

System Management in the BlueGene/L Supercomputer

G. Almasi, L. Bachega, R. Bellofatto, J. Brunheroto, C. Caşcaval, J. Castaños, P. Crumley, C. Erway,
J. Gagliano, D. Lieber, P. Mindlin, J.E. Moreira, R.K. Sahoo, A. Sanomiya, E. Schenfeld, R. Swetz

IBM T.J. Watson Research Center
Yorktown Heights, NY 10598-0218

{almasig,lrbacheg,ralphbel,brunhe,cascaval,castanos,pgc,cerway,
jgaglia,lieber,pamindli,jmoreira,rsahoo,sanomiya,eugen,swetz}@us.ibm.com

M. Bae, G. Laib, K. Ranganathan

IBM Unix Development Lab

Poughkeepsie, NY

{myungbae,laib,keshav}@us.ibm.com

Y. Aridor, T. Domany, Y. Gal, O. Goldshmidt, E. Shmueli

IBM Haifa Research Lab

Haifa, Israel

{yariv,tamar,yoavg,olegg,edi}@il.ibm.com

Abstract

The BlueGene/L supercomputer will use system-on-a-chip integration and a highly scalable cellular architecture to deliver 360 Teraflops of peak computing power. With 65,536 compute nodes, BlueGene/L represents a new level of scalability for parallel systems. As such, it is natural for many scalability challenges to arise. In this paper, we discuss challenges in the area of system management and control, including machine booting, software installation, user account management, system monitoring, and job execution. We address the issue of scalability by organizing the system hierarchically. The 65,536 compute nodes are organized in 1,024 clusters of 64 compute nodes each, called processing sets. Each processing set is under control of a 65th node, called an I/O node. The 1,024 processing sets can then be managed to a great extent as a regular Linux cluster, of which there are several successful examples. Regular cluster management is complemented by BlueGene/L specific services, performed by a service node over a separate control network. Our software development and experiments have been conducted so far in architecturally accurate simulators of BlueGene/L, and we are gearing up to test real prototypes in 2003.

1 Introduction

The BlueGene/L supercomputer [1] is a new parallel system being developed by IBM in collaboration with Lawrence Livermore National Laboratory. Through the use of system-on-a-chip integration [3], coupled with a highly scalable cellular architecture, BlueGene/L will deliver 360 Teraflops (360×10^{12} floating-point operations per second) of peak computing power. This computational capability is achieved through 65,536 *compute nodes* interconnected in a $64 \times 32 \times 32$ three-dimensional torus topology.

BlueGene/L represents a new level of scalability for parallel systems. Whereas existing large scale systems range in size from hundreds (ASCI White¹, Earth Simulator²) to a few thousands (Cplant³, ASCI Red⁴) of compute nodes, BlueGene/L makes a jump of almost two orders of magnitude. With such a jump, it is natural for many scalability challenges to arise. In this paper, we are particularly interested in the challenge of system management and control. We address the issues of machine booting, software installation, user account management, system monitoring, and job execution.

¹<http://www.llnl.gov/asci/platforms/white/>

²<http://www.es.jamstec.go.jp/>

³<http://www.cs.sandia.gov/cplant/>

⁴<http://www.sandia.gov/ASCI/Red/>

The conventional approach to system management in large clusters is to rely on a single centralized control workstation (CWS), which directly manages each of the compute nodes. This architecture provides a convenient single point of control for the cluster and has been used very successfully in numerous large systems. The major drawback is that the approach is inherently nonscalable, as the single CWS represents a strict bottleneck [5].

We address the problem of scalable resource management in BlueGene/L through a hierarchical approach [8]. We organize the 65,536 compute nodes into 1,024 *processing sets* (psets). A processing set is a cluster of 64 compute nodes under control of a 65th node, called an I/O node. The compute nodes are treated as controllable external entities (i.e., devices) attached to this I/O node.

From a system management perspective, the processing sets behave as monolithic entities. The 1,024 processing sets are managed from a *service node*, which plays the role of a control workstation in a conventional cluster.

This approach has the following benefits. First, cluster management is non-intrusive to the applications running on the compute nodes. Second, from a management perspective, only the I/O nodes are visible, resulting in a system which is effectively much smaller. Third, this approach supports a mode of operation in which the I/O nodes can be self managed.

A self-managed set of I/O nodes does not need a centralized control workstation and can rely on automatic fail-over mechanisms to recover from individual node failure. This self-managed node is one of the items we want to explore in the future and it is not discussed further in this paper.

Since the target time frame for completion and delivery of BlueGene/L is 2004/2005, we have yet to fully validate this approach. All our software development and experiments have, so far, been conducted on architecturally accurate simulators of BlueGene/L. Our goal in this paper is to present the principles of our approach.

The rest of this paper is organized as follows. Section 2 presents a brief description of the BlueGene/L supercomputer, including aspects of hardware and software organization. Section 3 discusses how we are implementing system management for BlueGene/L today. Section 4 discusses future activities in our work, as we continue to develop system management functionality for the 65,536-node machine. Finally, Section 5 presents our conclusions up to this point of our work.

2 The BlueGene/L supercomputer

A detailed description of the BlueGene/L supercomputer is provided in [1]. In this section, we give an overview of the hardware and software architecture of that machine, as background for our discussion on system management.

2.1 Overall organization

The high-level organization of BlueGene/L is shown in Figure 1. The BlueGene/L supercomputer is assembled from three types of nodes: (i) compute nodes (C-node), (ii) I/O nodes (IO-node), and (iii) service nodes (S-node).

The computational core of the system consists of 65,536 compute nodes and 1,024 I/O nodes. (This ratio of 64:1 is our current design point and can be easily changed by adding or removing I/O cards.) All compute nodes are interconnected in a three-dimensional torus topology. (For simplicity, we only show a two-dimensional interconnect in Figure 1.) Each compute node has 6 links for direct connection to each of its 6 nearest neighbors in that topology (along the $+x$, $-x$, $+y$, $-y$, $+z$, and $-z$ directions). The I/O nodes are not part of this torus. All nodes in the computational core (compute and I/O) are interconnected by a tree network. (The three links are not shown in Figure 1.) This network is optimized for global operations (reduction and broadcast) and for communication between I/O and compute nodes. The I/O nodes are connected to a switched Ethernet I/O network.

The BlueGene/L computational core can be subdivided into *partitions*, which are completely (electrically) isolated and self-contained subsets of the machine. Each partition has its own set of compute and I/O nodes, interconnected by the torus and tree networks. A partition is dedicated to the execution of one job. Isolating partitions and dedicating each to a single job simplify system management.

Completing the BlueGene/L machine, there is one service node, which communicates with all the compute and I/O nodes using a separate control network, shown in the bottom of Figure 1. The BlueGene/L core is completely stateless with the power off (no nonvolatile memory, no disk) and all the basic control infrastructure is offloaded to an external service node (a conventional multiprocessor system).

The service node communicates with the computational core through the IDo chips, which act as a bridge between a trusted private Ethernet control network and a low-level JTAG network. (JTAG is a mechanism for exercising bit-level control of hardware. It allows direct manipulation of hardware state in our nodes.) The IDo chips receive JTAG commands encapsulated in UDP packets from the service node and forward them to the BlueGene/L nodes through direct serial links.

Operations implemented through the control network include the reading/writing from/to a (compute or I/O) node memory and registers, sending interrupts to a (compute or I/O) node from the service node, applying reset to a (compute or I/O) node, and turning power on and off for the compute and I/O nodes. These functions are available without any software intervention in the target nodes. The IDo chips

also control other devices in the system, such as power supplies, fans and temperature sensors. For that purpose, they also support the I²C protocol in addition to JTAG. For convenience, the service node is also connected on the Ethernet I/O network.

Complementing the BlueGene/L machine proper, the BlueGene/L complex includes a variety of other commercially available systems, that connect to the Ethernet I/O network. The most important of these support systems are shown in Figure 1. The file server contains the shared file system of the machine, with the user directories and scratch space for computations. The front end nodes provide compilation and job submission services, for users to prepare and execute their programs on BlueGene/L.

2.2 Processing sets

The compute and I/O nodes of a partition are logically organized into processing sets (psets). A pset always consists of one I/O node and a set of compute nodes associated with it. Psets are nonoverlapping, so every I/O and compute node of a partition belongs to exactly one pset. As described below, the I/O node plays an important role in managing its pset. Because they are subsets of partitions, psets are associated with just one job (and one user) at any given time.

Psets are not physical entities in the architecture. They are assembled logically from the compute and I/O nodes of a partition, by assigning compute nodes to a particular I/O node. There is a certain degree of flexibility in assigning nodes to a pset and their configuration is part of machine booting, as discussed in Section 3.1.

2.3 The compute node

The BlueGene/L compute node consists of a single compute chip and 256 MB of external DRAM chips. The compute chip contains a PowerPC processor for computations, link logic for interconnection with other compute and I/O nodes, various levels of caches, DRAM control logic, and a variety of other devices. A second PowerPC processor, targeted to serve as a communication coprocessor, is also included in the chip. Each BlueGene/L compute node implements a separate address space. The PowerPC processor(s) in the compute node can only address the node's local memory. All inter-node communication is done exclusively with messages through the torus and/or tree.

The BlueGene/L compute nodes execute a custom lightweight compute node kernel (CNK). Compute nodes are dedicated to efficiently run user applications. No system daemons or sophisticated system services reside on compute nodes. Therefore, the CNK supports the execution of a single process by a single user. This way, it does not have to handle scheduling or sharing of resources. The user process

has access to all the available local memory (TLB entries are statically allocated and the CNK does not implement paging to secondary storage) and direct access to the communication hardware.

The CNK exports a traditional POSIX interface to user processes. Fortran, C, and C++ programs can be executed on the compute node. All I/O operations of a compute node (files, sockets, streams, etc) are shipped through the tree to the corresponding I/O node, where it is executed by a dedicated service process. The CNK does not need to maintain any I/O state for its compute process. This approach simplifies the implementation of the compute node kernel.

2.4 The I/O node

From a hardware perspective, the BlueGene/L I/O node is identical to the BlueGene/L compute node. It is built with exactly the same compute chip and extra memory: 512 MB of external DRAM per I/O node. In addition, the I/O node has its Ethernet device active. (The Ethernet device is also present in the compute nodes, but it is inactive and disconnected.)

Each I/O node executes a full image of the Linux operating system. Therefore, the Linux operating system in the I/O node is also responsible for overall management of the pset. The I/O node implements services to control compute processes in the compute nodes of its pset. In particular, from an I/O node it is possible to create a compute process, send signals to that compute process, and debug that compute process through a *ptrace*-like interface. Compute process control is not performed transparently by the Linux operating system in the I/O node. Rather, there are explicit services for performing those actions. Some authors use the term *asymmetric multiprocessor* to describe the organization of a pset.

As previously mentioned, all I/O operations of a compute node (files, sockets, streams, etc) are shipped to a service process in the corresponding I/O node. The service process in the I/O node then uses traditional Linux services to implement the operation. For example, it can mount file systems through NFS and validate access permissions using information on the user. A consequence of this approach is that the root file system is the same for all compute processes of a pset.

2.5 The service node

In our design, the service node for BlueGene/L is a commercially available multiprocessor machine. The service node performs a variety of system management services, including: (i) machine booting, (ii) software installation, (iii) user account management, (iv) system monitoring, and (v) job execution. The service node can directly communi-

cate with each compute and I/O node in the system through the dedicated control network. This network is not visible to users of the system and is completely separate from the torus, the tree, and the Ethernet I/O networks.

2.6 The BlueGene/L simulator

The first hardware prototypes of BlueGene/L are targeted to become operational in mid-2003. To support our system software efforts while that hardware does not exist, we have developed an architecturally accurate simulator for the BlueGene/L machine. The simulator is accurate at the level of machine instructions. That is, it executes exactly the same binary code as the real machine will execute. It can execute approximately 2 million simulated instructions per second when running on 1.2 GHz Pentium III machine. The simulator can simulate a multi-node system (compute and I/O nodes). One major deficiency of the simulator is that it only provides limited performance information. It can analyze the memory system behavior, including hit rates for the various cache levels, and can count and classify machine instructions. It cannot compute the actual number of machine cycles that a piece of code takes to execute.

3 System management in BlueGene/L

System management in BlueGene/L is addressed by a combination of functionality in the service node and in the I/O nodes. We discuss each of these system management services in more detail below. Machine booting is performed through the service node. As we will see, software installation is closely tied with machine booting. User management, system monitoring, and job execution are performed through a combination of I/O node and service node functionality.

3.1 Machine booting

The booting of compute and I/O nodes is a process controlled by the service node through the control network. The compute and I/O nodes are completely stateless with the power off. They have no built in code (ROM) and no internal storage (disks). Therefore, what is traditionally stored in ROM in conventional machines (e.g., BIOS and boot loader) has to be pushed into those nodes through the control network.

The control infrastructure has to be up and running for BlueGene/L to operate. Therefore, the first steps in machine booting are to bring up the service node and the control network. The service node then uses a discovery mechanism to identify the I/O chips in that network. Once the I/O chips are identified, the service node can start to apply power and boot the I/O and compute nodes. When booting the nodes

in a pset, we first boot the I/O node and after that the compute nodes. That way, when each compute node comes up, it can report to its corresponding I/O node.

The boot process for a node consists of the following steps. First, a small boot loader is directly written into the (compute or I/O) node memory from the service node using the control network. The service node only has direct access to a small portion of the (compute or I/O) node memory, so the boot loader cannot be very large. This boot loader executes a communication protocol with the service node (through the control network) that can load a much larger boot image into the memory of the (compute or I/O) node. We use a single boot image for all the compute nodes and another single boot image for all the I/O nodes.

The boot image for the compute nodes contains only the code for the compute node kernel, and it is approximately 64 kB in size. The boot image for the I/O nodes contains the code for the Linux operating system (approximately 2 MB in size) and the image of a ramdisk that contains the root file system for the I/O node. Today, that ramdisk is approximately 16 MB in size. After an I/O node boots, it can mount additional file systems, from external file servers, through an NFS client. We install any utilities, system software, and configuration files (the usual `/etc`, `/bin`, `/usr`, `/proc` directories) in the ramdisk. User directories (the usual `/home` directories) are mounted from an NFS file server.

Since the exactly same boot image is used for either all compute nodes or all I/O nodes, additional information must be loaded in each (compute or I/O) node to configure it. We call this information the *personality* of a node. For an I/O node, the personality includes, among other things, its MAC and IP addresses (for the I/O Ethernet network), a unique system-wide node id, the set of compute nodes in its pset, routes to communicate with those compute nodes, and the (x, y, z) coordinates of those nodes. For a compute node, the personality includes its unique system-wide node id, its (x, y, z) coordinate, the id of its corresponding I/O node, and routes to communicate with that I/O node. With the boot images and the personality info, the I/O and compute nodes can organize themselves into psets.

3.2 Software installation

Software installation is accomplished through reconfiguration of the ramdisk for the I/O nodes. To install a new software part, we build a new ramdisk with that new software installed in the appropriate directory(ies). This approach to software management is relatively straightforward. One of its disadvantages is that it requires direct manipulation of the ramdisk. Even more serious, for any change to take effect requires reboot of an I/O node, which cannot be done while jobs are running in the corresponding pset. Further-

more, unless all I/O nodes are rebooted at the same time, it may lead to inconsistencies between I/O nodes.

3.3 User account management

The concept of a “user” in BlueGene/L is implemented only in the I/O nodes. The compute nodes do not have any information about users. User management in BlueGene/L can be accomplished with standard UNIX services. The I/O nodes can use services such as NIS, in which user configuration information is kept in the service node. To manage user accounts, we modify the appropriate password, group, shell, and other information in the NIS server (the service node). In case of new users, it is also necessary to create a new user home directory in the file server.

3.4 System monitoring

System monitoring in BlueGene/L is accomplished through a combination of I/O node and service node functionality. Each I/O node is a full Linux machine. In particular, it uses Linux services to generate a system log. This system log contains information not only from the actual I/O node, but also from the compute nodes in that pset. This aspect of system monitoring in BlueGene/L is really no different from any other Linux cluster.

Each I/O node collects the information about events generated by software and hardware components in the compute nodes. I/O nodes also act like a filtering system for all the data collected from the compute nodes. Through various severity level and error condition criteria, only critical information is passed to a centralized data repository. This data repository can be mined for statistical analysis of data, isolation of failures, and prediction of future problems. Existing tools for monitoring and prediction in Linux clusters can be adapted for BlueGene/L [2, 4, 6, 9, 10].

A complementary monitoring service for BlueGene/L is implemented by the service node through the control network. Certain device information, like fan speeds and power supply voltages, can be obtained directly by the service node (and only the service node) through the control network. In addition, the service node establishes a communication protocol through the control network with each and every compute and I/O node. The (compute or I/O) nodes use this communication protocol to report events that can be logged or acted upon by the service node. This approach establishes a completely separate monitoring service that is independent of any other infrastructure (tree and torus networks, I/O nodes, Ethernet network) in the system. Therefore, it can be used even in the case of many system-wide failures to retrieve important information.

3.5 Job execution

Job execution in BlueGene/L is also accomplished through a combination of I/O nodes and service node functionality. In order to execute parallel jobs, compute nodes (and their associated I/O nodes) have to be organized into *partitions*. A partition is a rectangular and contiguous set of compute nodes assigned to a single user. A partition is electrically isolated from all the other nodes in BlueGene/L and cannot overlap with other partitions.

When submitting a job for execution in BlueGene/L, the user specifies the desired shape of the partition to execute that job. (Each compute node executes exactly one compute process of the parallel job.) The scheduler selects an appropriate set of compute nodes to form the partition. The compute (and corresponding I/O) nodes selected by the scheduler are configured into a partition by the service node using the control network.

Once a partition is created, a job can be launched through the I/O nodes in that partition. Each I/O node is responsible for creating compute processes in the compute nodes of its pset. The I/O node loads the job executable into its compute nodes using the tree network. We note that a partition can only execute one job at a time. Once all compute processes of the job are created, they then interact directly through message passing during job execution, without any involvement from the I/O nodes. As previously mentioned, all compute process I/O operations are shipped to its corresponding I/O node for execution. Once a job is running, it is possible to send signals and perform debugging operations on its compute processes, through services in the I/O nodes.

4 Future steps

So far, we have relied exclusively on architecturally accurate simulators of BlueGene/L for all our software development and experiments. Those simulators allow us to test the correctness and functionality of our code, but provide little insight into performance behavior.

Our first 512-node prototype is targeted to be operational in mid-2003. At that point, we will start to get insight into the performance behavior of our system software. As we then proceed to build the full 65,536-node machine, we will undoubtedly learn more about our system software and we will have to tune it for scalability and improve its robustness. In particular, we will need to investigate if centralizing all control information in a single service node is compatible with our system management performance requirements.

In addition to issues of performance, scalability, and robustness, we also want to better integrate our system with standard UNIX cluster management software. That will allow us to leverage new developments in that area.

To achieve a high degree of self-healing and automatic resource management [7], we are evaluating the use of existing monitoring methods and prediction models. These forecasting mechanisms use data from the past history of the system to predict future events. Preventive and corrective actions can then be taken proactively by the system management software, thus minimizing the down time of components and the entire system.

Today, we are using conventional mechanisms like sockets for intercomponent communication in our system management software. A major piece of future work is to investigate scalable infrastructure for that communication. In particular, we want to investigate the use of commercial databases as a communication mechanism. Databases can be used not only to communicate data but also to communicate actions. Database triggers can be used to cause actions to happen in specific nodes. Databases naturally provide scalability, reliability, security, portability, logging, and robustness.

We also want to explore different ways to boot the system and alternatives for the root file system of each I/O node. We currently boot all (compute and I/O) nodes by having the service node load their full boot images. A more distributed approach would have the service node loading only the boot loader in the nodes (only the service node can do this initial load). Once the boot loader is running on the nodes, the rest of the boot can proceed more independently. The I/O nodes could load their boot images from a boot server in the Ethernet I/O network and the compute nodes could load their boot images from the I/O nodes. Loading the boot image from a boot server in an Ethernet network has been done successfully in many large Linux clusters. We will have to compare this more traditional approach with our current approach, which is more specific to BlueGene/L.

Many diskless Linux clusters adopt the approach of placing the root file system for their nodes in an external file server. This approach has many benefits when compared to our approach of placing the root file system of each I/O node in a ramdisk in local memory. First, it supports software installation without the need of a rebuild of the ramdisk and reboot. Second, it provides the flexibility of both maintaining a consistent configuration for all nodes and supporting personalization on a per node basis. (Common files can be placed in shared directories, while private files can be placed in node-specific directories.) A disadvantage of the root file system in an external file server is that it puts more pressure on that file server to service all of our 1,024 I/O nodes. The pluses and minuses of the two approaches (root file system in ramdisk or file server) will require experimentation to quantify.

5 Conclusions

With 65,536 compute nodes and 1,024 I/O nodes, the BlueGene/L machine represents a new level of scalability for parallel systems. One of the areas of challenge for a machine of this size is system management. In this paper, we have discussed our approaches to machine booting, software installation, user account management, system monitoring, and job execution in BlueGene/L.

Our main strategy for addressing system management in the scale of BlueGene/L is to organize the machine hierarchically. The 65,536 compute nodes are organized into 1,024 processing sets of 64 compute nodes each. Each processing set also contains an additional I/O node, which centralizes all I/O operations and implements management functionality for all compute processes executing in the processing set.

Strictly local management decisions in a pset are controlled by services in the I/O node of that pset. All global management services are offloaded to a separate and independent control system that executes on the service node and controls the machine through a separate control network. These services include machine booting, pset configuration, partition creating and deletion, job execution, user management, and software installation.

A common theme of our approaches for each of the system management issues, critical for success in a machine of the scale of BlueGene/L, is simplicity. We adopt a single boot image for all the I/O nodes in the machine and another single boot image for all the compute nodes. Software installation and user account management is accomplished by configuring the ramdisk in the I/O nodes boot image. System monitoring is accomplished through a combination of standard Linux functionality, in the I/O nodes, and BlueGene/L specific features, through the dedicated service node and control network. Completely separate and isolated job partitions are created to execute each job, which simplifies resource management and protection.

In 2003, we will start moving from development and experiments with simulators to tests with real hardware. Our first prototype, a 512-node machine, will allow us to validate and measure the performance of much of our system software. We will gain additional experience, and make adjustments, as we proceed to building the full 65,536-node system.

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] E. Al-Shaer, H. Abdel-Wahab, and K. Maly. Hierarchical filtering-based monitoring system for large-scale distributed

- applications. In *10th International Conference on Parallel and Distributed Computing Systems*, New Orleans, LA, October 1997.
- [3] G. Almasi et al. Cellular supercomputing with system-on-a-chip. In *IEEE International Solid-state Circuits Conference ISSCC*, 2001.
 - [4] P. Dinda. Online prediction of the running time of tasks. *Cluster Computing*, 5(3), 2002.
 - [5] G. Goldszmidt, Y. Yemini, and S. Yemini. Network management by delegation – the mad approach. In *IBM/CAS Conferece*, pages 347–359, Toronto, Canada, October 1991.
 - [6] R. Hofmann, R. Klar, B. Mohr, A. Quick, and M. Siegle. Distributed performance monitoring: Methods, tools and applications. *IEEE Transactions on Parallel and Distributed Systems*, 5(6):585–597, 1994.
 - [7] P. Horn. Autonomic computing: IBM’s prospective on the state of information technology. <http://www.research.ibm.com/autonomic>, October 2001. IBM Corporation.
 - [8] R. Sahoo, M. Bae, and J. Moreira. Semi-hierarchical approach for reliability, availability, and serviceability of cellular systems. *ACM SIGARCH Computer Architecture News*, 30(3):9–10, June 2002.
 - [9] B. Schroeder. On-line monitoring: A tutorial. *IEEE Computer*, 28(6):72–77, June 1995.
 - [10] M. Sottile and R. Minnich. Supermon: A high-speed cluster monitoring system. In *Cluster2002*, Chicago, IL, September 2002. <http://www.acl.lanl.gov/supermon/papers.html>.

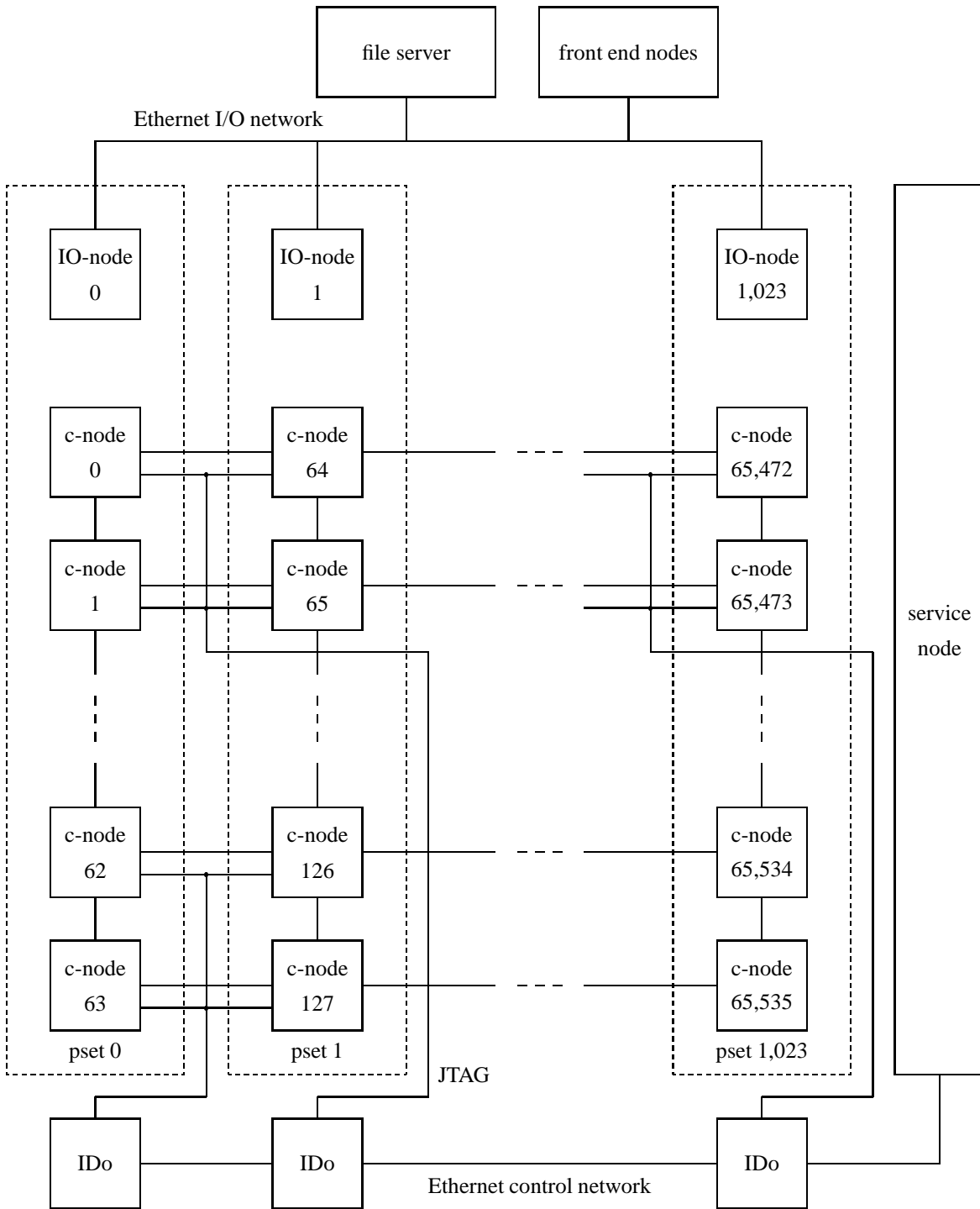


Figure 1. High-level organization of the BlueGene/L supercomputer. All 65,536 compute nodes are organized in a $64 \times 32 \times 32$ three-dimensional torus. The I/O node and 64 compute nodes of a processing set are interconnected in a three topology (links not shown.) The service node has a direct connection to all compute and I/O nodes through a separate control network.