ALCF BLUE GENE /Q SYSTEMS
PART 2: INTER-NODE COMMUNICATION
BG/Q COMPUTE CHIP
BG/Q COMPUTE CHIP
BG/Q NETWORK

Network
BG/Q NETWORK

Network

5D Torus
BG/Q NETWORK

Network

A
B
C
D
E
BG/Q NETWORK

Network

A
B
C
D
E

T

Process number within node
BG/Q 512 NODE TORUS PARTITION
BG/Q 512 NODE TORUS PARTITION

4 × 4 × 4 × 4 × 2
MAPPING RANKS/PROCESSES TO NODES

- Permutation of ABCDET
  - ABCDET on midplane --mode c1
    - 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>
  
- Mapping file
  - 0 0 0 0 0 0 # rank 0
    0 0 0 1 0 0 # rank 1
    0 0 0 1 0 0 # rank 2
    ...
  - runjob --mapping mapfilename

- runjob --mapping TEDCBA
MAPPING RANKS/PROCESSES TO NODES (CONT'D)

- **Goal**: in cartesian topology
  - Preserve locality for nearest-neighbor
  - Minimize extra hops in partition

- **Example**: 2D logical topology
  - Midplane c16 <4,4,4,4,2,16>

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

```c
MPI_Comm_split(MPI_COMM_WORLD, color, key, new2DComm);
```

Order in $64 \times 128$
TOPOLOGY ACCESS: MPIX

#include <mpi.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
NETWORK SPEED IS A MAJOR STRENGTH OF BG/Q

- Each A/B/C/D/E link bandwidth: 4 GB/s
- Bisection bandwidth (32 racks): 13.1 TB/s
- HW latency
  - Best: 80 ns (nearest neighbor)
  - Worst: 3 μs (96-rack 20 PF system, 31 hops)
- MPI latency (zero-length, nearest-neighbor): 2.2 μs
BLUE GENE/Q COMMUNICATION PROGRAMMING

Application

MPICH2  Global Arrays  Charm++  GASNet

PAMI – Parallel Active Messaging Interface

SPI (System Programming Interface) for messaging

Network Hardware (MU)

High-level APIs
MPI ON BG/Q

- Based on MPICH
- Fully open source
- MPI-2.2
  - *Except* incompatible features (needing fork, e.g. MPI_Comm_spawn)
MPI ON BG/Q

+mpiwrapper-xl
MPI ON BG/Q

+mpiwrapper-xl

IBM XL compilers

Fine-grain locking
MPI ON BG/Q

+mpiwrapper-xl

Fine-grain locking

IBM XL compilers

Overlapping access to MPI by multiple threads
MPI ON BG/Q

+mpiwrapper-xl.legacy
MPI ON BG/Q

+mpiwrapper-xl.legacy

Coarse-grain locking
MPI ON BG/Q

+mpiwrapper-xl.legacy

Coarse-grain locking

Mutual exclusion between threads at MPI function level
MPI ON BG/Q

+mpiwrapper-xl.ndebug

+mpiwrapper-xl.legacy.ndebug

No error checking or asserts
MPI ON BG/Q

+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-3**

- No official support on BG/Q – consider it a supported beta
- Nonblocking collectives: *use PAMI*
- Remote Memory Access (RMA): *use PAMI*
- Other MPI-3 features:
  - MPI + MPIX + PAMI + SPI
- There’s also a OFI-based version under development
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>$\leq 112$ 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 $\geq 2048$ 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
REFERENCES

- PAMI Doxygen documentation
- /bgsys/drivers/ppcfloor/comm/sys/include/pami.h
- IPDS 2012 Talk (Sameer Kumar)
- OpenSHMEM 2013 talk (Alan Benner)
- Mysteries of the Deep (J. Hammond)
- pami-examples on Google Code
END