RC25151 (W1104-096) April 20, 2011
Computer Science

IBM Research Report

IBM Blue Gene Supercomputer

Alan Gara, José E. Moreira
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

IBM Blue Gene Supercomputer

Alan Gara
IBM Fellow

alangara@us.ibm.com

José E. Moreira
Research Staff Member
jmoreira@us.ibm.com

IBM T.J. Watson Research Center
1101 Kitchawan Rd., Yorktown Heights, NY 10598, USA

DEFINITION

The IBM Blue Gene Supercomputer is a massively parallel system based on the PowerPC processor. A Blue Gene/L system at Lawrence Livermore National Laboratory held the number 1 position in the TOP500 list of fastest computers in the world from November 2004 to November 2007.

DISCUSSION

Introduction

The IBM Blue Gene Supercomputer is a massively parallel system based on the PowerPC processor. It derives its computing power from scalability and energy efficiency. Each computing node of Blue Gene is optimized to achieve high computational rate per unit of power and to operate with other nodes in parallel. This approach results in a system that can scale to very large sizes and deliver substantial aggregate performance.

Most large parallel systems in the 2000-2010 time frame followed a model of using off-the-shelf processors (typically from Intel, AMD or IBM) and interconnecting them with either an industry standard network (e.g., Ethernet or Infiniband) or a proprietary network (e.g., as used by Cray or IBM). Blue Gene took a different approach by designing a dedicated system-on-a-chip (SoC) that included not only processors optimized for floating-point computing but also the networking infrastructure to interconnect these building blocks into a large system. This customized approach led to the scalability and power efficiency characteristics that differentiated Blue Gene from the other machines that existed at the time of its commercial introduction in 2004.

As of 2011, IBM has produced two commercial versions of Blue Gene, Blue Gene/L and Blue Gene/P, which were first delivered to customers in 2004 and 2007, respectively. A third version, Blue Gene/Q, was under development. Both delivered versions follow the same design principles, the same system architecture and the same software architecture. They differ on the specifics of the basic SoC that serves as the building block for Blue Gene. The November 2010 TOP500 list includes four Blue Gene/L system and ten Blue Gene/P systems (and one prototype Blue Gene/Q system). In this article we cover mostly the common aspects of both versions of the Blue Gene supercomputing and discuss details specific to each version as appropriate.

System architecture

A Blue Gene system consists of a *compute section*, a *file server section* and a *host section* (Figure 1). The compute and I/O nodes in the compute section form the computational core of Blue Gene. User jobs run in the compute nodes, while the I/O nodes connect the compute section to the file servers and front-end nodes through an Ethernet network. The file server section consists of a set of file servers. The host section consists of a *service node* and one or more *front-end nodes*. The service node controls the compute section through an Ethernet control network. The front-end nodes provide job compilation, job launch and job debugging services.

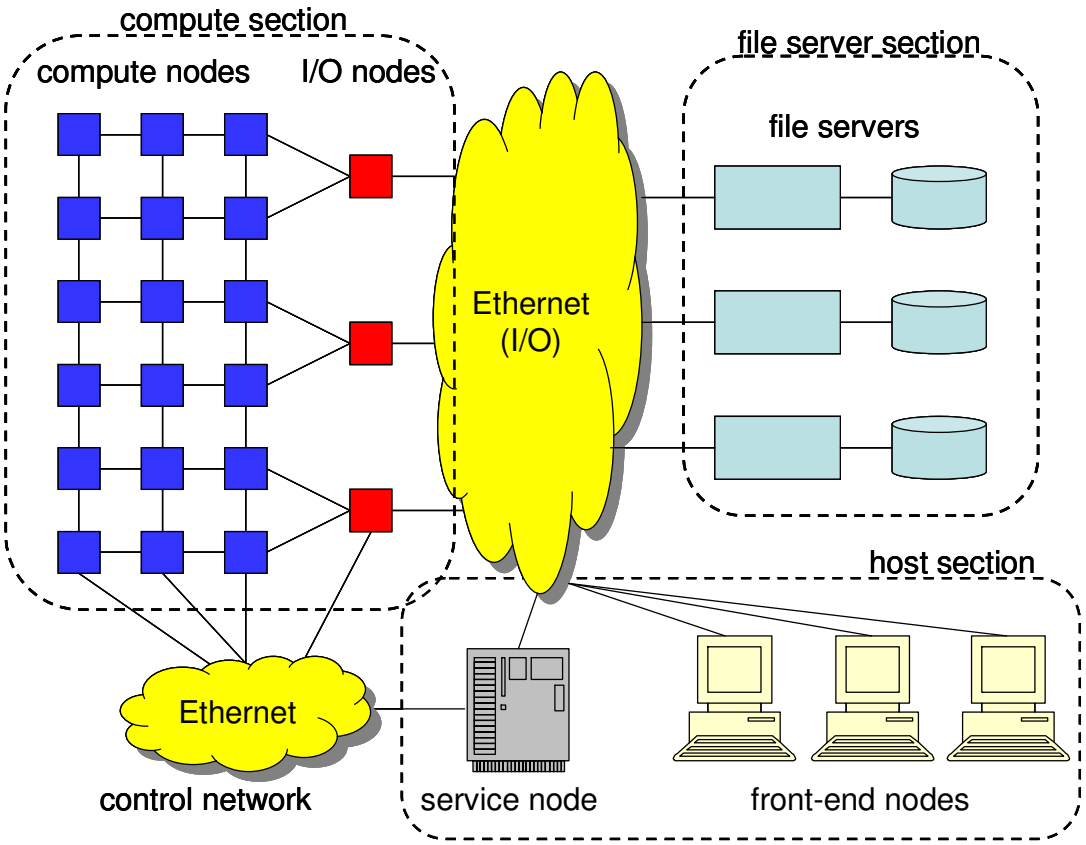


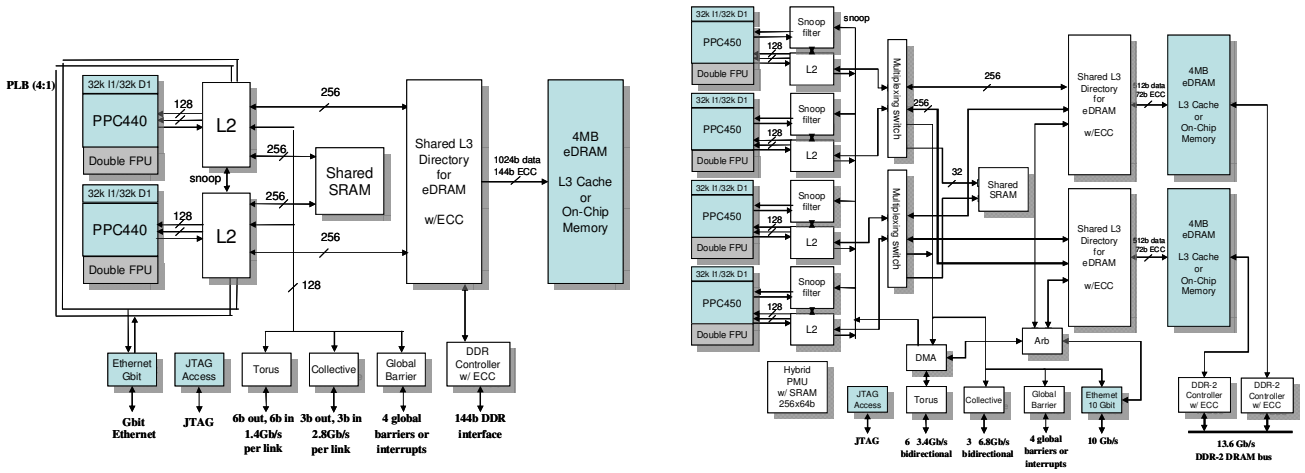
Figure 1: High-level view of a Blue Gene system.

Compute section

The compute section of Blue Gene is what is usually called a Blue Gene machine. It consists of a three-dimensional array of compute nodes interconnected in a toroidal topology along the *x*, *y*, and *z* axes. I/O nodes are distinct from compute nodes and not in the toroidal interconnect, but also belong to the compute section. A collective network interconnects all I/O and compute nodes of a Blue Gene machine. Each I/O node communicates outside of the machine through an Ethernet link.

Compute and I/O nodes are built out of the same Blue Gene compute ASIC (application specific integrated circuit) and memory (DRAM) chips. The difference is in the function they perform. Whereas compute nodes connect to each other for passing application data, I/O nodes form the interface between the compute nodes and the outside world by connecting to an Ethernet network. Reflecting the difference in function, the software stacks of the compute and I/O nodes are also different, as will be discussed below.

The particular characteristics of the Blue Gene compute ASIC for the two versions (Blue Gene/L and Blue Gene/P) are illustrated in Figure 2 and summarized in Table 1. The Blue Gene/L compute ASIC contains two non-memory coherent PowerPC 440 cores, each with private L1 data and instruction caches (32 KiB each). Associated with each core is a small (2 KiB) L2 cache that acts as a prefetch egine. Completing the on-chip memory hierarchy is 4 MiB of embedded DRAM (eDRAM) that is configured to operate as a shared L3 cache. Also on the ASIC is a memory controller (for external DRAM) and interfaces to the five networks used to interconnect Blue Gene/L compute and I/O nodes: torus, collective, global barrier, Ethernet, and control network.



(a) (b)

Figure 2: Blue Gene compute ASIC for Blue Gene/L (a) and Blue Gene/P (b).

The Blue Gene/P compute ASIC contains four memory coherent PowerPC 450 cores, each with private L1 data and instruction caches (32 KiB each). Associated with each core are both a small (2 KiB) L2 cache that acts as a prefetch engine, as in Blue Gene/L, and a snoop filter to reduce coherence traffic into each core. Two banks of 4 MiB EDRAM are configured to operate as a shared L3 cache. Completing the ASIC are dual memory controller (for external DRAM), interfaces to the same five networks as Blue Gene/L and a DMA engine to transfer data directly from the memory of one node to another.

Both the PowerPC 440 and PowerPC 450 cores include a vector floating-point unit that extends the PowerPC instruction set architecture with instructions that ad in matrix and complex-arithmetic operations [15]. The vector floating-point unit can perform two fused multiply-add operations per second for a total of four floating-point operations per cycle per core.

The interconnection networks are primarily used for communication primitives used in parallel high-performance computing applications. The main interconnection network is the torus network, which provides point-to-point and multicast communication across compute nodes. A collective network in a tree topology interconnects all compute and I/O nodes and supports efficient collective operations, such as reductions and broadcasts. Arithmetic and logical operations are implemented as part of the communication primitives to facilitate low-latency collective operations. A global barrier network supports fast synchronization and notification across compute and I/O nodes. The Ethernet network is used to connect the I/O nodes to outside the machine, as previously discussed. Finally, the control network is used to control the hardware from the service node. Figure 3 illustrates the topology of the various networks.

Table 1: Summary of differences between Blue Gene/L and Blue Gene/P nodes.

Node characteristics	***Blue Gene/L***	***Blue Gene/P***
Processors	Two PowerPC 440	Four PowerPC 450
Processor frequency	700 MHz	850 MHz
Peak computing capacity	5.6 Gflop/s (4 flops/cycle/core)	13.6 Gflop/s (4 flops/cycle/core)
Main memory capacity	512 MiB or 1 GiB	2 GiB or 4 GiB
Main memory bandwidth	5.6 GB/s (16 byte @ 350 MHz)	13.6 GB/s (2 $\times$ 16 byte @ 425 MHz)
Torus link bandwidth (6 links)	175 MB/s per direction	425 MB/s per direction
Collective link bandwidth (3 links)	350 MB/s per direction	850 MB/s per direction

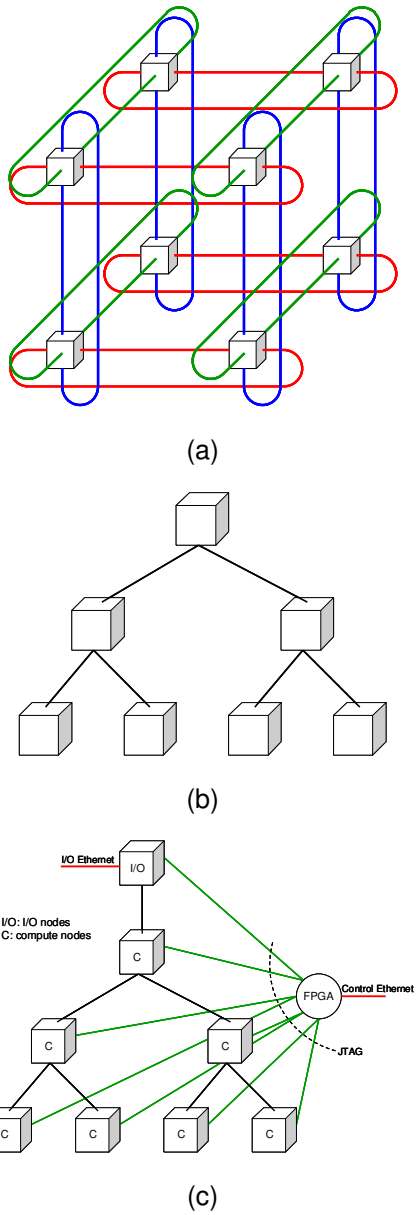


Figure 3: Blue Gene networks. Three-dimensional torus (a), collective network (b), control network and Ethernet (c).

Compute and I/O nodes are grouped into units of *midplanes*. A midplane contains 16 node cards and each node card contains 32 compute nodes and up to 4 (Blue Gene/L) or 2 (Blue Gene/P) I/O nodes. The 512 compute nodes in a midplane are arranged as an $8 \times 8 \times 8$ three-dimensional mesh. Midplanes are typically configured with 8 to 32 (Blue Gene/P) or 64 (Blue Gene/L) I/O nodes. Each midplane also has 24 link chips used for inter-midplane connection to build larger systems. The link chips also implement the "closing" of the toroidal interconnect, by connecting the two ends of a dimension. Midplanes are arranged two to a rack, and racks are arranged in a two-dimensional layout of rows and columns. Because the midplane is the basic replication unit, the dimensions the array of compute nodes must be a multiple of 8.

The configuration of the original Blue Gene/L system at Lawrence Livermore National Laboratory is shown in Figure 4. That machine was later upgraded to 104 racks and is the largest Blue Gene system delivered to date. The highest performing Blue Gene system delivered to date is a 72-rack Blue Gene/P system at Juelich Research Center. Several single-rack (1024 compute nodes) Blue Gene systems, as well as systems of intermediate size, were also delivered to various customers.

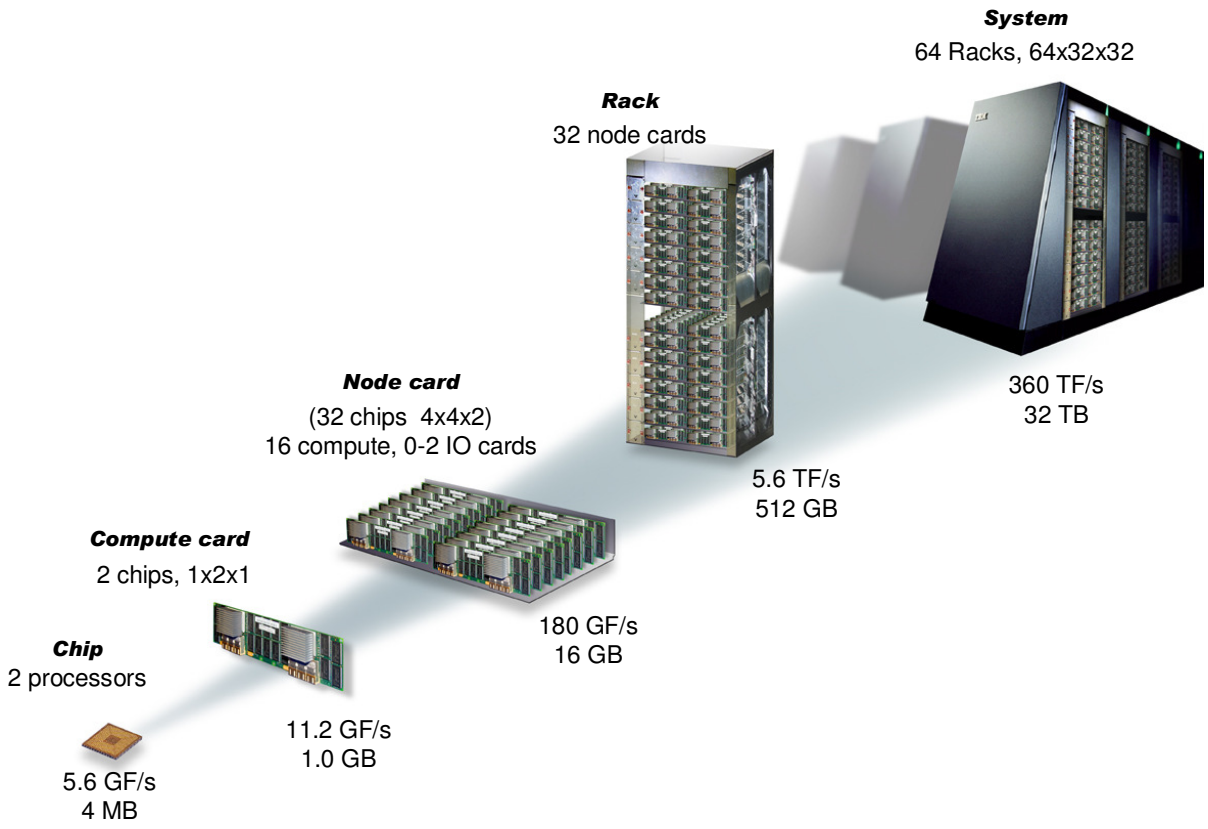


Figure 4: Packaging hierarchy for the original Blue Gene/L at Lawrence Livermore National Laboratory.

A given Blue Gene machine can be partitioned along midplane boundaries. A partition is formed by a rectangular arrangement of midplanes. Each partition can run one and only one job at any given time. During each job, all the compute nodes of a partition stay in the same execution mode for the duration of the job. These modes of execution are described in more detail below.

File Server Section

The file server section of a Blue Gene system provides the storage for the file system that runs on the Blue Gene I/O nodes. Several parallel file systems have been ported to Blue Gene, including GPFS, PVFS2, and Lustre. To feed data to a Blue Gene system, multiple servers are required to achieve the required bandwidth. The original Blue Gene/L system at LLNL, for example, uses 224 servers operating in parallel. Data is striped across those servers, and a multi-level switching Ethernet network is used to connect the I/O nodes to the servers. The servers themselves are standard rack-mounted machines, typically Intel, AMD or POWER processor based.

Host Section

The host section for a Blue Gene/L system consists of one service node and one or more front-end nodes. These nodes are standard POWER processor machines. For the LLNL machine, the service node is a 16-processor POWER4 machine, and each of the 14 front-end nodes is a Power-PC 970 blade.

The service node is responsible for controlling and monitoring the operation of the compute section. The services it implements include: machine partitioning, partition boot, application launch, standard I/O routing, application signaling and termination, event monitoring (for events generated by the compute and I/O nodes), and environmental monitoring (for things like power supply voltages, fan speeds, and temperatures).

The front-end nodes are where users work. They provide access to compilers, debuggers and job submission services. Standard I/O from user applications are routed to the submitting front-end node.

Blue Gene System Software

The primary operating system for Blue Gene compute nodes is a lightweight operating system called the Compute Node Kernel (CNK). This simple kernel implements only a limited set of services, which are complemented by services provided by the I/O nodes. The I/O nodes, in turn, run a version of the Linux operating system. The I/O nodes act as gateways between the outside world and the compute nodes, complementing the services provided by CNK with file and socket operations, debugging, and signaling.

This split of functions between I/O and compute nodes, with the I/O nodes dedicated to system services and the compute nodes dedicated to application execution, resulted in a simplified design for both components. It also enables Blue Gene scalability and robustness, and achieves a deterministic execution environment.

Scientific middleware for Blue Gene includes a user-level library implementation of the MPI standard, optimized to take advantage of Blue Gene networks, and various math libraries, also in user level. Implementing all the message passing functions in user mode simplifies the supervisor (kernel) code of Blue Gene and results in better performance by reducing the number of kernel system calls that an application performs.

Overall Operating System Architecture

A key concept in the Blue Gene operating system solution is the organization of compute and I/O nodes into logical entities called processing sets or psets. A pset consists of one I/O node and a collection of compute nodes. Every system partition, in turn, is organized as a collection of psets. All psets in a partition must have the same number of compute nodes, and the psets of a partition must cover all the I/O and compute nodes of the partition. The psets of a partition never overlap. The supported pset sizes are 8 (Blue Gene/L only), 16, 32, 64, and 128 compute nodes, plus the I/O node. The psets are a purely logical concept implemented by the Blue Gene system software stack. They are built to reflect the topological proximity between I/O and compute nodes, thus improving communication performance within a pset.

A Blue Gene job consists of a collection of N compute processes. Each process has its own private address space and two processes of the same job communicate only through message passing. The primary communication model for Blue Gene is MPI. The N compute processes of a Blue Gene job correspond to tasks with ranks 0 to $N - 1$ in the MPI COMM WORLD communicator. Compute processes run only on compute nodes; conversely, compute nodes run only compute processes.

In Blue Gene/L, the CNK implements two modes of operation for the compute nodes: coprocessor mode and virtual node mode. In coprocessor mode, the single process in the node has access to the entire node memory. One processor executes user code while the other performs communication functions. In virtual node mode, the node memory is split in half between the two processes running on the two processors. Each process performs both computation and communication functions.

In Blue Gene/P, the CNK provides three modes of operation for the compute nodes: SMP mode, dual mode and quad mode. SMP mode supports a single multithreaded (up to 4 threads) application process per compute node. That process has access to the entire node memory. Dual mode supports two multithreaded (up to 2 threads each) application processes per compute node. Quad mode supports four single-threaded application processes per compute node. In dual and quad modes, the application processes in a node split the memory of that node. Blue Gene/P supports at most one application thread per processor. It also supports an optional communication thread in each processor.

Each I/O node runs one image of the Linux operating system. It can offer the entire spectrum of services expected in a Linux box, such as multiprocessing, file systems, and a TCP/IP communication stack. These services are used to extend the capabilities of the compute node kernel, providing a richer functionality to the compute processes. Due to the lack of cache coherency between the processors of a Blue Gene/L node, only one of the processors of each I/O node is used by Linux, while the other processor remains idle. Since the processors in a Blue Gene/P node are cache coherent, Blue Gene/P I/O nodes can run a true multiprocessor Linux instance.

The Compute Node Kernel

CNK is a lean operating system that performs a simple sequence of operations at job start time. This sequence of operations happens in every compute node of a partition, at the start of each job:

1. It creates the address space(s) for execution of compute process(es) in a compute node.
2. It loads code and initialized data for the executable of that (those) process(es).
3. It transfers processor control to the loaded executable, changing from supervisor to user mode.

The address spaces of the processes are flat and fixed, with no paging. The entire mapping is designed to fit statically in the TLBs of the PowerPC processors. The loading of code and data occurs in push mode. The I/O node of a pset reads the executable from the file system and forwards it to all compute nodes in the pset. The CNK in a compute node receives that executable and stores the appropriate memory values in the address space(s) of the compute process(es).

Once the CNK transfers control to the user application, its primary mission is to "stay out of the way." In normal execution, processor control stays with the compute process until it requests an operating system service through a system call. Exceptions to this normal execution are caused by hardware interrupts: either timer alarms requested by the user code, communication interrupts caused by arriving packets, or an abnormal hardware event that requires attention by the compute node kernel.

When a compute process makes a system call, three things may happen:

1. "Simple" system calls that require little operating system functionality, such as getting the time or setting an alarm, are handled locally by the compute node kernel. Control is transferred back to the compute process at completion of the call.
2. "I/O" system calls that require infrastructure for file systems and IP stack are shipped for execution in the I/O node associated with that compute node (that is, the I/O node in the pset of the compute node). The compute node kernel waits for a reply from the I/O node, and then returns control back to the compute process.
3. "Unsupported" system calls that require infrastructure not present in Blue Gene are returned right away with an error condition.

There are two main benefits from the simple approach for a compute node operating system: robustness and scalability. Robustness comes from the fact that the compute node kernel performs few services, which greatly simplifies its design, implementation, and test. Scalability comes from lack of interference with running compute processes.

System software for the I/O Node

The I/O node plays a dual role in Blue Gene. On one hand, it acts as an effective master of its corresponding pset. On the other hand, it services requests from compute nodes in that pset. Jobs are launched in a partition by contacting corresponding I/O nodes. Each I/O node is then responsible for loading and starting the execution of the processes in each of the compute nodes of its pset. Once the compute processes start running, the I/O nodes wait for requests from those processes. Those requests are mainly I/O operations to be performed against the file systems mounted in the I/O node.

Blue Gene I/O nodes execute an embedded version of the Linux operating system. It is classified as an embedded version because it does not use any swap space, it has an in-memory root file system, it uses little memory, and it lacks the majority of daemons and services found in a server-grade configuration of Linux. It is, however, a complete port of the Linux kernel and those services can be, and in various cases have been, turned on for specific purposes. The Linux in Blue Gene I/O nodes includes a full TCP/IP stack, supporting communications to the outside world through Ethernet. It also includes file system support. Various network file systems have been ported to the Blue Gene/L I/O node, including GPFS, Lustre, NFS, and PVFS2.

Blue Gene/L I/O nodes never run application processes. That duty is reserved to the compute nodes. The main user-level process running on the Blue Gene/L I/O node is the control and I/O daemon (CIOD). CIOD is the process that links the compute processes of an application running on compute nodes to the outside world. To launch a user job in a partition, the service node contacts the CIOD of each I/O node of the partition and passes the parameters of the job (user ID, group ID, supplementary groups, executable name, starting working directory, command line arguments, and environment variables). CIOD swaps itself to the user's identity, which includes the user ID, group ID, and supplementary groups. It then retrieves the executable from the file system and sends the code and initialized data through the collective network to each of the compute nodes in the pset. It also sends the command-line arguments and environment variables, together with a start signal.

Figure 5 illustrates how I/O system calls are handled in Blue Gene. When a compute process performs a system call requiring I/O (e.g., open, close, read, write), that call is trapped by the compute node kernel, which packages the parameters of the system call and sends that message to the CIOD in its corresponding I/O node. CIOD unpacks the message and then reissues the system call, this time under the Linux operating system of the I/O node. Once the system call completes, CIOD packages the result and sends it back to the originating compute node kernel, which, in turn, returns the result to the compute process.

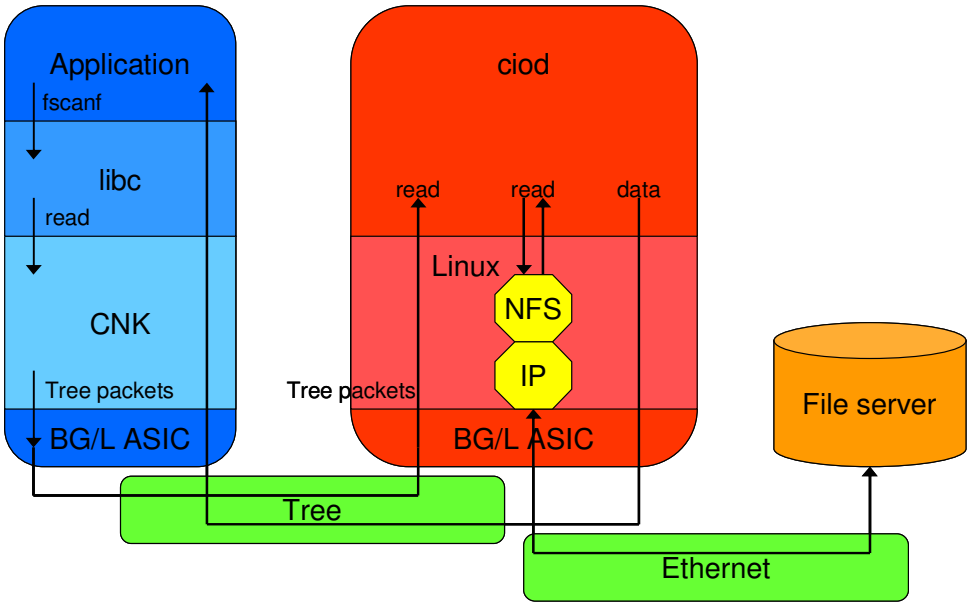


Figure 5: Function shipping between CNK and CIOD.

There is a synergistic effect between simplification and separation of responsibilities. By offloading complex system operations to the I/O node, Blue Gene keeps the compute node operating system simple. Correspondingly, by keeping application processes separate from I/O node activity, it avoids many security and safety issues regarding execution in the I/O nodes. In particular, there is never a need for the common scrubbing daemons typically used in Linux clusters to clean up after misbehaving jobs. Just as keeping system services in the I/O nodes prevents interference with compute processes, keeping those processes in compute nodes prevents interference with system services in the I/O node. This isolation is particularly helpful during performance debugging work. The overall simplification of the operating system has enabled the scalability, reproducibility (performance results for Blue Gene applications are very close across runs), and high-performance of important Blue Gene/L applications.

System Software for the Service Node

The Blue Gene service node runs its control software, typically referred to as the Blue Gene control system. The control system is responsible for operation and monitoring of all compute and I/O nodes. It is also responsible for other hardware components such as link chips, power supplies, and fans. Tight integration between the Blue Gene control system and the I/O and compute nodes operating systems is central to the Blue Gene software stack. It represents one more step in the specialization of services that characterize that stack.

In Blue Gene, the control system is responsible for setting up system partitions and loading initial code and state in the nodes of a partition. The Blue Gene compute and I/O nodes are completely stateless: no hard drives and no persistent memory. When a partition is created, the control system programs the hardware to isolate that partition from others in the system. It computes the network routing for the torus, collective, and global interrupt networks, thus simplifying the compute node kernel. It loads operating system code for all compute and I/O nodes of a partition through the dedicated control network. It also loads an initial state in each node (called the personality of the node). The personality of a node contains information specific to the node.

Blue Gene and its Impact on Science

Blue Gene was designed to deliver a level of performance for scientific computing that enables entire new studies and new applications. One of the main areas of applications of the original LLNL system is in materials science. Scientists at LLNL use a variety of models, including quantum molecular dynamics, classical molecular dynamics, and dislocation dynamics, to study materials at different levels of resolution. Typically, each model is applicable to a range of system sizes being studied. First principle models can be used for small systems, while more phenomenological models have to be used for large systems. The original Blue Gene/L at LLNL was the first system that allowed crossing of those boundaries. Scientists can use first principle models in systems large enough to validate the phenomenological models, which in turn can be used in even larger systems.

Applications of notable significance at LLNL include ddcMD, a classical molecular dynamics code that has been used to simulate systems with approximately half a billion atoms (and in the process win the 2005 Gordon Bell award), and QBox, a quantum molecular dynamics code that won the 2006 Gordon Bell award. Other success stories for Blue Gene in science include: (1) Astrophysical simulations in Argonne National Laboratory (ANL) using the FLASH code; (2) Global climate simulations in the National Center for Atmospheric Research (NCAR) using the HOMME code; (3) Biomolecular simulations at the T.J. Watson Research Center using the Blue Matter code; (4) Quantum chromo dynamics (QCD) at IBM T.J. Watson Research Center and LLNL, San Diego Supercomputing Center, Juelich Research Center, Massachusetts Institute of Technology, Boston University, University of Edinburgh, and KEK (Japan) using a variety of codes. One of the most innovative uses of Blue Gene is as the central processor for the large scale LOFAR radio telescope in the Netherlands.

BIBLIOGRAPHIC NOTES

Additional information about the system architecture of Blue Gene can be found in [6,8,10,11,15]. For details on the Blue Gene system software, the reader is referred to [910,10]. Finally, examples of the impact of Blue Gene on science can be found in [1,2,3,4,5,7,12,13,14].

RELATED ENTRIES

TOP500; LINPACK; PowerPC processor; MPI

BIBLIOGRAPHY

1. G. Almasi, G. Bhanot, A. Gara, M. Gupta, J. Sexton, B. Walkup, V. V. Bulatov, A. W. Cook, B. R. de Supinski, J. A. Greenough, F. Gygi, A. Kubota, S. Louis, F. H. Streitz, R. Yates, C. Archer, J. Moreira, C. Rendleman. Scaling Physics and Material Science Applications on a Massively Parallel Blue Gene/L System. Proceedings of the 19th ACM International Conference on Supercomputing. June 20-22, 2005. Cambridge, MA.

2. G. Almasi, S. Chatterjee, A. Gara, J. Gunnels, M. Gupta, A. Henning, J. E. Moreira, and B. Walkup. Unlocking the Performance of the BlueGene/L Supercomputer, In Proc IEEE/ACM SC04, Pittsburgh, PA (November 2004).

3. V. Bulatov, W. Cai, J. Fier, M. Hiratani, G. Hommes, T. Pierce, M. Tang, M. Rhee, R. K. Yates, and T. Arsenlis, Scalable Line Dynamics in ParaDiS, In Proc IEEE/ACM SC04, Pittsburgh, PA (November 2004).

4. B. G. Fitch, A. Rayshubskiy, M. Eleftheriou, T. J. Christopher Ward, M. E. Giampapa, M. C. Pitman, and Robert S. Germain. Blue Matter: Approaching the Limits of Concurrency for Classical Molecular Dynamics, In Proc IEEE/ACM SC06, Tampa, FL (November 2006).

5. B. Fryxell, K. Olson,P. Ricker,F. X.Timmes,M.Zingale,D.Q.Lamb,P. MacNeice, R.Rosner, and H. Tufo. FLASH: An Adaptive Mesh Hydrodynamics Code for Modeling Astrophysical Thermonuclear Flashes, Astrophysical J. Supplement, 131:273 (2000).

6. A. Gara, M. A. Blumrich, D. Chen, G. L. -T. Chiu, P. Coteus, M. E. Giampapa, R. A. Haring, P. Heidelberger, D. Hoenicke, G. V. Kopcsay, T. A. Liebsch, M. Ohmacht, B. D. Steinmacher-Burow, T. Takken, and P. Vranas. Overview of the Blue Gene/L System Architecture, IBM J. Res. Dev., 49(2/3):195–212 (March/May 2005).

7. F. Gygi, E. W. Draeger, B. R. de Supinski, R. K. Yates, F. Franchetti, S. Kral, J. Lorenz, C. W. Ueberhuber, J. A. Gunnels, and J. C. Sexton. Large-scale First-Principles Molecular Dynamics Simulations on the BlueGene/L Platform using the Qbox code, IEEE/ACM SC05, Seattle, WA (November 2005).

8. IBM Blue Gene Team. Overview of the IBM Blue Gene/P project. IBM Journal of Research and Development. Vol. 52. No. 1/2. January/March 2008. pp 199-220.

9. J. E. Moreira, G. Almasi, C. Archer, R. Bellofatto, P. Bergner, J. R. Brunheroto, M. Brutman, J. G. Casta˜nos, P. G. Crumley, M. Gupta, T. Inglett, D. Lieber, D. Limpert, P. McCarthy, M. Megerian, M. Mendell, M. Mundy, D. Reed, R. K. Sahoo, A. Sanomiya, R. Shok, B. Smith, and G. G. Stewart, Blue Gene/L Programming and Operating environment. IBM J. Res. Dev., 49(2/3): (March/May 2005).

10. J. E. Moreira *et al*. The Blue Gene/L Supercomputer: A Hardware and Software Story. International Journal of Parallel Programming. Vol. 35, No. 3, June 2007. pp 181-206.

11. V. Salapura, R. Bickford, M. Blumrich, A. Bright, D. Chen, P. Coteus, A. Gara, M. Giampapa, M. Gschwind, M. Gupta, S. Hall, R. A. Haring, P. Heidelberger, D. Hoenicke, G. V. Kopcsay, M. Ohmacht, R. A. Rand, T. Takken, and P. Vranas. Power and Performance Optimization at the System Level, In Proc ACM Computing Frontiers 2005, Ischia, Italy (May 2005).

12. K. van der Schaaf. Blue Gene in the Heart of a Wide Area Sensor Network, In Proc QCDOC and Blue Gene: Next Generation of HPC Architecture Workshop. Edinburgh, UK (October 2005).

13. F. H. Streitz, J. N. Glosli, M. V. Patel, B. Chan, R. K. Yates, B. R. de Supinski, J. Sexton, J. A. Gunnels. 100+ TFlop solidification simulations on BlueGene/L, In Proc IEEE/ACM SC05, Seattle, WA (November 2005).

14. P. Vranas, G. Bhanot, M. Blumrich, D. Chen, A. Gara, P. Heidelberger, V. Salapura, and J. Sexton. The BlueGene/L Supercomputer and Quantum ChromoDynamics. 2006 Gordon Bell Prize. In Proc IEEE/ACM SC06, Tampa, FL (November 2006).

15. C. D. Wait. IBM PowerPC 440 FPU with complex-arithmetic extensions. J. Res. Dev., 49(2/3): (March/May 2005). pp 249-254.