Mysteries of the Deep: What happens inside of MPI on Blue Gene/Q and why it matters

Jeff Hammond

Leadership Computing Facility
Argonne National Laboratory

May 23, 2012
The view from the boat
A reason to dive

Jeff Hammond

PAMI and MPI on BGQ
But not too deep

Jeff Hammond

PAMI and MPI on BGQ
Blue Gene/P Communication architecture

[Diagram showing the communication architecture with layers for Application, DCMF API (C), SPI, and Network Hardware. Key components include Converse/Charm++, MPICH, Global Arrays, ARMCI, UPC, and Other Paradigms. The diagram highlights protocols and devices such as pt2pt, collective, DMA Device, Collective Device, GI Device, Shmem Device, Message Layer Core (C++), and Network Hardware (DMA, Collective Network, Global Interrupt Network). Source: IBM]
Blue Gene/Q Communication architecture

Applications

MPICH2 2.x

CAF* Runtime

X10* Runtime

UPC* Runtime

GA ARMCI*

CHARM++*

GASNet*

IBM MPI 2.x

MPCI

APGAS Runtime

System Software

Middleware

PAMI API

BG/Q messaging implementation

x86 messaging implementation

PERCS messaging implementation

MU SPI

BG/Q

x86

HAL API

PERCS

*all runtimes are not supported on all platforms
Performance results
Testing injection (send) bandwidth along 1 to 10 links.
Explicitly mapped to torus.
Nonblocking recv and send followed by waitall.
No repetition in test but warmup along all 6 links done first.
Neighbor exchange

MPI neighbor exchange - total bandwidth (all links, all ranks)

bandwidth (MB/s) vs. message size (bytes)

1 link
2 links
3 links
4 links
5 links
6 links
7 links
8 links
9 links
10 links

Jeff Hammond
PAMI and MPI on BGQ
OSU MPI Latency Test v3.6 (internode)

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

Jeff Hammond  PAMI and MPI on BGQ
OSU MPI Benchmarks

OSU MPI Latency Test v3.6 (intranode)

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

Jeff Hammond
PAMI and MPI on BGQ
OSU MPI Benchmarks

OSU MPI Latency Test v3.6 (internode)

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

Jeff Hammond

PAMI and MPI on BGQ
OSU MPI Multi-threaded Latency Test v3.6 (intranode)

latency (us) vs size (bytes)

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

Jeff Hammond

PAMI and MPI on BGQ
OSU MPI Bandwidth Test v3.6 (intranode)

bandwidth (MB/s)
size (bytes)

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

Jeff Hammond
PAMI and MPI on BGQ
OSU MPI Bi-Directional Bandwidth Test v3.6 (internode)

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

Jeff Hammond
PAMI and MPI on BGQ
OSU MPI Bi-Directional Bandwidth Test v3.6 (intranode)

Bandwidth (MB/s) vs. Size (bytes)

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

Jeff Hammond
PAMI and MPI on BGQ
OSU MPI Benchmarks

OSU One Sided latency Test v3.6 (internode)

- Put - gcc.legacy
- Get - gcc.legacy
- Acc - gcc.legacy
- Put - xl.legacy.ndebug
- Get - xl.legacy.ndebug
- Acc - xl.legacy.ndebug

Jeff Hammond
PAMI and MPI on BGQ
Topology and MPIX
Topology API

From /bgsys/drivers/ppcfloor/comm/*/include/mpix.h

#define MPIX_TORUS_MAX_DIMS 5
typedef struct {
    unsigned ppn;
    unsigned coreID;
    unsigned torus_dimension;
    unsigned Size[MPIX_TORUS_MAX_DIMS];
    unsigned Coords[MPIX_TORUS_MAX_DIMS];
    unsigned isTorus[MPIX_TORUS_MAX_DIMS];
} MPIX_Hardware_t;
MPIX_Hardware_t hw;
MPIX_Hardware(&hw);

int hopCoord = (hw.Coords[0]+1) % (hw.Size[0]);
int tempCoords[MPIX_TORUS_MAX_DIMS+1] =
    { hopCoord, hw.Coords[1], hw.Coords[2],
      hw.Coords[3], hw.Coords[4], hw.coreID};
MPIX_Torus2rank(tempCoords, &rank_ap);
Environment variables
PAMI Verbosity

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

PAMID_VERBOSE
Increases the amount of information dumped during an MPI_Abort() call. *Disabled by default.*
PAMI High-level tuning options

PAMID\_CONTEXT\_POST
Handoff-based communication. *Enabled by default (I think).*
Optimizing collectives through thinking

**MPI\_Reduce\_scatter**: reduce a buffer to root, then scatter from root. This MPI-2.1 function requires vector arguments, so this is really reduce, then scatterv. MPI\_Scatterv is not optimized. In addition, the arguments to scatterv must be allocated internally. At scale, this consumes a lot of memory (perhaps as much as 8x).

**MPI\_Reduce\_scatter\_block**: reduce a buffer to root, then scatter from root. This MPI-2.2 function takes scalar arguments and therefore uses less memory at scale.

**MPI\_Allreduce**: reduce a buffer everywhere, then copy out my portion. This MPI-1 function requires scalar arguments and is highly optimized on Blue Gene/P. Using MPI\_IN\_PLACE means no extra memory is used (although the input buffer is modified).

On BGQ, allreduce will be faster than reduce at least some of the time.
“People first, then money, then things” – Suze Orman

“Science first, then algorithms, then communication” – Me

However, you know your algorithms are good and communication is holding you back, understanding the internals of MPI will help you write faster and more scalable code on Blue Gene/P.

P.S. If profiling shows you spend too much time in MPI_Barrier, your *algorithm* is the problem (load imbalance), not communication.