, semi-local pseudopotentials.

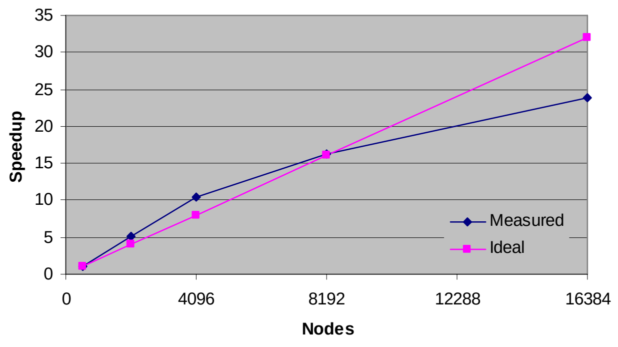

Figure 3: Qbox Strong Scaling for 686 Molybdenum Atoms

Overall performance was found to depend sensitively on the combination of task mapping to the BG/L torus, and on choices of process grids used by the ScaLAPACK library. For each BG/L partition size, several node mappings were used (by way of the BGLMPI_MAPPING environment variable). BG/L partitions have a predefined shape that cannot be modified by the user. For this reason, the node mapping leading to the best performance for a given partition size may not be optimal for another partition size. The results reported in Figure 3 show the speedup obtained with the best mapping for each partition size. An initial superlinear scaling between 512 and 4096 nodes is attributed to the effect of node mappings as well as some improved cache use on larger partitions. Speedup between 8192 and 16384 nodes reflects the imperfect scaling of some ScaLAPACK functions. A parallel efficiency of 75% is obtained on 16K nodes relative to the 512 node configuration. Qbox has demonstrated outstanding overall performance in coprocessor mode, partly through optimized libraries that provide effective utilization of the DFPU. On the 32K nodes of LLNL's current system, the code has achieved sustained performance of over 22 Teraflop/s. Current work is developing custom implementations of some ScaLAPACK functions to improve scaling beyond 32K nodes.

4.4 ddcMD

Figure 4 shows weak scaling results for ddcMD in virtual node mode. The elapsed time per particle per task per step with 500 particles per task goes up from 0.99 milliseconds at two tasks to 1.39 milliseconds at 32K tasks, for a parallel efficiency of 71%. For a larger data size, 2000 particles per processor, the elapsed time per particle per task per step increases from 0.95 seconds at 2 tasks to only 1.20 seconds at 64K tasks, representing a parallel efficiency of 79%. On LLNL's 32,768 node system in virtual node mode, ddcMD has achieved 20 Teraflop/s sustained performance, which will enable scientists to develop meso-scale and continuum level models of behavior of materials for the first time.

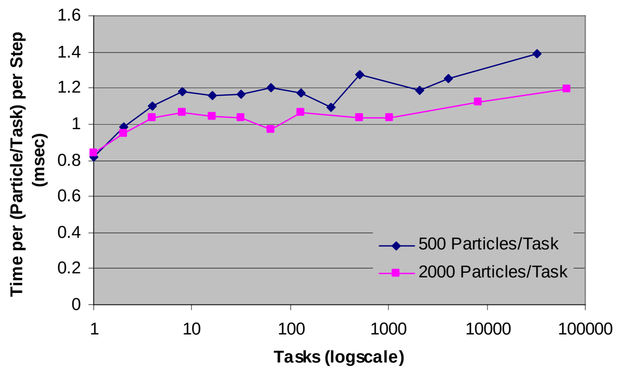

Figure 4: ddcMD Weak Scaling for High-Pressure Solidification of Tantalum

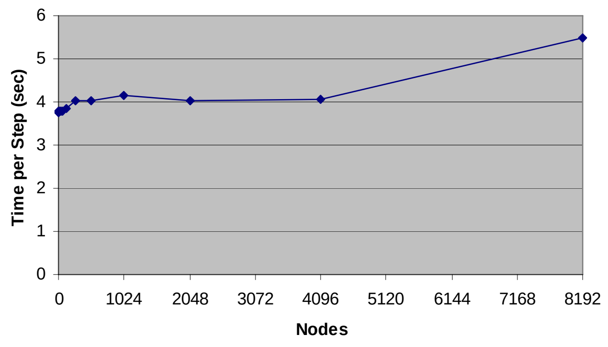

Figure 5: MDCASK Weak Scaling

4.5 MDCASK

We have conducted a weak scaling test of MDCASK in which a constant workload per processor (of about 250,000 atoms) was tested in powers of two from one node to 8K nodes. The runtime remains constant up to 4K nodes, as shown in Figure 5. Communication in MDCASK is primarily broadcast and point-to-point. Performance drops off somewhat in going to 8K nodes, while initial tests at 16K nodes have revealed scaling issues within the application. Current work is focused on removing these limitations and scaling to 64K processors.

4.6 ParaDiS

We targeted ParaDiS towards a large simulation to cover the length and time scales sufficient to observe the hardening transitions that occur naturally as a result of motion and rearrangement of dislocations. A full simulation should include from 1M to 100M dislocation segments and should be traced over millions of time steps. Line dynamics capabilities available up to now at LLNL and elsewhere stop short of these target performance figures by about 2-3 orders of magnitude.

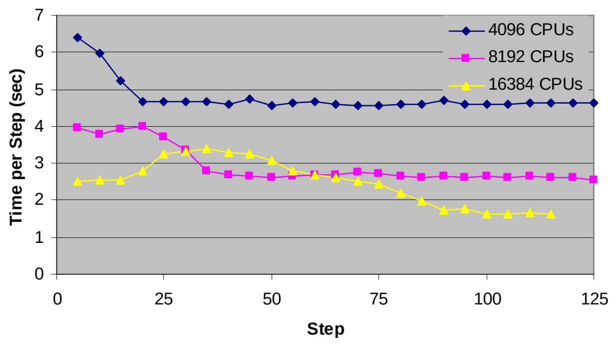

Figure 6: ParaDiS Time per Step with Constant Problem Size

The experimental results, presented in Figure 6, show strong scaling for an identical problem running an identical number of steps at processor counts of 4K, 8K and 16K BG/L nodes in co-processor mode. The runs show a speedup of 1.8x when doubling the processor count from 4K to 8K, and a speedup of 2.8x in quadrupling the processor count to 16K. The results may appear somewhat disappointing. However, our analysis indicates that the results were skewed by ParaDiS's dynamic load-balancing. Given a specific initial problem, the load-balance of the problem decreases as the number of processors is increased. ParaDiS then dynamically adjusts the load balance to settle into the optimal work distribution. However, our tests show that the load-balancing mechanism requires longer to converge as the number of processors increases. Thus, we expect greater parallel efficiency for longer runs. In addition, we are investigating techniques to achieve better load balance, which would also lead to higher parallel efficiency.

4.7 Summary

The above results show that all of the applications we studied were able to scale to 16K or 32K nodes, representing unprecedented levels of scaling. We encountered some problems in both system software and the applications, which represent opportunities for further improvement of these results. Some of the key performance issues were:

- *Poor single node performance*: We find that despite incorporating several novel techniques for automatic SIMDization [12,13], the compiler is unable to effectively exploit the second FPU of the DFPU for the applications used in this study. Applications that use tuned BLAS routines and MASS libraries are able to benefit from the second FPU for portions that use those libraries.

- *All-to-all communication*: The scalability of many applications is limited by the performance of the all-to-all communication. In particular, improvements in the performance of the MPI_Alltoallv primitive led to dramatic improvements in the scaling of the Miranda code.

- *Mapping of application topology*: For some applications such as Miranda and Qbox, the performance is sensitive to how the application tasks are mapped to the 3D torus topology. A good mapping reduces the average number of hops traversed by messages, enhancing the effective bandwidth for communication (since fewer messages need to share the same link) and reducing the latency of messages.

- *Load imbalance*: For the ParaDiS application, load imbalance in the application is the primary inhibitor to better scaling.

On BG/L, we did not encounter scaling problems due to computational noise, which often manifest themselves in the form of poor scaling of collective communication operations like *allreduce*. Given the powerful networks and good message passing performance on BG/L, communication overhead for all of the applications we studied remained modest even at the levels of 16K and 32K processors.

5. CONCLUSIONS

In this paper, we have presented our experience so far with several physics and materials applications on a 32,768 node BG/L system. Our study shows that these applications can effectively scale to the large BG/L system with a modest level of additional effort. We have obtained the highest level of performance ever delivered for all of these applications, enabling new computational capabilities for science, and in the process, have crossed the threshold of successful scaling of scientific applications to over ten thousand processors. These results also validate the scalability of the hardware and the software architecture of Blue Gene/L. We expect to improve these results further via software enhancements. We also plan to pursue the scaling of applications to over a hundred thousand processors in the near future, and using that capability to help solve grand challenge problems of scientific importance.

6. ACKNOWLEDGMENTS

We thank the entire Blue Gene/L team for creating the platform that made this work possible.

7. REFERENCES

[1] N. R. Adiga et al. An overview of the BlueGene/L supercomputer. In *SC2002 – High Performance Networking and Computing*, Baltimore, MD, November 2002.

[2] S. Habata, M. Yokokawa, S. Kitawaki. The Earth Simulator. In *NEC Research and Development*, 44(1), January 2003.

[3] ASC Purple: Fifth Generation ASC Platform. http://www.llnl.gov/asci/platforms/purple/.

[4] NASA Columbia Supercomputer. http://www.nasa.gov/centers/ames/research/lifeonearth/lifeonearth-projectColumbia.html.

[5] G. Almasi, R. Bellofatto, J. Brunheroto, C. Cascaval, J. Castaños, L. Ceze, P. Crumley, C. Erway, J. Gagliano, D. Lieber, X. Martorell, J. Moreira, A. Sanomiya, and K. Strauss. An Overview of the BlueGene/L System Software Organization (Distinguished Paper). *Proceedings of the 2003 International Conference on Parallel and Distributed Computing (Euro-Par 2003).* August 26-29, 2003. Klagenfurt, Austria. pp. 543-555.

[6] G. Almasi, S. Chatterjee, A. Gara, J. Gunnels, M. Gupta, A. Henning, J. Moreira, B. Walkup, A. Curioni, C. Archer, L. Bachega, B. Chan, B. Curtis, S. Brunett, G. Chukkapalli, R. Harkness, W. Pfeiffer. Unlocking the Performance of the BlueGene/L Supercomputer. *SC 2004: High Performance Computing, Networking and Storage Conference*, Pittsburgh, PA, November 2004.

[7] F. Petrini, D. Kerbyson and S. Pakin. The Case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q. In *IEEE/ACM SC2003*, Phoenix, AZ, November 2003.

[8] TOP500 Supercomputer Sites, http://www.top500.org.

[9] L. Bachega, S. Chatterjee, K. Dockser, J. Gunnels, M. Gupta, F. Gustavson, C. Lapkowski, G. Liu, M. Mendell, C. Wait, T.J.C. Ward. A High-Performance SIMD Floating Point Unit Design for BlueGene/L: Architecture, Compilation, and Algorithm Design. *Parallel Architecture and Compilation Techniques (PACT 2004)*, Antibes Juan-les-Pins, France, Sept-Oct 2004.

[10] K. Davis, A. Hoisie, G. Johnson, D. Kerbyson, M. Lang, S. Pakin and F. Petrini. A Performance and Scalability Analysis of the BlueGene/L Architecture. In *IEEE/ACM SC2004*, Pittsburgh, PA, November 2004.

[11] G. Almasi, C. Archer, J. Castanos, C. Erway, J. Gunnels, P. Heidelberger, X. Martorell, J. Moreira, K. Pinnow, J. Ratterman, B. Steinmacher-burow, W. Gropp, B. Toonen. Design and implementation of message passing services for the BlueGene/L supercomputer. *IBM Journal of Research and Development*, No 2/3, 2005.

[12] A. Eichenberger, P. Wu, and K.O'Brien. Vectorizing for Short SIMD Architectures with Alignment Constraints. *ACM SIGPLAN conference of Programming Languages Design and Implementation (PLDI'04),* Washington DC, June, 2004.

[13] P. Wu, A. Eichenberger, and A. Wang. Efficient Code Generation for Runtime Alignment and Length Conversion. *Code Generation and Optimization (CGO'05),* March, 2005.