The Anatomy of a Blue Gene /Q: Mira
Network Speed is a Major Strength of BG/Q

• 11 network links per node
  • Bi-directional bandwidth per-link: 4 GB/s
  • 10 links for 5D torus, 1 link for I/O
• Bisection bandwidth (32 racks): 13.1 TB/s
• HW latency
  ▪ Best: 80ns (nearest neighbor)
  ▪ Worst: 3 μs (96-rack 20 PF system, 31 hops)
• MPI latency (zero-length, nearest-neighbor): 2.2 μs
Network Design is a Major Strength of BG/Q

- Partitions provide network isolation
  - Low noise
  - You are the only user on your partition
  - I/O nodes may be shared on partitions smaller than a midplane
- Partitions are rebooted and configured for each Cobalt job
  - This does slow job startup times
  - Nothing left over from other user’s jobs - tabula rasa
- Excellent performance counters
- Well-designed APIs for topology and affinity information
# Mira multiple rack partitions ("blocks")

The number of large block sizes possible is:

<table>
<thead>
<tr>
<th># of nodes</th>
<th># of blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>49152</td>
<td>1</td>
</tr>
<tr>
<td>32768</td>
<td>3</td>
</tr>
<tr>
<td>24576</td>
<td>2</td>
</tr>
<tr>
<td>16384</td>
<td>9</td>
</tr>
<tr>
<td>12288</td>
<td>12</td>
</tr>
<tr>
<td>8192</td>
<td>6</td>
</tr>
<tr>
<td>4096</td>
<td>12</td>
</tr>
<tr>
<td>2048</td>
<td>24</td>
</tr>
<tr>
<td>1024</td>
<td>64</td>
</tr>
<tr>
<td>512</td>
<td>96</td>
</tr>
</tbody>
</table>


**partlist** will show you if a large free block is busy due to a wiring dependency.
Mira Decomposed

Multi-rack system
Mira: 48 racks, 10 PF/s
Geometry: 8x12x16x16x2

Rack
2 midplanes
1 I/O drawer (in I/O rack)
Geometry: 4x4x4x8x2

I/O drawer
8 I/O cards w/16 GB
8 PCIe Gen2 x8 slots
IB to Storage Network

Midplane
16 node boards
Electrical network backplane
The smallest full 5D torus!
Geometry: 4x4x4x4x2

Node board
32 compute cards
5D Torus Link Chips
Optical modules connect to other midplanes and I/O
Electrical connection to the midplane
Geometry: 2x2x2x2x2

Module
Single chip

Chip
16+2 cores

Compute card
One single chip module
16 GB DDR3 Memory
Heat Spreader for H2O Cooling
Electrical Connection to the network

Electrical Connection to the network...
## Partition dimensions on ALCF Blue Gene/Q systems

### Mira

<table>
<thead>
<tr>
<th>Nodes</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>1024</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>2048</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>4096</td>
<td>4/8</td>
<td>4</td>
<td>8/4</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>8192</td>
<td>4</td>
<td>4</td>
<td>16</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>12288</td>
<td>8</td>
<td>4</td>
<td>12</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>16384</td>
<td>4/8</td>
<td>8/4</td>
<td>16</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>24576</td>
<td>4</td>
<td>12</td>
<td>16</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>32768</td>
<td>8</td>
<td>8</td>
<td>16</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>49152</td>
<td>8</td>
<td>12</td>
<td>16</td>
<td>16</td>
<td>2</td>
</tr>
</tbody>
</table>

### Cetus

<table>
<thead>
<tr>
<th>Nodes</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>256</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>512</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>1024</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>2048</td>
<td>4/8</td>
<td>4/4</td>
<td>8/4</td>
<td>8/8</td>
<td>2</td>
</tr>
<tr>
<td>4096(*)</td>
<td>8</td>
<td>4</td>
<td>8</td>
<td>8</td>
<td>2</td>
</tr>
</tbody>
</table>

### Vesta

<table>
<thead>
<tr>
<th>Nodes</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>64</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>128</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>256</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>512</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>1024</td>
<td>4</td>
<td>4</td>
<td>4/8</td>
<td>8/4</td>
<td>2</td>
</tr>
<tr>
<td>2048(*)</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>8</td>
<td>2</td>
</tr>
</tbody>
</table>

(*) Partition not active.

Command: partlist

http://www.alcf.anl.gov/user-guides/machine-partitions
BG/Q 512 Node Torus Partition – A Midplane

A × B × C × D × E

4 × 4 × 4 × 4 × 2
BG/Q Compute Chip
BG/Q Compute Chip
BG/Q Compute Chip
BG/Q Compute Chip
BG/Q Network

Network

5D Torus

Process number within node
BG/Q Network: Use the Network Fully!
Mapping Ranks/Processes to Nodes

Rank to hardware mapping rules:
• Default is ABCDET
• Rightmost letter increments first
• Example: ABCDET on midplane --mode c1 <4,4,4,4,2,1>
  Rank 0 coordinates <0,0,0,0,0,0>
  Rank 1 coordinates <0,0,0,0,1,0>
  Rank 2 coordinates <0,0,0,1,0,0>
  Rank 3 coordinates <0,0,0,1,1,0>
  Rank 4 coordinates <0,0,0,2,0,0>
  Rank 5 coordinates <0,0,0,2,1,0>
  Rank 6 coordinates <0,0,0,3,0,0>
  Rank 7 coordinates <0,0,0,3,1,0>
  Rank 8 coordinates <0,0,1,0,0,0>
  ...
  Rank 511 coordinates <3,3,3,3,1,0>

Alternate mappings are possible:
• Shortcut:
  runjob --mapping TEDCBA ...
• For more control, a mapping file can be created:
  # note # is ignored by runjob
  # making it useful for comments
  0 0 0 0 0 0 # rank 0
  0 0 0 0 1 0 # rank 1
  0 0 0 1 0 0 # rank 2
  ...
  • The mapping file is then supplied to runjob:
    runjob --mapping mapfilename
  • See the Redbook for more details
Mapping Ranks/Processes to Nodes (cont’d)

Goal for Cartesian topologies:
• Preserve locality for nearest-neighbor
• Minimize extra hops in partition
• Example:
  • 2D logical topology:
    • 64 x 128 Cartesian grid
  • 5D Network topology
    • Midplane booted in mode c16 <4,4,4,4,2,16>

Two ways to implement mapping:
1. Generate map file
2. Order the ranks in a new MPI communicator

MPI_Comm_split(MPI_COMM_WORLD, color, key, new2DComm);

Order in 64 × 128

Partition: 4 4 4 4 2 16

4 4 4
4 2 16
64 × 128
Topology Access: MPIX

/* from /bgsys/drivers/ppcfloor/comm/include/mpix.h */
#include <mpix.h>

MPIX_Init_hw(MPIX_Hardware_t *hw)

int MPIX_Torus_ndims(int *numdimensions)
int MPIX_Rank2torus(int rank, int *coords)
int MPIX_Torus2rank(int *coords, int *rank)

MPIX_Hardware_t
- Physical rank irrespective of mapping
- Size of block irrespective of mapping
- Number of processes per node
- Core-thread ID of this process
- Frequency of the processor clock
- Size of the memory on the compute node
- Number of torus dimensions
- Size of each torus dimension
- Torus coordinates of this process
- Wrap-around link attribute for each torus dimension
Blue Gene/Q Communication Programming

Application

MPICH2  Global Arrays  Charm++  GASNet  High-level APIs

PAMI – Parallel Active Messaging Interface

SPI (System Programming Interface) for messaging

Network Hardware (MU)
MPI on BG/Q
Default MPI:
• Based on MPICH MPI-2.2
  • Forgoes incompatible features (those needing fork, e.g.)
• Slightly dated at this point
• Fully Open Source!
• Accessed through wrappers
• Hardware accelerated collectives that scale
  – you won’t even notice the lack of non-blocking collectives
• We officially support this one

MPICH 3
• Based on upstream MPICH 3.x
• Accessed through wrappers exposed in SoftEnv
• We in no way officially support these builds
  • There is unofficial support
• Lacks full hardware acceleration
MPI on BG/Q: Wrapper Anatomy

+mpiwrapper-xl

IBM XL compilers

Fine-grain locking

Overlapping access to MPI by multiple threads
MPI on BG/Q: Wrapper Anatomy

+mpiwrapper-xl.legacy

Coarse-grain locking

Mutual exclusion between threads at MPI function level
MPI on BG/Q: XL Wrappers

+mpiwrapper-xl.ndebug

+mpiwrapper-xl.legacy.ndebug

No error checking or asserts
MPI on BG/Q: GNU and Clang Wrappers

- +mpiwrapper-gcc
- +mpiwrapper-gcc.legacy
- +mpiwrapper-gcc.ndebug
- +mpiwrapper-gcc.legacy.ndebug

- +mpiwrapper-bgclang
- +mpiwrapper-bgclang.legacy
- +mpiwrapper-bgclang-nightly
- +mpiwrapper-bgclang.legacy-nightly

GNU compilers

Clang/LLVM compilers
MPI on BG/Q

OSU MPI Latency Test v3.6 (intranode)

- gcc.legacy
- xl
- xl.ndebug
- xl.legacy
- xl.legacy.ndebug

_latency (us)

_size (bytes)_

1 2 4 8 16

1 8 64 512 4096 32768
MPI on BG/Q

OSU MPI Latency Test v3.6 (internode)

- gcc.legacy
- xl
- xl.ndebug
- xl.legacy
- xl.legacy.ndebug

Latency (us) vs. size (bytes)

1  2  4  8  16  32
1  2  4  8  16  32

size (bytes)

1  8  64  512  4096  32768
MPI on BG/Q
MPI on BG/Q

OSU MPI Multi-threaded Latency Test v3.6 (intranode)
PAMI: Where Does It Fit In?

- MPICH2
- Global Arrays
- Charm++
- GASNet

- PAMI – Parallel Active Messaging Interface
- SPI (System Programming Interface) for messaging
- Network Hardware (MU)
PAMI: Environment Variables

**PAMID_STATISTICS** Turns on statistics printing for the message layer such as the maximum receive queue depth. *Disabled by default.*

**PAMID_VERBOSE** When set to 1, it’ll add about 20 lines to the top of your output file, but contains extremely valuable information such as the PAMID_, PAMI_, MUSPI_, COMMAGENT_, and BG_ environment variables and other variables that the user specifies.

Setting this to 2 or 3 provides substantial output that is useful for debugging performance, particularly of collectives. *Disabled by default.*
Simple Tuning with PAMI

• PAMI is to BG/Q as IBVERBs is to a Beowulf or uGNI is to a Cray
• point-to-point communication routing can either be:
  • Deterministic:
    ▪ packets always take the same route
    ▪ lower latency
    ▪ hotspots are possible
  • Adaptive:
    ▪ packets can take several different routes determined at runtime based on load
    ▪ keeps things balanced
    ▪ adds latency
## Simple Tuning with PAMI
Routing depends on protocol – defaults:

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Packet Size</th>
<th>Routing</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Immediate</td>
<td>&lt;= 112 bytes</td>
<td>Deterministic</td>
<td>Cut off set by PAMID_SHORT variable</td>
</tr>
<tr>
<td>Short</td>
<td>512 bytes (496 usable)</td>
<td>Deterministic</td>
<td>Single packet messages only</td>
</tr>
<tr>
<td>Eager</td>
<td>Medium sized &lt; 2048 bytes</td>
<td>Deterministic</td>
<td>Sends without negotiating that the receiver is ready which can eat memory.</td>
</tr>
<tr>
<td>Rendezvous</td>
<td>Large messages &gt;= 2048 bytes. Provides highest bandwidth.</td>
<td>Adaptive</td>
<td>Handshaking required. Receiver negotiates a DMA transfer from the sender.</td>
</tr>
</tbody>
</table>
Simple Tuning with PAMI

• One can choose to use rendezvous protocol with the PAMID_RZV variable
• Profile for your communication patterns, then:
  • Lower if:
    ▪ There’s high overlap of communication and computation
    ▪ Eager is creating congestion
    ▪ Latency isn’t a huge factor for medium size messages
    ▪ You run out of memory due to MPI_*Sends
  • Raise if:
    ▪ Most communication is nearest-neighbor
    ▪ Latency is important for medium-sized messages
  • Drop to 0 if:
    ▪ Eager messages are causing full-system jobs to run out of memory
Writing Applications with PAMI and SPI: Just Say No

No vendor is carrying forward PAMI – we encourage users to explore alternatives.
Final Thoughts

• Before doing anything to optimize your code – profile it
• If you don’t use MPI_THREAD_MULTIPLE use the .legacy wrappers
• If you do use MPI_THREAD_MULTIPLE and the IBM compilers, use the wrappers ending with _r to ensure thread safety
• Think about your topology
• Use the collectives
  • The Blue Gene /Q has great collectives
  • Vendors on other platforms usually have pretty good collectives
  • Portability for years trumps a couple of years of performance
References

- Blue Gene/Q Application Development Redbook
- /bgsys/drivers/ppcfloor/comm/sys/include/pami.h
- PAMI Programming Guide
- IPDS 2012 Talk (Sameer Kumar)
- OpenSHMEM 2013 talk (Alan Benner)
- Mysteries of the Deep (J. Hammond)
- Jeff Hammond’s HPCInfo github site
Thanks!