CFD on the BlueGene/L Supercomputer

Suga A. Sugavanam*
High Performance Computing, IBM Server Group
2455, South Road, Poughkeepsie, NY 12601, USA

The BlueGene/L is a supercomputer developed jointly by IBM, and Lawrence Livermore National Laboratory as a part of the United States Department of Energy Accelerated Strategic Computing Initiative (ASCI) Advanced Architecture Research Program (Ref. 1). This massively parallel system exploits system-on-a-chip technology to deliver a target peak of 360 teraFLOPS (trillion floating-point operations per second). The machine is scheduled to be operational in the 2004-2005 timeframe. The primary advantages of this system are price/performance, and power consumption/performance. This system is mainly targeted for the discipline of life sciences – molecular dynamics computations. This paper will address the suitability of this system for large problems in computational fluid dynamics (CFD). The following sections will describe the system hardware, the system software, programming considerations, characteristics of some very large problems in CFD, and the examination of some specific codes on such a system.

System Hardware

BlueGene/L (BG/L) is a scalable system with a maximum number of compute nodes set to be 2**16 = 65,536 nodes. Each node consists of a single ASIC and maximum memory of 2 GB. The system utilizes IBM PowerPC 440 FP2 processor targeted at a clock rate of 700 MHz. The current design calls for 2 nodes per compute card, 16-compute cards per node board, 16 node boards per 512-node midplane, 2 midplanes in a 1024-node rack. Each processor can perform 4 floating-point operations per cycle in the form of two 64-bit multiply-add per cycle. At the target frequency this amounts to approximately 1.4 teraFLOPS peak performance for a single midplane of BG/L, if we count only a single processor per node. Each node contains a second processor, identical to the first although not included in the 1.4 teraFLOPS performance number, intended primarily for handling message-passing operations. In addition, the system provides for a flexible number of additional dual-processor I/O nodes, up to a maximum of one I/O node for every eight-compute nodes. Each compute node executes a lightweight kernel. The compute node kernel handles basic communication tasks and all the functions necessary for high performance scientific code. An I/O node handles communication between a compute node and other systems, including the host and file servers. The choice of host will depend on the class of applications and their bandwidth and performance requirements.

The nodes are interconnected through five networks: a 3D torus network for point-to-point messaging between compute nodes, a global combining/broadcast tree for collective operations such as MPI_Allreduce over the entire application, a global barrier and interrupt network, a Gigabit Ethernet to JTAG network for machine control, and another Gigabit Ethernet network for connection to other systems, such as hosts and file systems. For cost and overall system efficiency, compute nodes are not hooked directly up to the Gigabit Ethernet, but rather use the global tree for communicating with their I/O nodes, while the I/O nodes use the Gigabit Ethernet to communicate with other systems.

System fault tolerance is a critical aspect the BlueGene/L machine. BlueGene/L will have many layers of fault tolerance that are expected to allow for good availability despite the large number of system nodes. In addition, the BlueGene/L platform will be used to investigate many avenues in autonomic computing.

The memory system is being designed for high bandwidth, low latency memory and cache accesses. An L2 hit returns in 6 to 10 processor cycles, an L3 hit in about 25 cycles, and an L3 miss in about 75 cycles. L3 misses are serviced by external memory, the system in design

*E-mail: suga@us.ibm.com

has a 16 byte interface to nine 256Mb SDRAM-DDR devices operating at a speed of one half or one third of the processor.

Operating System Architecture

The goal in developing the system software for BG/L is to create an environment, which looks familiar and also delivers high levels of application performance. The applications get a feel of executing in an Unix-like environment.

The approach adopted is to split the operating system functionality between compute and I/O nodes. Each compute node is dedicated to the execution of a single application process. The I/O node provides the physical interface to the file system. The I/O nodes are also available to run processes, which facilitate control, bring-up, job launch and debug of the BlueGene/L machine. The approach allows the compute node software to be kept very simple.

The compute node operating system, also called the BlueGene/L compute node kernel, is a simple, lightweight, single-user operating system that supports execution of a single dual-threaded application compute process. Each thread of the compute process is bound to one of the processors in the compute node. The compute node kernel is complemented by a user-level runtime library that provides the compute process with direct access to the torus and tree networks. Together, kernel and runtime library implement compute node-to-compute node communication through the torus network and compute node-to-I/O node communication through the tree network. Compute node-to-I/O node communication is used primarily/ for extending the compute process into an I/O node, so that it can perform services available only in that node.

I/O nodes are expected to run the Linux operating system, supporting the execution of multiple processes. Only system software is run on the I/O nodes, and the application code is not executed on these. The purpose of the I/O nodes during application execution is to complement the compute node partition with services that are not provided by the compute node software. I/O nodes provide an actual file system to the running applications. When a compute process in a compute node performs an I/O operation (on a file or a socket), that I/O operation (e.g., a read or a write) is shipped through the tree network to a service process in the I/O node. That service process then issues the operation against the I/O node operating system. The results of the operation (e.g., return code in case of write, actual data in case of read) are shipped back to the originating compute node. The I/O node also performs process authentication, accounting, and authorization on behalf of its compute nodes.

I/O nodes also provide debugging capability for user applications. Debuggers running on an I/O node can debug application processes running on the compute nodes. In this case, the shipping occurs in the opposite direction. Debugging operations performed on the I/O nodes are shipped to the compute node for execution against the compute process. Results are shipped back to the debugger in the I/O node.

Programming Models

Message passing is expected to be the dominant parallel programming model for BG/L applications. It is supported through an implementation of the MPI message-passing library. In developing MPI for BG/L, attention is paid to the issue of efficient mapping of operations to the torus and tree networks. Also important is the issue of efficient use of the second (communication) processor in a compute node.

Control System

The BG/L system software includes a set of control services that execute on the host system. Many of these services, including system bring up, machine partitioning, measuring system performance, and monitoring system health, are nonarchitected from a user perspective, and are performed through the backdoor JTAG. The resource management system of BG/L provides services to create electronically isolated partitions of the machine and to allocate

resources to jobs. Each partition is dedicated to the execution of a single job at a time. The host also performs job scheduling and job control.

Advances in Parallel CFD

In the past dozen years or so, dramatic advances have been made in solving complex real life problems in CFD with structured, multi-block structured and unstructured grid codes. Most commercial CFD codes, proprietary codes, and the public domain (NASA, and government developed) codes have embraced both the shared memory and distributed memory models of parallel computing. Few of the CFD codes exploit both models. We rarely hear yesteryear limitations of distributed memory systems for CFD any more with the plethora of Linux clusters in the CFD marketplace. Parallel algorithms for sparse solvers [Ref. 2], and advanced domain decomposition techniques that can assign weights for the heterogeneous computing elements or account for variety of physics with differing computational work load [Ref. 3] are beginning to change the CFD computing landscape. Acceptance of new paradigms such as the Grid computing, fault tolerant autonomic computing, and on-demand computing are already changing the way in which CFD is practiced today in the industry. Emphasis on interaction aerodynamics, and early detailed design require full scale models in the aerospace industry to be analyzed with much more geometric details requiring between 15 to 25 million of grids or vertices. Some of these simulation run for many hours if not days on dozens of processors of existing computing systems. So, parallel CFD simulations on few dozen processors are the routine in the aerospace and the automotive industry. Is CFD ready to go beyond that? Can we run practical CFD codes on hundreds or thousands of processors?

The BlueGene/L and CFD

BlueGene/L is a machine under development, and performance measurements of real codes are not possible now. However, based on the details of timings of computations, and communication characteristics of large-scale parallel CFD codes, performance projections can be made for this system. This paper will discuss many of the challenges of parallel computing on thousands of processors, such as performance of parallel algorithms, specific issues of communication overhead, and load balancing. In addition, this paper will examine the performance of some existing CFD codes on BlueGene/L. This will include the examination of the NASA unstructured FUN3D aerodynamics code [Ref. 4], and the University of Minnesota sPPM code for gas dynamics simulations [Ref. 5]. These codes were chosen since they have been run on thousands of processors on the ASCI-Blue, and the ASCI-Red systems. The paper will also discuss serial and parallel tuning necessary for high performance on the BlueGene/L, domain decomposition challenges in large scale computing, implications of adaptive meshing, and the future of parallel CFD.

References

1. The BlueGene/L Team “An Overview of the BlueGene/L Supercomputer”, Supercomputing 2002, Technical Papers, November 2002.
2. Gupta, A., Joshi, M., Kumar, V., “WSSMP – A High Performance Serial and Parallel Symmetric Sparse Linear Solver, PARA 98 – Workshop on Applied Parallel Computing in Large Scale Scientific and Industrial Problems, Umea, Sweden, June, 1998.
3. Schloegel, K., Karypis, G., and Kumar, V “A Unified Algorithm for Load-balancing Adaptive Scientific Simulations, Department of Computer Science and Engineering, University of Minnesota, Technical report TR 00-033
4. Gropp, W.D., Kaushik, D.K., Keyes, D.E., Smith, B.F., “Performance Modeling and Tuning of an Unstructured Mesh CFD Application”, in: Proceedings of the SC2000, IEEE Computer Society, 2000.
5. Colella, P., and Woddward, P.R. “The Piecewise Parabolic Methods (PPM) for Gas-Dynamical Simulations”, J. Comp. Physics, 54, 174, 1984.