Application Note

Porting Applications to the IBM eServer
Blue Gene/L System Solution

April 18, 2005
# Contents

List of Tables ................................................................................................................... 4

Revision History .............................................................................................................. 5

1. Abstract ................................................................................................................... 7

2. Introduction ................................................................................................................. 7

3. Programming Environment ........................................................................................ 7
   3.1 Technology ............................................................................................................. 7
   3.2 Operation Modes ..................................................................................................... 8
   3.3 Networks .................................................................................................................. 8
   3.4 Software .................................................................................................................. 8

4. Programming Requirements .................................................................................... 10
   4.1 Size ...................................................................................................................... 10
   4.2 Memory Management ............................................................................................. 10
   4.3 Programming Language ........................................................................................ 10
   4.4 Operating System ................................................................................................... 10
   4.5 Libraries ................................................................................................................. 11
   4.6 Compilers ................................................................................................................. 11
   4.7 Testing ................................................................................................................... 11

5. Performance Profiling .............................................................................................. 11

6. Optimizing Blue Gene/L Applications using the IBM XL Compiler ...................... 12

7. Double FPU Considerations .................................................................................... 13

8. MASS and MASSV Libraries .................................................................................... 13

9. Debuggers .................................................................................................................. 14
List of Tables

Table 3-1. Blue Gene/L Characteristics ................................................................. 9
## Revision History

<table>
<thead>
<tr>
<th>Date</th>
<th>Pages</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>April 18, 2005</td>
<td>—</td>
<td>Initial publication.</td>
</tr>
</tbody>
</table>
1. Abstract

This application note describes how to develop applications or modify existing applications so that they will run on the IBM eServer™ Blue Gene®/L solution. It describes the Blue Gene/L programming environment, outlines programming requirements, and discusses performance profiling. It also describes how to optimize your Blue Gene/L application using the IBM XL compiler and using optimized math library routines for double floating-point unit operation.

2. Introduction

Blue Gene/L is the inaugural member of the Blue Gene family of system solutions. It is optimized for bandwidth, scalability, and the ability to handle large amounts of data. In addition, Blue Gene/L consumes a fraction of the power and floor space required by the next fastest system. The massively parallel Blue Gene/L computer can help enable high-performance computing applications in a variety of fields. Blue Gene/L consists of one or more racks, each with 1024 compute nodes and up to 128 I/O nodes. Each node contains two IBM PowerPC® 440 processor cores.

This application note is intended to help application programmers port applications to the eServer Blue Gene/L solution.

3. Programming Environment

Blue Gene/L's processing power is encapsulated in its custom chips. Developed with IBM's leading ASIC tools and corresponding design and integration methodologies, each Blue Gene/L chip contains two embedded processors and over 4 MB of embedded DRAM. This permits the integration of all system functions, including compute processor, communications processor, three cache levels, and multiple high-speed interconnection networks with sophisticated routing, onto a single ASIC. See Table 3-1 on page 9 for a summary of Blue Gene/L characteristics.

3.1 Technology

The Blue Gene/L custom chip contains two standard 32-bit embedded PowerPC 440 cores. Each processor core is supported by a 32-KB L1 instruction cache and a 32-KB L1 data cache. Each core also has a 2-KB L2 cache, and the cores share a 4-MB L3 embedded DRAM cache. The L1 caches are not coherent. However, the L2 caches are coherent and act as prefetch buffers.

Each core has a double floating-point unit (FPU) that can perform four double-precision floating-point operations per cycle. This custom FPU consists of two conventional FPUs joined together; each has a 64-bit register file with 32 registers. The PowerPC instruction set has been extended to perform single-instruction, multiple data (SIMD) floating-point operations on the two FPUs.
3.2 Operation Modes

Depending on the nature of the application to be run on Blue Gene/L, the programmer can choose one of two modes of operation: coprocessor mode or virtual node mode. In coprocessor mode, the application runs as a single thread of execution on the main processor. The coprocessor is used as an off-load engine to help with communication. In virtual node mode, Blue Gene/L supports two independent application processes in a compute node, thus allowing both processors on a chip to be used for computation. The two processes share the L3 cache, memory, and the networks on the compute node. The two processors communicate through a non-cached region of shared memory.

3.3 Networks

The compute nodes are interconnected through five networks: a 3-dimensional torus network for point-to-point messaging between compute nodes, a global collective network for operations over the entire application, a global barrier and interrupt network, a joint test action group (JTAG) interface for machine control, and a gigabit Ethernet network for connection to external systems. The networks of interest to the application programmer are the torus and the global collective networks.

The 3-dimensional torus is the main communication network for point-to-point messages. It allows each compute node to have low-latency, high-bandwidth, bidirectional links with its six nearest neighbors. The global collective network supports fast reduction and broadcast operations. It is useful for speeding up message passing interface (MPI) constructs for collective communications. An arithmetic logic unit (ALU) in the network can combine incoming packets using bitwise and integer operations, and forward the resulting packet along the network.

3.4 Software

Blue Gene/L runs a proprietary compute node kernel (CNK) that provides a simple, flat, fixed-size, 512-MB address space, with no paging. The CNK also provides a familiar POSIX\(^1\) interface, where the GNU glibc runtime library has been ported and basic file I/O operations are supported through system calls.

The Blue Gene/L programming environment is based on familiar programming languages, libraries, job management tools, and parallel file systems. The front-end nodes of a Blue Gene system are the portals through which programmers access the compute nodes. The front-end nodes run a standard Linux\(^\circledR\) distribution; from this platform, users compile and debug programs and submit jobs. Blue Gene systems are supported by standard IBM XL Fortran, C, and C++ compilers for PowerPC that have been augmented with a backend that takes advantage of the dual floating-point unit that is unique to Blue Gene.

The same set of math libraries is provided for Blue Gene/L as for other IBM platforms. Programmers can use the IBM Engineering and Scientific Subroutine Library (ESSL), a collection of over 400 mathematical subroutines for scientific applications written in FORTRAN, C, or C++. In addition, the IBM Mathematical Acceleration Subsystem libraries, MASS and MASSV, provide elementary, trigonometric, and hyperbolic math functions in scalar and vector form.

---

1. Portable operating-system interface for UNIX\(^\circledR\)
MPI is used for high-performance message-passing on the Blue Gene/L computer. The Blue Gene/L implementation of MPI is based on MPICH2 (http://www-unix.mcs.anl.gov/mpi/mpich2/), a public domain version of the MPI2 protocol. MPICH2 provides scalability, low overhead, and a modular software approach. The MPI library for Blue Gene/L automatically makes use of multiple networks to deliver efficient, scalable performance.

Table 3-1. Blue Gene/L Characteristics

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>Two 700-MHz PowerPC 440 CPUs per node</td>
</tr>
<tr>
<td>Architecture</td>
<td>32-bit architecture</td>
</tr>
<tr>
<td>Memory</td>
<td>512 MB of double data rate (DDR) dynamic random access memory (DRAM) per node at 350 MHz; approximately 85-cycle latency</td>
</tr>
<tr>
<td>Caches</td>
<td>L1 data cache: 32 KB per processor; 32-B cache-line size; 64-way set associative; round-robin replacement</td>
</tr>
<tr>
<td></td>
<td>L2 data cache: 2 KB per processor; a prefetch buffer with 16 128-byte lines</td>
</tr>
<tr>
<td></td>
<td>L3 data cache: 4 MB embedded DRAM shared by the processors; approximately 35-cycle latency</td>
</tr>
<tr>
<td>Networks</td>
<td>3-dimensional torus—175 MBps in each direction</td>
</tr>
<tr>
<td></td>
<td>Global collective—350 MBps; 1.5 µs latency</td>
</tr>
<tr>
<td></td>
<td>Global barrier and interrupt</td>
</tr>
<tr>
<td></td>
<td>JTAG</td>
</tr>
<tr>
<td></td>
<td>Gigabit Ethernet (external)</td>
</tr>
<tr>
<td>Compute Nodes</td>
<td>1024 nodes per rack</td>
</tr>
<tr>
<td>I/O Nodes</td>
<td>Configurable from 16 to 128 nodes per rack</td>
</tr>
<tr>
<td>Operating Systems</td>
<td>Compute node—Lightweight proprietary kernel</td>
</tr>
<tr>
<td></td>
<td>I/O node—Linux</td>
</tr>
<tr>
<td></td>
<td>Front-end and service nodes—SUSE LINUX Enterprise Server 9</td>
</tr>
<tr>
<td>Compilers</td>
<td>IBM XL Fortran</td>
</tr>
<tr>
<td></td>
<td>IBM XL C/C++</td>
</tr>
<tr>
<td>Libraries</td>
<td>ESSL</td>
</tr>
<tr>
<td></td>
<td>MASS</td>
</tr>
<tr>
<td></td>
<td>MASS V</td>
</tr>
<tr>
<td></td>
<td>MPICH2 tuned for Blue Gene</td>
</tr>
<tr>
<td>System Software</td>
<td>DB2 Universal Database™</td>
</tr>
<tr>
<td></td>
<td>LoadLeveler® job scheduler</td>
</tr>
<tr>
<td></td>
<td>General Parallel File System (GPFS)</td>
</tr>
<tr>
<td>Processing Units</td>
<td>Single integer unit (FXU)</td>
</tr>
<tr>
<td></td>
<td>Single load/store unit (LSU)</td>
</tr>
<tr>
<td></td>
<td>Special double floating-point unit (DFPU)—32 primary floating-point registers, 32 secondary floating-point registers; supports both standard PowerPC and SIMD instructions</td>
</tr>
<tr>
<td>Instruction Sets</td>
<td>Standard PowerPC instructions (fadd, fmadd, fadds, fdiv)—Execute on FPU0; 5-cycle latency in the floating-point pipeline</td>
</tr>
<tr>
<td></td>
<td>SIMD instructions (fpadd, fpmadd, fpre, and so forth)—Execute on data in matched primary and secondary register pairs, generating up to two results per processor clock cycle; 5-cycle latency in the floating-point pipeline</td>
</tr>
</tbody>
</table>

1. The theoretical floating-point performance limit is one fpmadd per cycle, resulting in four floating-point operations per cycle. This amounts to $(4 \times 700 \times 10^6)$ floating-point operations per second (FLOPS) per processor core, or a peak performance of 5.6 GFLOPS per compute node.
4. Programming Requirements

This section discusses some important considerations for application programmers.

4.1 Size

In coprocessor mode, design the application to fit within the 512-MB DRAM available for each Blue Gene/L compute node. In virtual node mode, design the application to fit within the 256-MB DRAM available for each processor.

Some space must be reserved for the CNK. The CNK consumes a very small amount of space. In practice, over 500 MB is available on each compute node for the application plus the program code. Statically linked libraries also consume space. See Section 4.5 Libraries on page 11 for more information.

To determine how much static memory the program will allocate, use the Linux size command\(^1\). Information about the size command is available on most Linux systems by typing `man size` or `info size` on the Linux command line.

4.2 Memory Management

The Blue Gene/L computer implements a 32-bit memory model. It does not support a 64-bit memory model, but does support 64-bit file pointers.

The Blue Gene/L computer uses memory distributed across the nodes, and uses networks to provide high bandwidth and low-latency communication. If the memory requirement per MPI task is greater than 256 MB in virtual node mode or greater than 512 MB in coprocessor mode, then the application will not run on Blue Gene/L unless steps are taken to reduce the memory footprint. In some cases, you can reduce the memory requirement by distributing data that was replicated in the original code. In this case, additional communication might be needed. It might also be possible to reduce the memory footprint by being more careful about memory management in the application.

4.3 Programming Language

Code the application in C, C++, or Fortran with MPI for communication. MPI lets you create parallel processes and exchange information among these processes. Blue Gene/L expects a single thread of execution for each MPI process. The application should not require the OpenMP application program interface (API) or POSIX threads (pthreads).

4.4 Operating System

Since the compute node kernel is not UNIX, a number of UNIX features are not supported. Multiprocessing services are not available in the single-process compute node kernel. Therefore, the compute node kernel does not support the generation of additional processes at runtime using routines such as fork() or system().

---

\(^1\) The size command will not indicate how much memory will be dynamically allocated.
In addition, there is no support for the memory-mapping of files or of the server components of sockets (using routines such as listen, accept, and so forth). However, the socket client components (socket, connect, and so forth) are supported. Standard UNIX interprocess communication routines are not supported.

4.5 Libraries

The Blue Gene/L computer does not support dynamically linked libraries. All libraries must be statically linked. Statically linked libraries consume memory, and this memory is not available for application data. Typically, statically linked Blue Gene/L executables are 10 MB to 30 MB in size.

4.6 Compilers

Use an IBM XL Fortran or IBM XL C/C++ compiler.

4.7 Testing

IBM recommends testing an application on an IBM pSeries® system before running it on a Blue Gene/L system. Use a memory size per compute node that is compatible with the Blue Gene/L architecture (for more information, see Section 4.1 Size on page 10). This approach makes it possible to check both memory utilization and performance issues. Both pSeries and the Blue Gene/L computer use IBM XL compilers, which aids portability between the two systems.

5. Performance Profiling

For the best performance, it is good practice to obtain a performance profile for your application. IBM is porting its comprehensive performance analysis tools, the High Performance Computing Toolkit, to the Blue Gene/L computer. In the mean time, we recommend profiling on a similar system, such as pSeries. Most computational performance issues are the same on the Blue Gene/L computer as on other reduced instruction set computer (RISC) processors, so this method will usually identify the main issues.

For parallel performance, several MPI profiling tools are available, including:

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM High Performance</td>
<td>This toolkit is the foundation for all performance tools for Blue Gene and</td>
</tr>
<tr>
<td>Computing Toolkit</td>
<td>other eServer systems. The tools provide source code traceback of the</td>
</tr>
<tr>
<td></td>
<td>performance data to help the user quickly identify any bottlenecks in the</td>
</tr>
<tr>
<td></td>
<td>code. The toolkit includes low-overhead measurement of time spent in MPI</td>
</tr>
<tr>
<td></td>
<td>routines for applications written in any mixture of Fortran, C, and C++.</td>
</tr>
<tr>
<td></td>
<td>Tools include Xprofiler, MPI_tracer, MPI_Profiler, and PeekPerf. The tool</td>
</tr>
<tr>
<td></td>
<td>provides a text summary and an optional graphical display.</td>
</tr>
<tr>
<td>Paraver</td>
<td>Paraver is a GUI-based performance visualization and analysis tool that can</td>
</tr>
<tr>
<td></td>
<td>be used to analyze parallel programs. It lets you obtain detailed information</td>
</tr>
<tr>
<td></td>
<td>from raw performance traces (see <a href="http://www.cepba.upc.es/paraver/">http://www.cepba.upc.es/paraver/</a>).</td>
</tr>
<tr>
<td>MPE/jumpshot</td>
<td>MPICH2 has extensions for profiling MPI applications, and the MPE extensions</td>
</tr>
<tr>
<td></td>
<td>have been ported to Blue Gene (see <a href="http://www-unix.mcs.anl.gov/mpi/mpich/">http://www-unix.mcs.anl.gov/mpi/mpich/</a>).</td>
</tr>
</tbody>
</table>
Other performance analysis tools have been ported to Blue Gene/L including:

**KOJAK**
Kit for objective judgement and knowledge-based detection of performance bottlenecks (see [http://www.fz-juelich.de/zam/kojak/](http://www.fz-juelich.de/zam/kojak/))

**TAU**
Tuning and analysis utilities (see [http://www.cs.uoregon.edu/research/paracomp/tau/tautools/](http://www.cs.uoregon.edu/research/paracomp/tau/tautools/))

### 6. Optimizing Blue Gene/L Applications using the IBM XL Compiler

Simple compilation is the translation or transformation of the source code into an executable or shared object. An optimizing transformation gives your application better overall performance at run time. The XL compiler provides a portfolio of optimizing transformations tailored to the IBM hardware. For a complete description of optimization, see the *IBM XL User’s Guide* for the language used by your application. This section summarizes that information and provides recommendations for setting the XL compiler flags to optimize the performance of your application on Blue Gene/L.

The default optimization level for the XL compiler is none. The following optimization levels are available:

- **-O** This optimization level is a good place to start; use it with the `-qmaxmem=64000` flag.
- **-O2** This optimization level is the same as `-O`.
- **-O3** This is an aggressive optimization level. It allows reassociation, and will replace division with multiplication by the reciprocal when possible.
  - `O3-qstrict` indicates that optimization must strictly obey program semantics.
- **-O4** The `-O4` option is short for `-O3 -qhot -qipa=level=1 -qarch=auto -qtune=auto`. Therefore, with this option, add `-qarch=440d -qtune=440` to restore the proper architecture and tuning options for Blue Gene/L.
- **-O5** The `-O5` option is short for `-O3 -qhot -qipa=level=2 -qarch=auto -qtune=auto`. Therefore, with this option, add `-qarch=440d -qtune=440` to restore the proper architecture and tuning options for Blue Gene/L.

In addition, the following architecture flags are available:

- **-qarch=440** This flag generates standard PowerPC floating-point code.
- **-qarch=440d** This flag will try to generate double FPU code.
- **-qhot** This turns on the high-order transformation module. It will add vector routines, unless `-qhot=novector`.
- **-qipa** This performs interprocedure analysis. There are many suboptions such as `-qipa=level=2`.
- **-qtune=440** This is the default tuning option for Blue Gene/L.

For Blue Gene/L, IBM recommends starting with `-g -O -qarch=440 -qmaxmem=64000`. Then try `-O3 -qarch=440/440d` in selected routines. You can also try `-O5 -qarch=440d -qtune=440`. 
It is best to compile and link with `-g` to save information for debugging. Most application failures will provide a function call stack as a list of instruction addresses. You can use the GNU `addr2line` utility, which is available after you run the program under debug control, to associate a source file and line number with each instruction address.

7. Double FPU Considerations

For efficient double FPU code generation, quadword loads and stores are possible. However, they require 16-byte alignment. The IBM XL compilers accept alignment assertions. Fortran and C code samples are shown below.

Fortran

```fortran
    call alignx(16,x(1))
    call alignx(16,y(1))
!ibm* unroll(10)
    do i = 1, n
      y(i) = a*x(i) + y(i)
    end do
```

C

```c
    double * x, * y;
    #pragma disjoint (*x, *y)
    __alignx(16,x);
    __alignx(16,y);
    #pragma unroll(10)
    for (i=0; i<n; i++) y[i] = a*x[i] + y[i];
```

The easiest way to generate efficient double FPU code is to use optimized math library routines.

8. MASS and MASSV Libraries

The MASS and MASSV libraries, available at [http://www.ibm.com/software/awdtools/mass/support/](http://www.ibm.com/software/awdtools/mass/support/), consist of a set of mathematical functions for C, C++, and Fortran-language applications that are tuned for specific POWER architectures. Both scalar (libmass.a) and vector (libmassv.a) intrinsic routines are tuned for the Blue Gene/L computer.

In many situations, using these libraries has been shown to result in significant code performance improvement. Routines such as sin, cos, exp, log, and so forth from these libraries are significantly faster than the standard routines from GNU `libm.a`. For example, a `sqrt()` call costs about 106 cycles with `libm.a`, about 46 cycles for `libmass.a`, and 8 to 10 cycles per evaluation for a vector of `sqrt()` calls in `libmassv.a`. To link with `libmass.a`, include the following option on the link line:

```
-Wl,--allow-multiple-definition.
```
9. Debuggers

Blue Gene/L system software includes support for the GNU debugger (gdb). In addition, Etnus, Inc. is developing TotalView for Blue Gene/L (see http://www.etnus.com/TotalView/index.html).