Performance Optimization II: Multi-Node Communication/Network Topology Libraries on Blue Gene /Q

William Scullin
ALCF Catalyst Team

ALCF Computational Performance Workshop
May 15-17, 2018

The Anatomy of a Blue Gene /Q: Mira

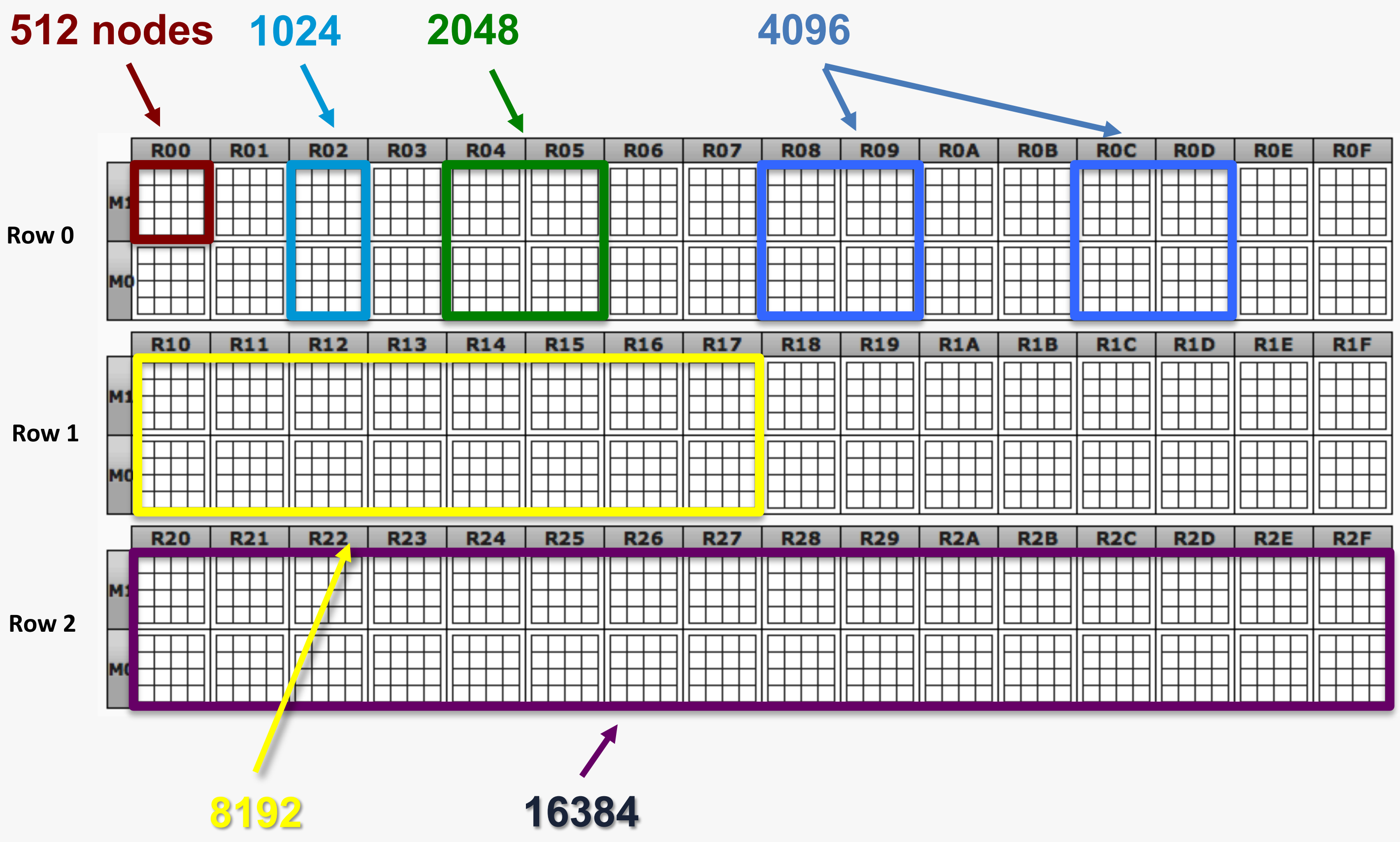
Network Speed is a Major Strength of BG/Q

- 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")



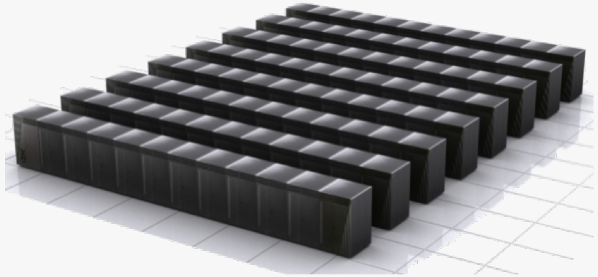
The number of large block sizes possible is:

# of nodes	# of blocks
49152	1
32768	3
24576	2
16384	9
12288	12
8192	6
4096	12
2048	24
1024	64
512	96

http://status.alcf.anl.gov/mira/activity (beta, a.k.a. The Gronkulator)

partlist will show you if a large free block is busy due to a wiring dependency

Mira Decomposed



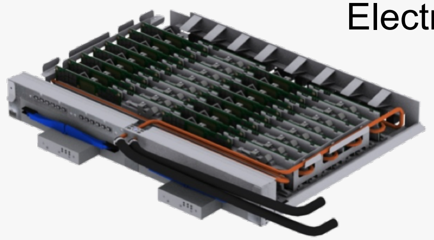
Multi-rack system
Mira: 48 racks, 10 PF/s
Geometry: 8x12x16x16x2

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

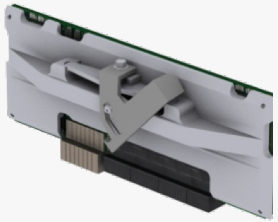
I/O drawer
8 I/O cards w/16 GB
8 PCIe Gen2 x8 slots
IB to Storage Network

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

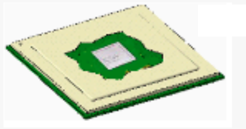
Node board
32 compute cards
5D Torus Link Chips
Optical modules connect to other midplanes and I/O
Electrical connection to the midplane
Geometry: 2x2x2x2x2



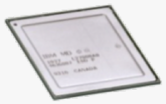
Compute card
One single chip module
16 GB DDR3 Memory
Heat Spreader for H_2O Cooling
Electrical Connection to the network



Module
Single chip



Chip
16+2 cores



Partition dimensions on ALCF Blue Gene/Q systems

Mira

Nodes	A	B	C	D	E
512	4	4	4	4	2
1024	4	4	4	8	2
2048	4	4	4	16	2
4096	4/8	4	8/4	16	2
8192	4	4	16	16	2
12288	8	4	12	16	2
16384	4/8	8/4	16	16	2
24576	4	12	16	16	2
32768	8	8	16	16	2
49152	8	12	16	16	2

Command: partlist

http://www.alcf.anl.gov/user-guides/machine-partitions

Cetus

Nodes	A	B	C	D	E
128	2	2	4	4	2
256	4	2	4	4	2
512	4	4	4	4	2
1024	4	4	4	8	2
2048	4/8/8	4/4/4	8/4/8	8/8/4	2
4096(*)	8	4	8	8	2

Vesta

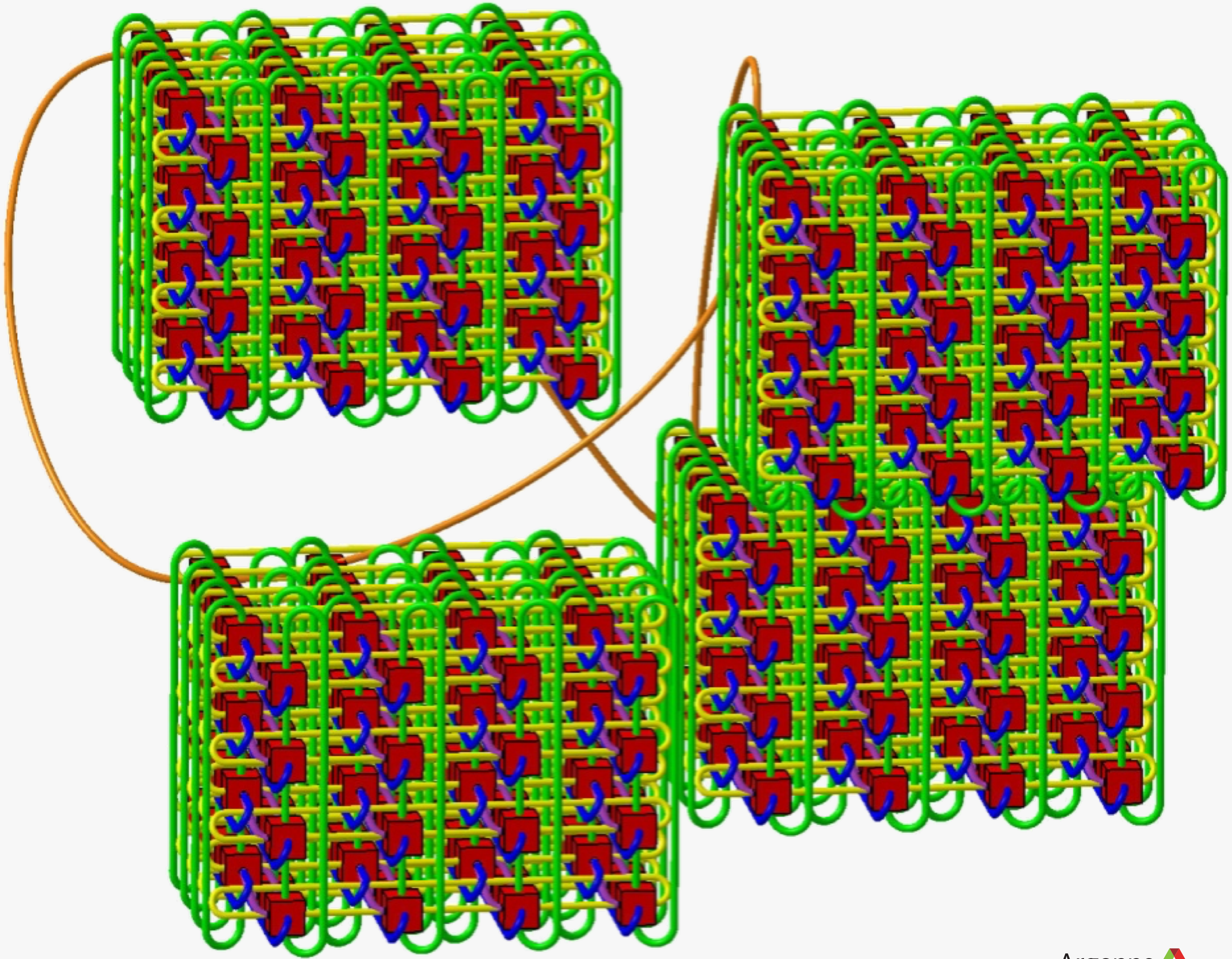
Nodes	A	B	C	D	E
32	2	2	2	2	2
64	2	2	4	2	2
128	2	2	4	4	2
256	4	2	4	4	2
512	4	4	4	4	2
1024	4	4	4/8	8/4	2
2048(*)	4	4	8	8	2

(*) Partition not active.

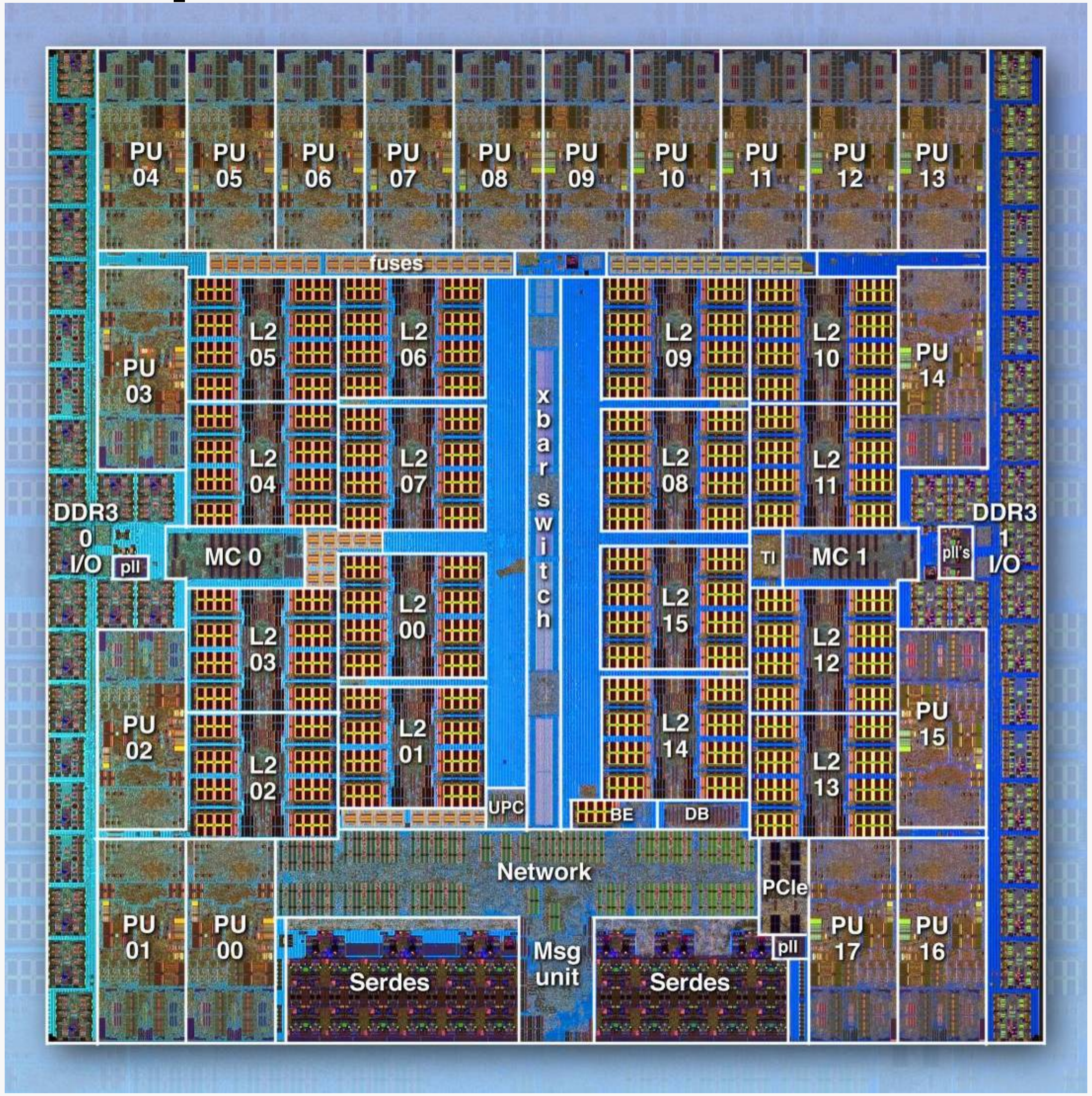
BG/Q 512 Node Torus Partition – A Midplane

A × B × C × D × E

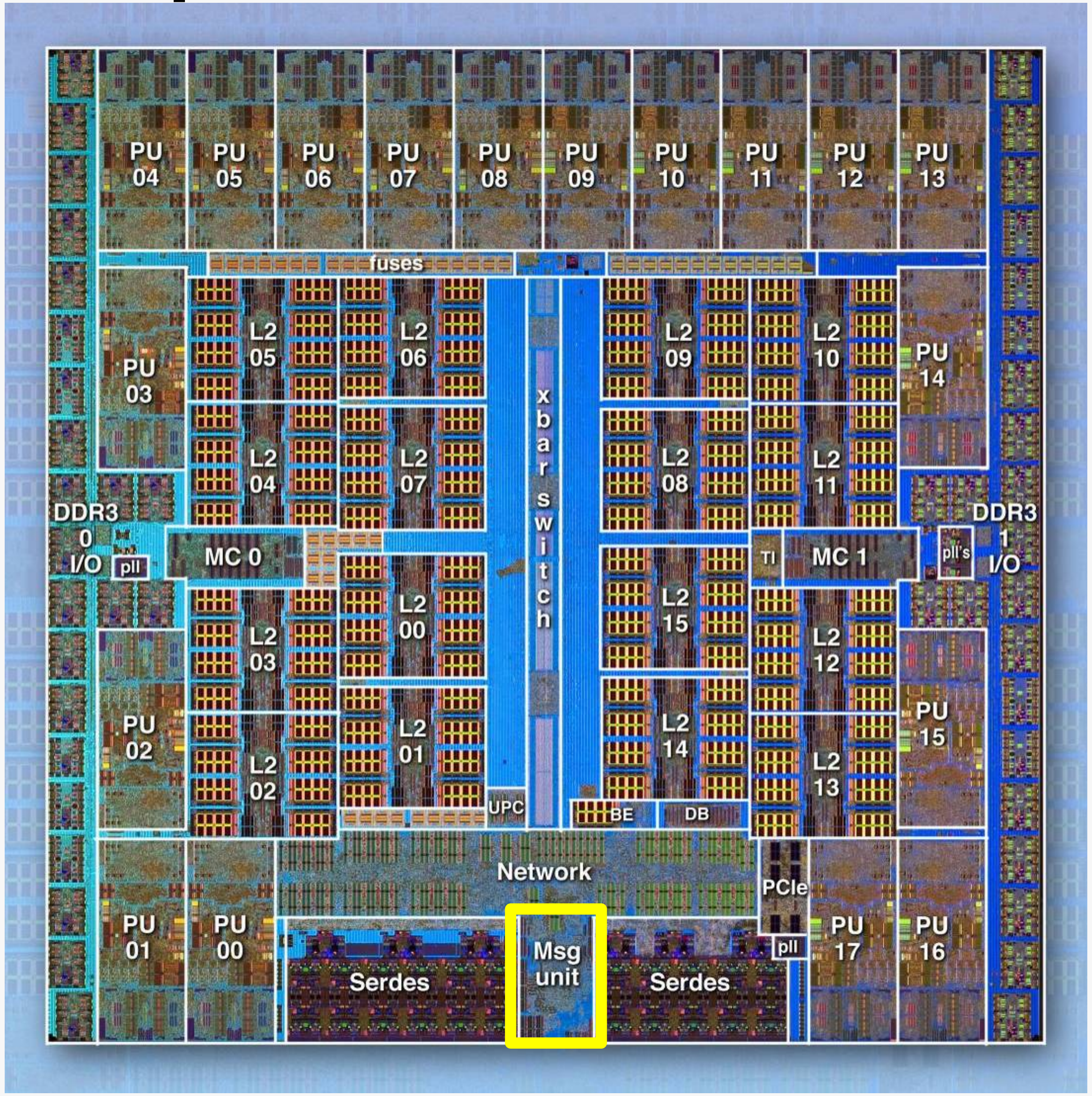
4 × 4 × 4 × 4 × 2



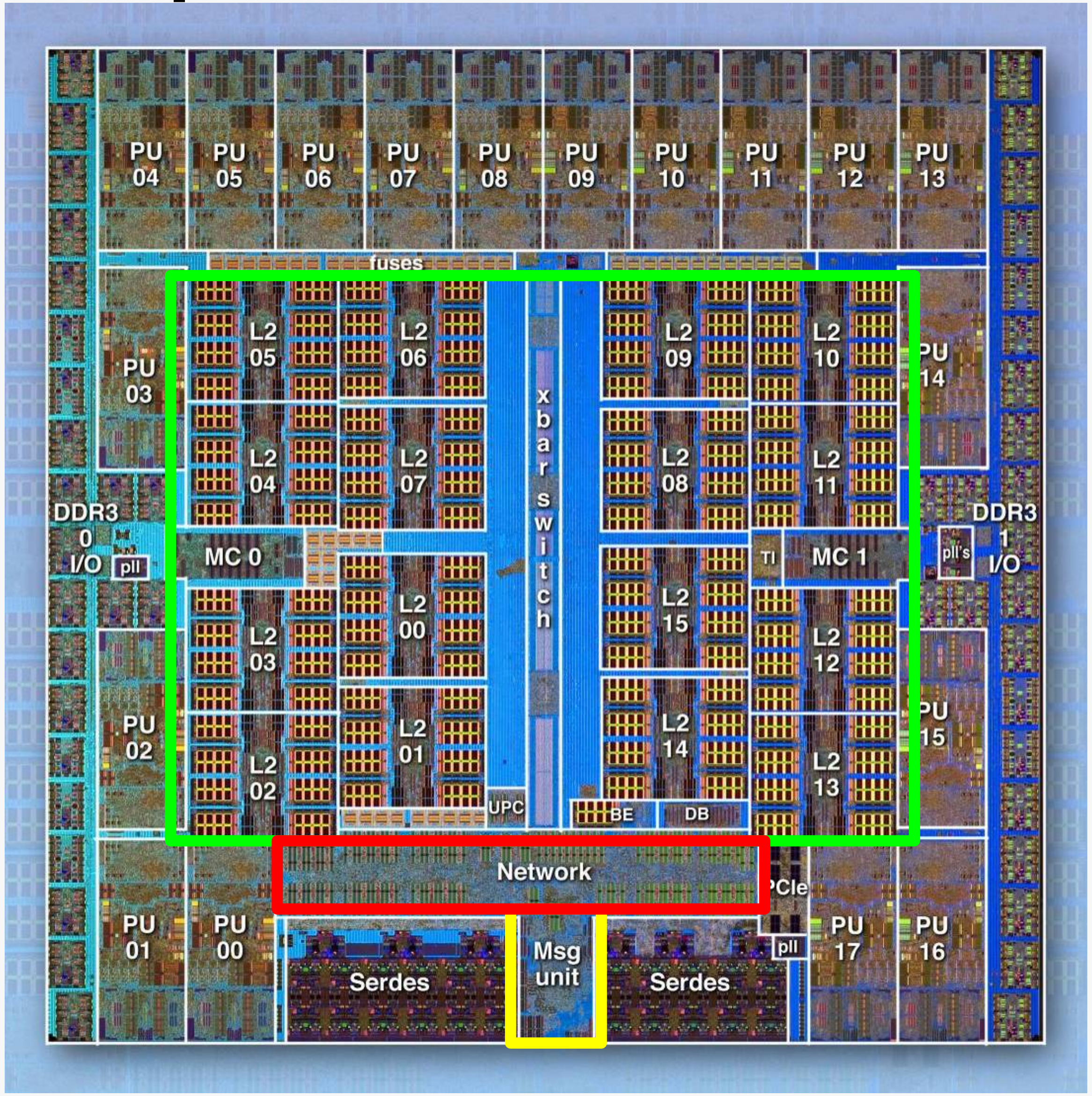
BG/Q Compute Chip



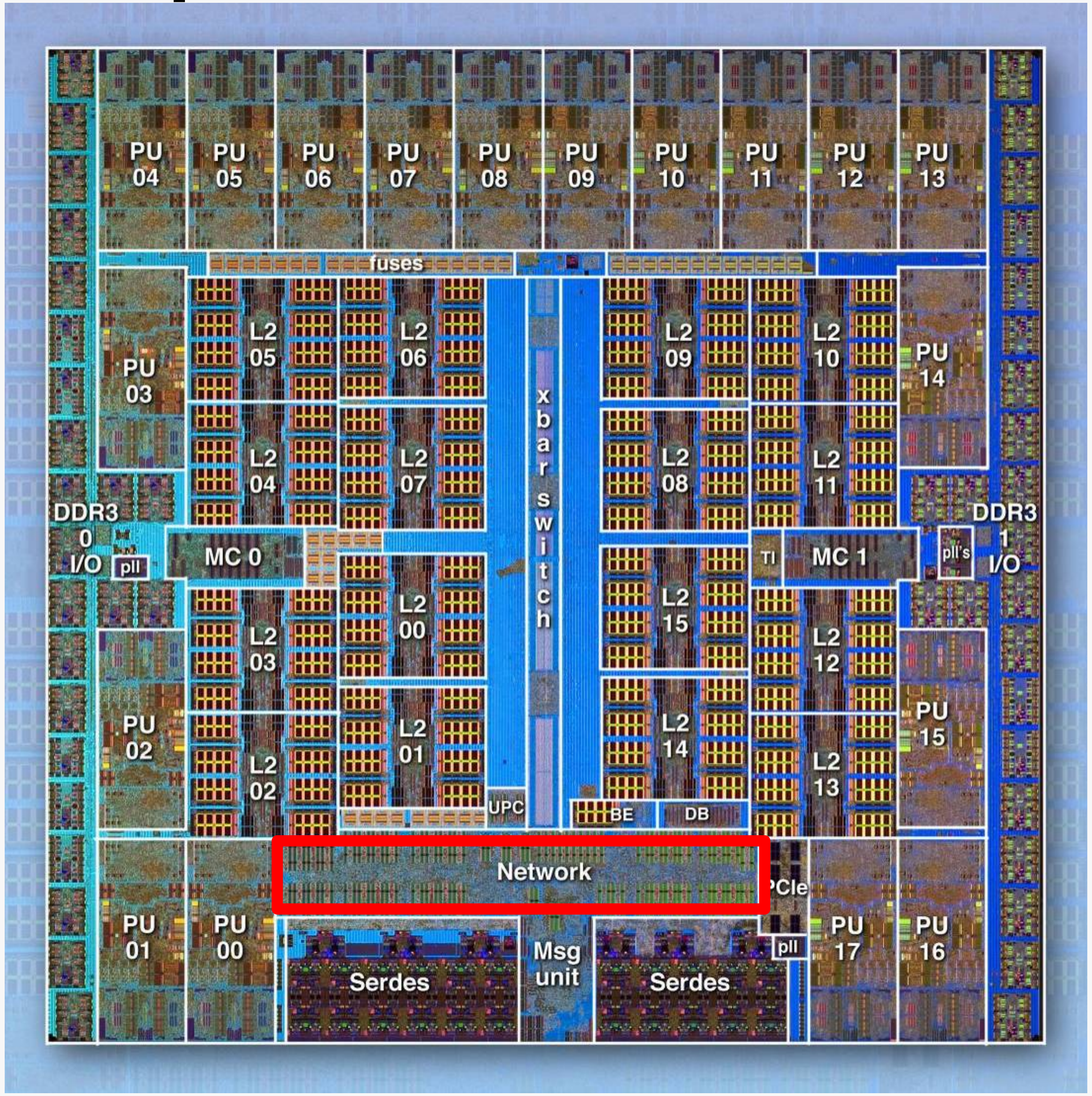
BG/Q Compute Chip



BG/Q Compute Chip

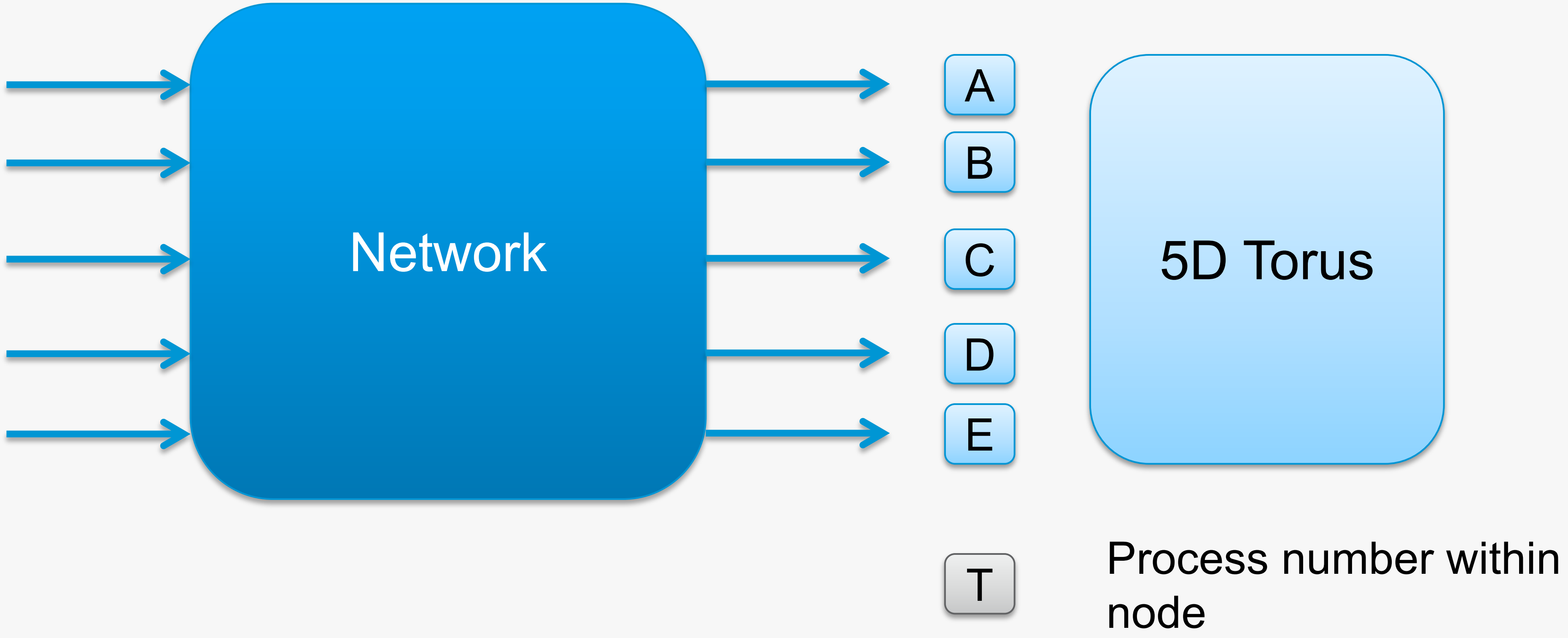


BG/Q Compute Chip

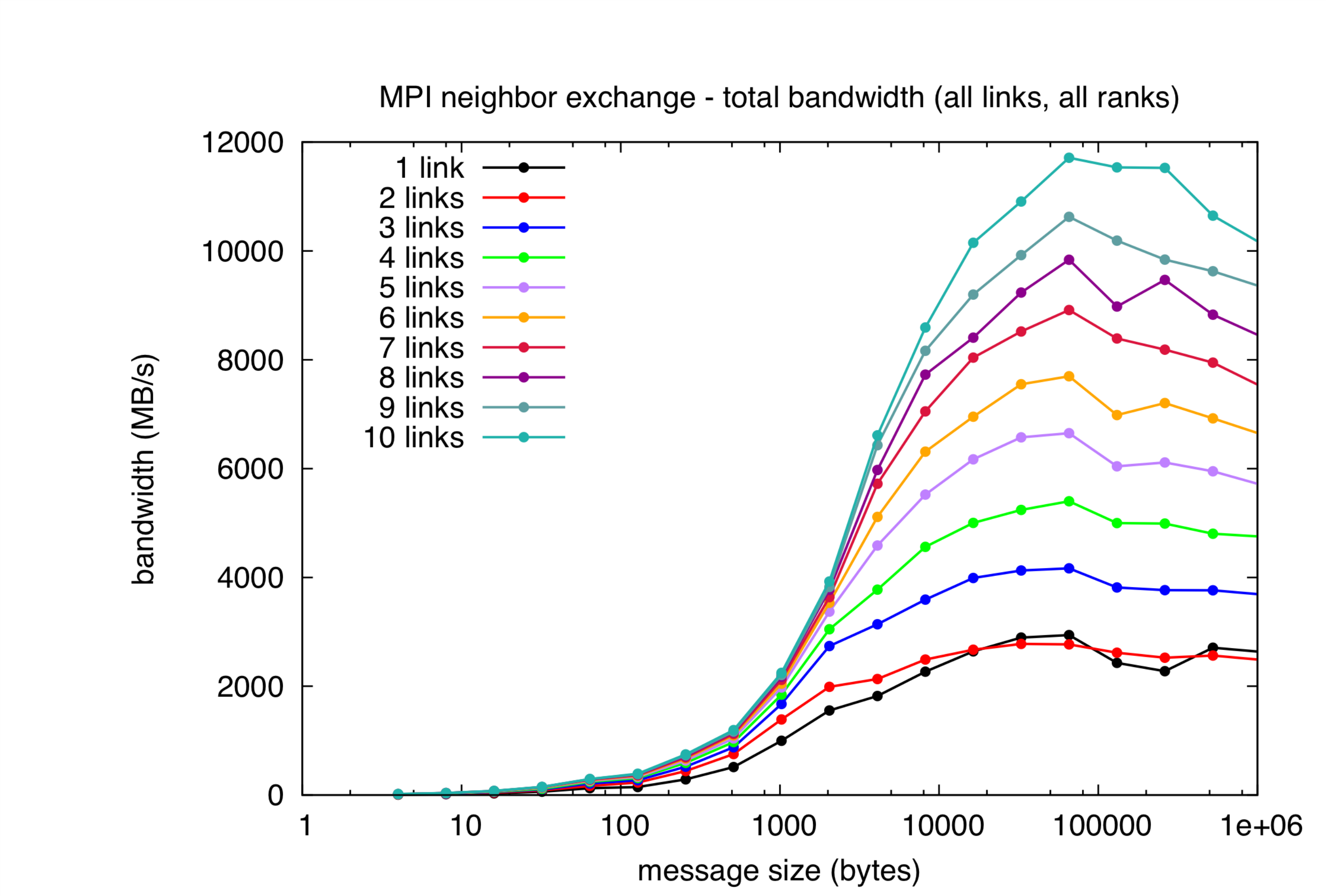


Argonne NATIONAL LABORATORY

BG/Q Network



BG/Q Network: Use the Network Fully!



Argonne NATIONAL LABORATORY

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>

Partition:4 4 4 4 2 16

4 4 4

4 2 16

64 × 128

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

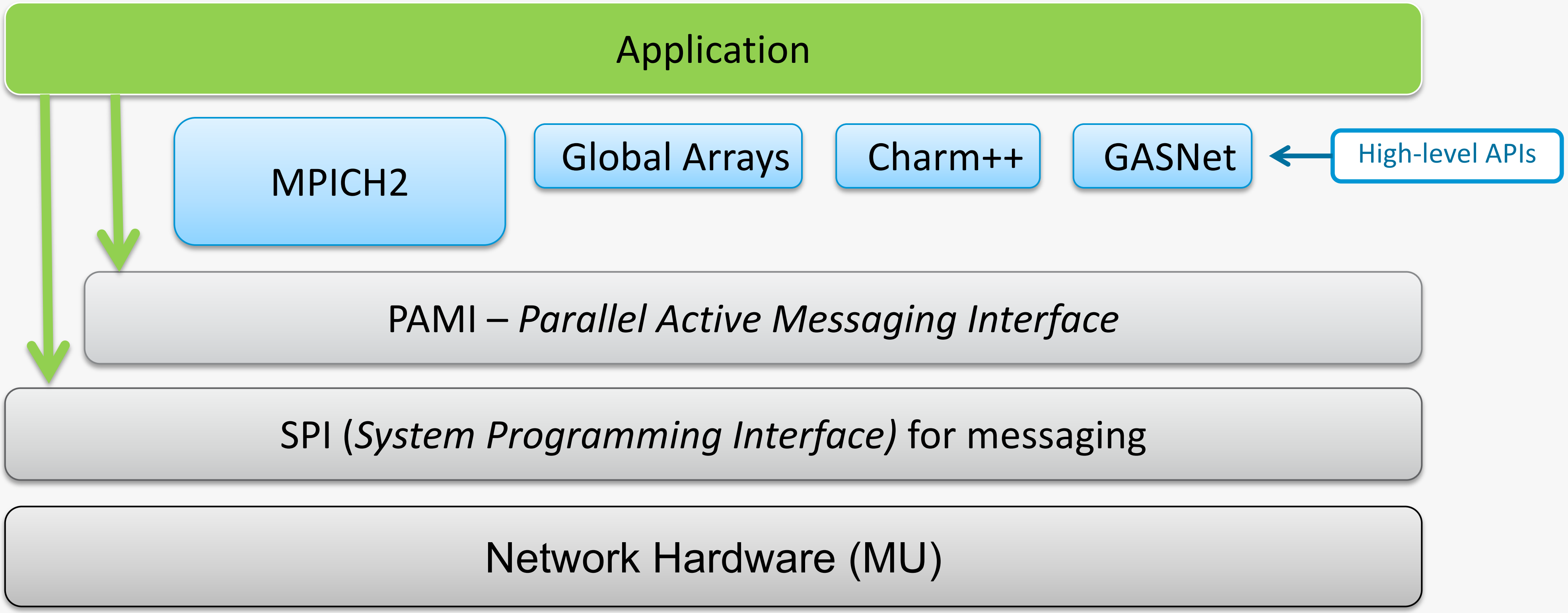
Topology Access: MPIX

```
/* 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



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

Fine-grain locking

IBM XL compilers

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

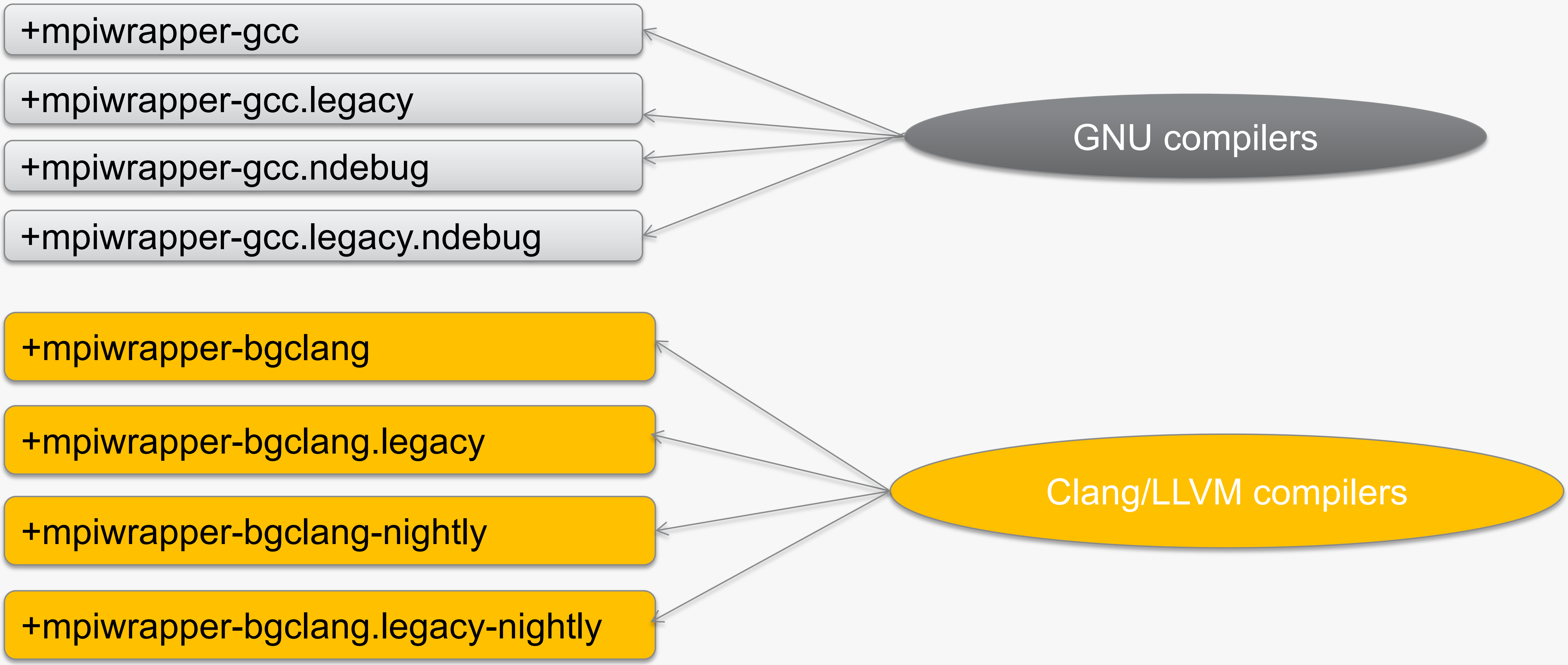
MPI on BG/Q: XL Wrappers

+mpiwrapper-xl.**ndebug**

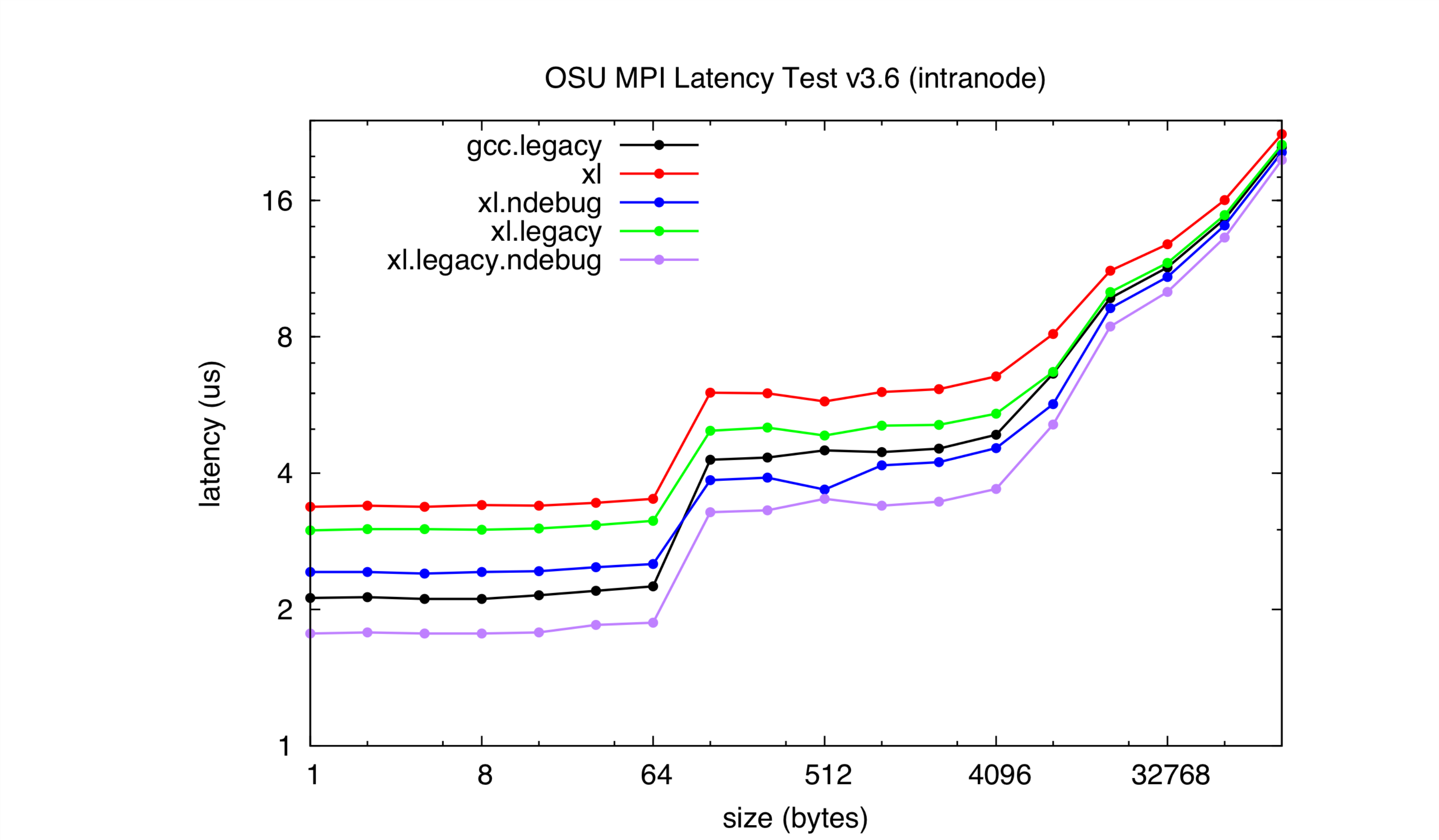
+mpiwrapper-xl.**legacy.ndebug**

No error checking or asserts

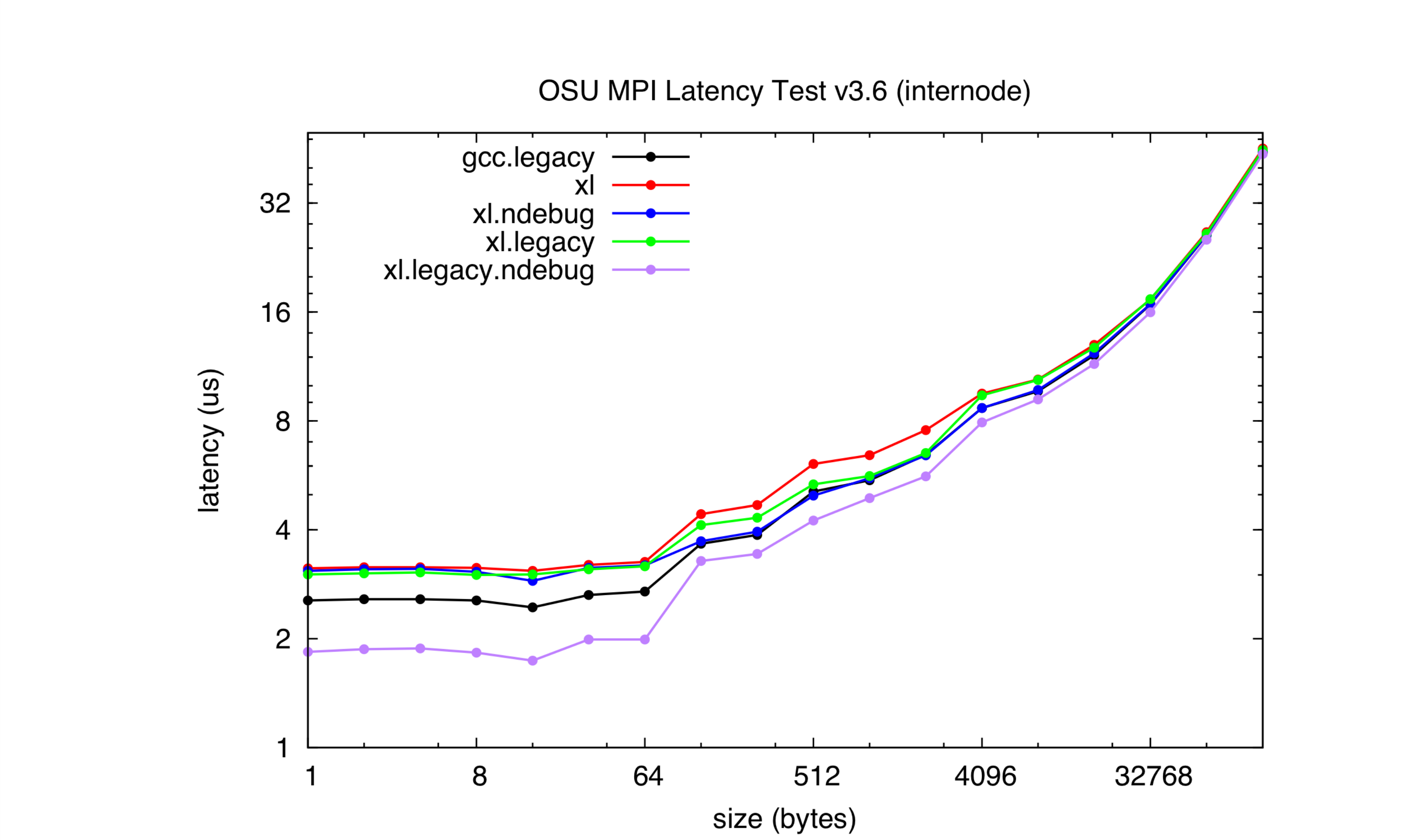
MPI on BG/Q: GNU and Clang Wrappers



MPI on BG/Q

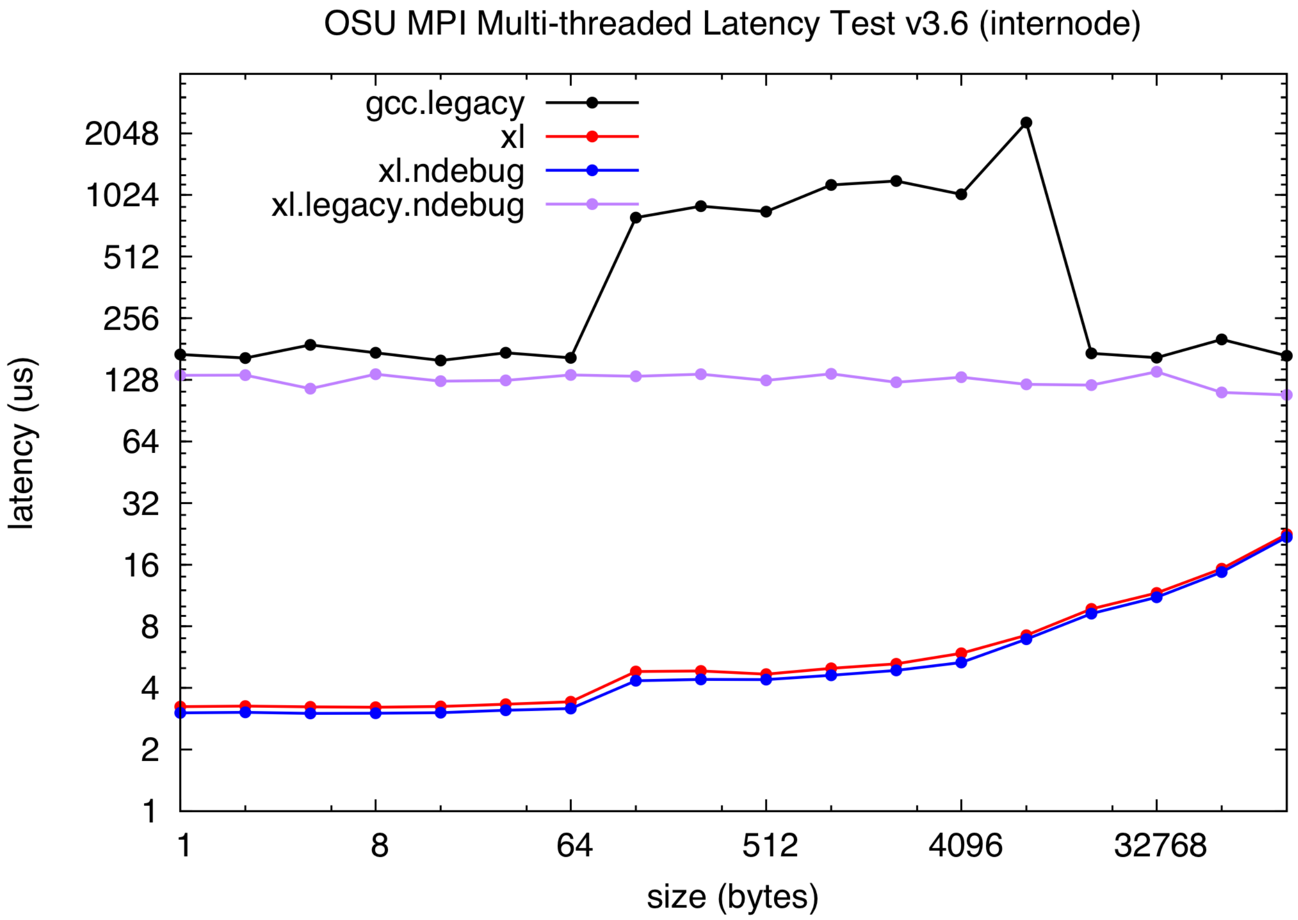

OSU MPI Latency Test v3.6 (intranode)
gcc.legacy
xl
xl.ndebug
xl.legacy
xl.legacy.ndebug
latency (us)
16
8
4
2
1
1
8
64
512
4096
32768
size (bytes)

MPI on BG/Q



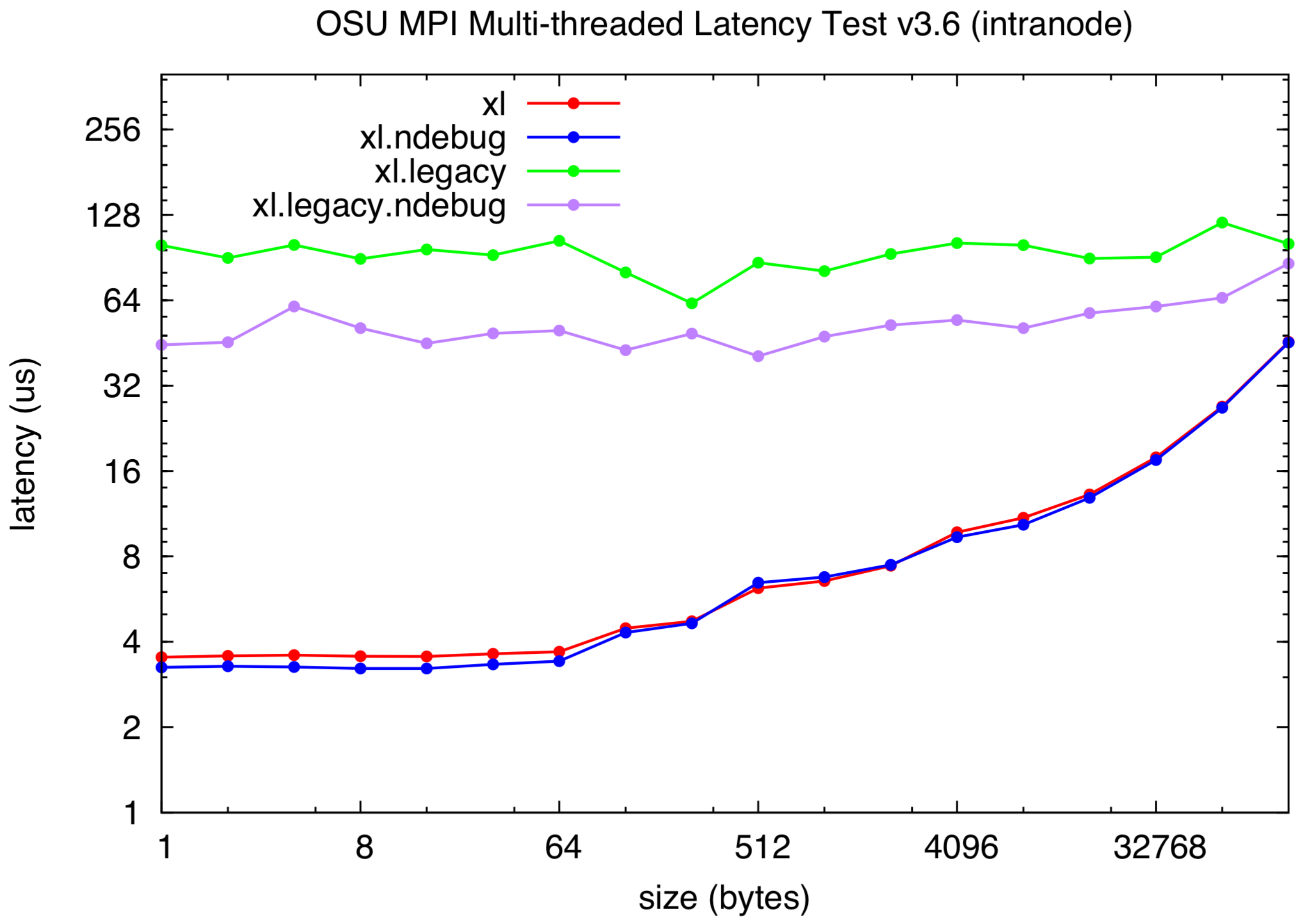
OSU MPI Latency Test v3.6 (internode)
gcc.legacy
xl
xl.ndebug
xl.legacy
xl.legacy.ndebug
latency (us)
32
16
8
4
2
1
1
8
64
512
4096
32768
size (bytes)

MPI on BG/Q



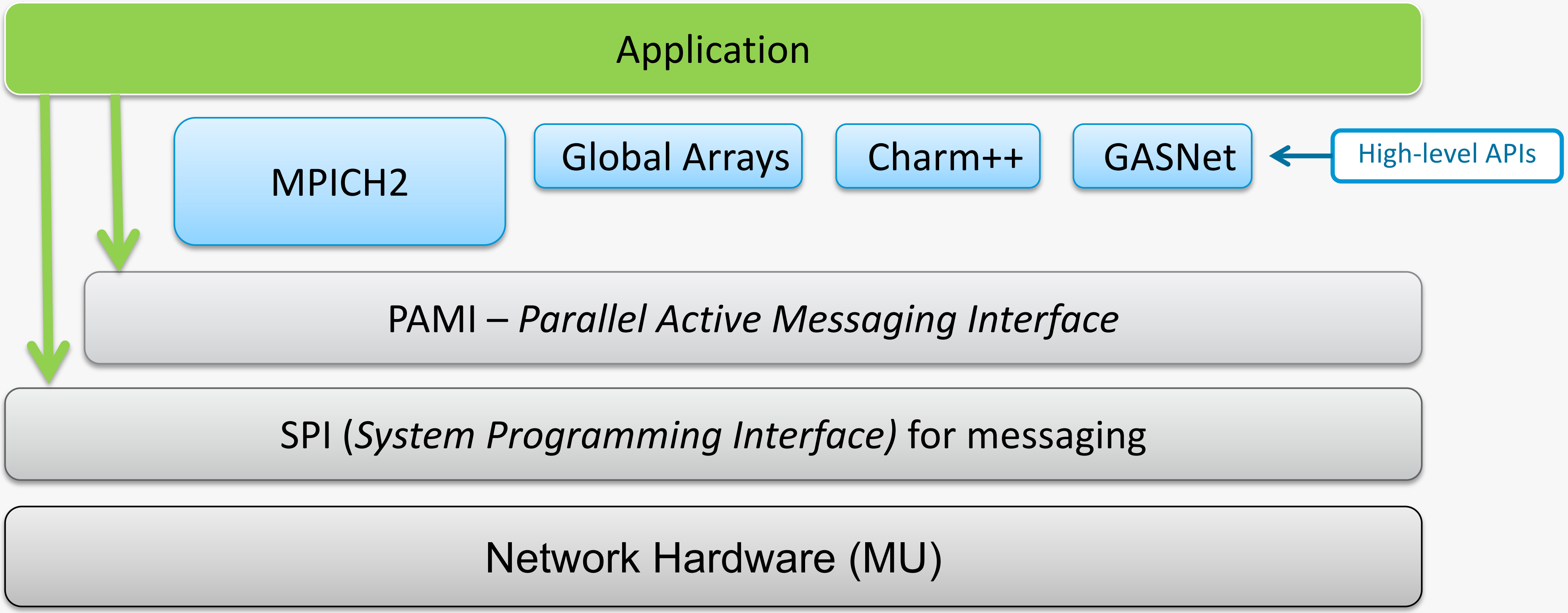
Argonne NATIONAL LABORATORY

MPI on BG/Q



OSU MPI Multi-threaded Latency Test v3.6 (intranode)
xl
xl.ndebug
xl.legacy
xl.legacy.ndebug
latency (us)
256
128
64
32
16
8
4
2
1
1
8
64
512
4096
32768
size (bytes)

PAMI: Where Does It Fit In?



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:

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

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

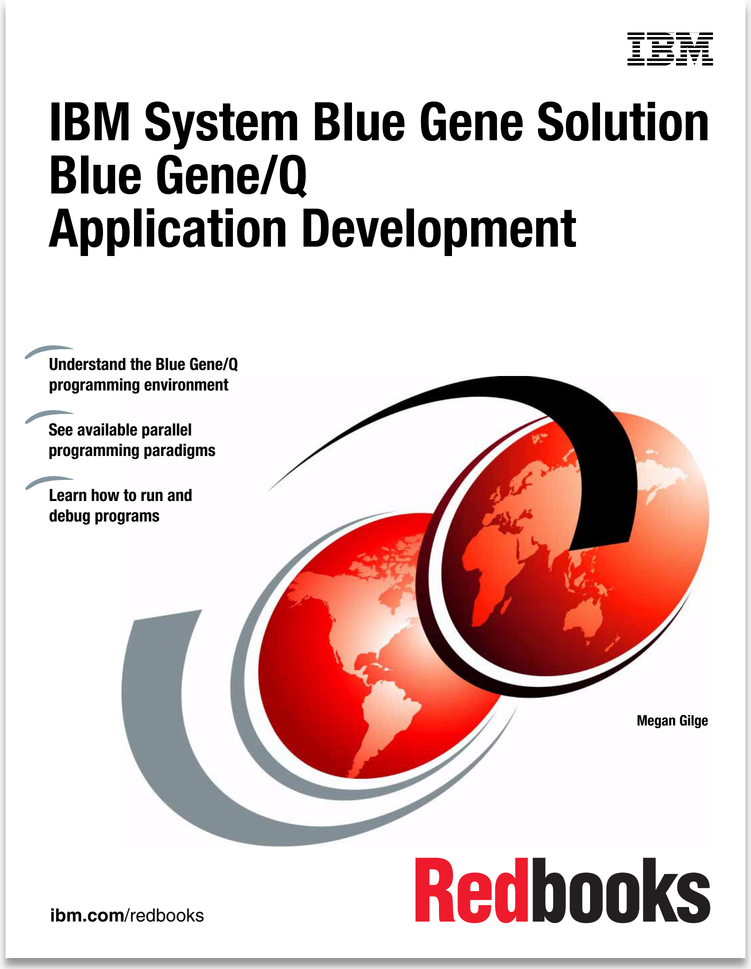
Writing Applications with PAMI and SPI: Just Say No

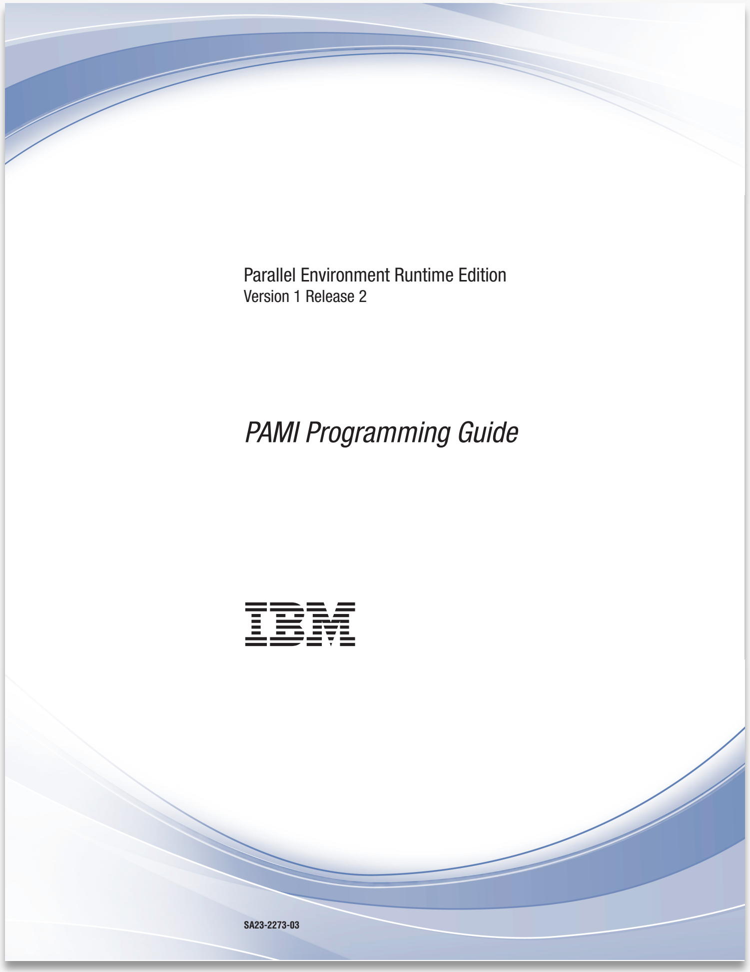
No vendor is carrying forward PAMI – we encourage users to explore alternatives.

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!