Performance Impact of Accelerated Portals

Ron Brightwell    Kevin Pedretti    Keith Underwood
Sandia National Laboratories
Center for Computation, Computers, Information, and Mathematics

Trammell Hudson
OS Research

Cray User Group Meeting
May 9, 2007
Outline

• SeaStar
• Portals
• Flow control protocols
• Performance results
  – Lots of micro-benchmarks
  – One application
• Ongoing and future work
SeaStar Block Diagram

- **Opteron CPU**
  - Memory Controller
  - HyperTransport
  - Local SRAM
  - PowerPC 440

- **SeaStar**
  - Router
  - TX DMA Engine
  - RX DMA Engine
  - Serial
  - Processor Local Bus

- **3-D Mesh**
  - 3-D Mesh

- **Host Memory**

- **RAS Controller**

- **SeaStar National Laboratories**
Portals 3.3 for SeaStar

- Cray started with Sandia reference implementation
- Needed single version of NIC firmware that supports all combinations of
  - User-level and kernel-level API
  - NIC-space and kernel-space library
- Cray added bridge layer to reference implementation to allow NAL to interface multiple API NALs and multiple library NALs
  - qkbridge for Catamount applications
  - ukbridge for Linux user-level applications
  - kbridge for Linux kernel-level applications
SeaStar NAL

- Portals processing in kernel-space
  - Interrupt-driven
  - “generic” mode
- Portals processing in NIC-space
  - No interrupts
  - “accelerated” mode
Flow Control Protocols

• CAM Overflow Remediation Protocol SystEm (CORPSE)
  – Sandia’s protocol that runs entirely on the SeaStar
• CAM Overflow Protocol
  – Cray’s protocol that runs entirely on the Opteron
Platform Info

- Pre-upgrade Red Storm
  - 10,368 2.0 GHz AMD Opteron
  - SeaStar 1.2
- Post-upgrade Red Storm
  - 12,960 2.4 GHz dual-core AMD Opteron
  - SeaStar 2.1
- Cray development tree with Accelerated Portals
Benchmarks

• MPI
  – Pallas MPI Benchmarks v2.1
  – Ohio State University streaming bandwidth
  – Sandia overhead benchmark
  – HPC Challenge Baseline RandomAccess
  – Rotate latency
    • Average latency between all pairs of processes

• GASNet performance
• Atomic memory operations
• NIC-based barrier
PMB PingPong Latency

![Graph showing PMB PingPong Latency with different message sizes and latency values.](image-url)
PMB PingPong Bandwidth

![Bandwidth Graph]

- SS1 Generic
- SS1 Accelerated
- SS2 Generic
- SS2 Accelerated

Bandwidth (MB/s) vs Message Size (Bytes)
PMB Sendrecv Bandwidth

[Graph showing bandwidth versus message size]
OSU Streaming Bandwidth

![Graph showing OSU Streaming Bandwidth with different message sizes and bandwidths for SS1 Generic, SS1 Accelerated, SS2 Generic, and SS2 Accelerated.](image)
SS2 - MPI Message Rate
SS1 - CPU Availability – Send
SS1 - CPU Availability – Receive

Default rendezvous mode for long MPI messages significantly decreases availability.
SS2 - Rotate Latency - SN
SS2 - HPCC Baseline RandomAccess

Accelerated mode increases small message throughput for dual-core nodes

![Graph showing GUP/s vs. Number of Processes]

- **GUP/s**
- **Number of Processes**
- **Generic**
- **Accelerated**
SS2 - Partition Transport – $12^3$ cells/processors

- Parallel Efficiency (%)
- Number of Processors

Graph showing the parallel efficiency (%) against the number of processors for accelerated and generic modes.
SS2 - Partisn Diffusion – $12^3$ cells/processors

Parallel Efficiency (%) vs Number of Processors

- Accelerated
- Generic
SS2 - Partisn Transport – $15^3$ cells/processors

![Graph showing parallel efficiency vs number of processors for Accelerated and Generic models.](image-url)
SS2 - Partisn Diffusion – $15^3$ cells/processors
Accelerated Portals AMOs
(Atomic Memory Operations)

• Perform an atomic operation on remote memory
  – PtlGetPut()  Return old value, swap in new one
  – PtlICGetPut()  Return old val, conditionally swap in new one
  – PtlGetAdd()  Return old val, add specified amount to val

  • PtlGetPut() part of Portals 3.3 spec; Cray added PtlICGetPut() and PtlGetAdd() as an optimization for SHMEM

  • Implementing NIC-based AMOs using SeaStar’s HTB_MAP[ ] MMR
    – SeaStar PPC440+localbus is 32-bit, host is 64-bit (40-bit)
    – HTB_MAP[ ] used to map in 256 MB regions of host memory
    – NIC-based AMO ops map in windows on-the-fly as necessary

  • Preliminary PtlGetAdd() results (non-pipelined):
    – Host-based (generic):  10.6 us per op
    – NIC-based (accelerated):  3.9 us per op
PtlGetAdd() Implementation at Target

- Generic Portals (Host-Based)
- Accelerated Portals (Host-Based)

**GetAdd Request**
- add=1

**GetAdd Response**
- old=5

**Diagram:**
- App
  - 5
  - 6
- Kernel
  - 5
  - +1
  - 6
- NIC
- GetAdd Request: add=1
- GetAdd Response: old=5
NIC-Based Barrier

![Graph showing time (in microseconds) vs. number of nodes for MPICH2 and PtlBarrier]
MPI Unexpected Messages

for (i=0; i<10000000; i++) {
    if ( rank == 0 ) {
        MPI_Send( NULL, 0, MPI_BYTE, 1, 0,
                   MPI_COMM_WORLD );
    } else if ( rank == 1) {
        MPI_Recv( NULL, 0, MPI_BYTE, 0, 0,
                   MPI_COMM_WORLD,       &MPI_Status );
    }
}

• Interrupts will cause the receiver to fall behind
GASNet Get Latency
GASNet Put Bandwidth
GASNet Get Bandwidth

![Graph showing GASNet Get Bandwidth vs Message Size (Bytes)]
Ongoing and Future Work

• Complete Accelerated implementation in Cray’s development tree
• Quantifying the benefit of Accelerated Portals
  – GASNet/UPC and Global Arrays
  – Impacts of quad core
• NIC-based reduction
  – Experimented with software floating-point
  – Achieved 20 MFLOPS
  – About 20 entries at 1 us interrupt overhead
Acknowledgments

• Sandia
  – Sue Kelly, Jim Laros and Courtenay Vaughan
  – David Holman
  – On-site Cray support team
• Cray
  – Portals development team