Portals 3.3 on the Sandia/Cray Red Storm System

Ron Brightwell    Trammell Hudson    Kevin Pedretti
Rolf Riesen       Keith Underwood

Sandia National Laboratories
Scalable Computing Systems Department

Cray User Group Annual Technical Conference
May 18, 2005
Outline

• Portals history
• Portals objects
• Portals implementation
• Portals for Red Storm
• Performance on Red Storm
• Conclusions
Portals Timeline

• Portals 0.0 - 1991
  – SUNMOS (Sandia/UNM OS)
  – nCUBE, Intel Paragon
  – Direct access to network FIFOs
  – Message co-processor

• Portals 1.0 - 1993
  – Data structures in user-space
  – Kernel-managed and user-managed memory descriptors
  – Published but never implemented

• Portals 2.0 - 1994
  – Puma/Cougar
  – Message selection (match lists)
  – Four types of memory descriptors (three implemented)

• Portals 3.0 - 1998
  – Cplant/Linux
  – Functional API
  – Target intelligent/programmable network interfaces
Portals 3.3 Features

- Best effort, in-order delivery
- Well-defined transport failure semantics
- Based on expected messages
- One-sided operations
  - Put, Get, Atomic swap
- Zero-copy
- OS-bypass
- Application offload
  - No polling or threads to move data
  - No host CPU overhead
- Runtime system independent
Portal Put

Initiator ─────── Data Transmission ─────── Target

Optional Acknowledgement

Portal Translation
Portal Get

Initiator

Request

Target

Portal Translation

Data Transmission
Portals Addressing

- Portal Table
- Match List
- Memory Descriptors
- Event Queue
- Memory Regions
- Access Control Table
- Portal Space
- Application Space
Match Entry Contents

- Source node id
- Source process id
- 64 match bits
- 64 ignore bits
Memory Descriptor

- Start address
  - Optionally supports gather/scatter list
- Length in bytes
- Threshold
  - Number of operations allowed
- Max size
  - Low-water mark
- Options
  - Put/get
  - Receiver/sender managed offset
  - Truncate
  - Ack/no ack
  - Ignore start/end events
- 64 bits of user data
- Event queue handle
- Auto-unlink option
Event Queue

- Circular queue that records operations on MDs
- Types of events
  - **Get** *(PTL_EVENT_GET_{START,END})*
    - MD has received a get request
  - **Put** *(PTL_EVENT_PUT_{START,END})*
    - MD has received a put request
  - **Reply** *(PTL_EVENT_REPLY_{START,END})*
    - MD has received a reply to a get request
  - **Send** *(PTL_EVENT_SEND_{START,END})*
    - Put request has been processed
  - **Ack** *(PTL_EVENT_ACK)*
    - MD has received an ack to a put request
Event Scenarios

- send start
- send end
- ack
- reply start
- reply end
- put
- put start
- put end
- get
- get start
- get end
Event Entry Contents

- Event type
- Initiator of event (nid,pid)
- Portal index
- Match bits
- Requested length
- Manipulated length
- Offset
- MD
- 64 bits of out-of-band data
- Link
- Sequence number
What Makes Portals Different?

• Provides elementary building blocks for supporting higher-level protocols well
• Allows structures to be placed in user-space, kernel-space, or NIC-space
• Receiver-managed offset allows for efficient and scalable buffering of “unexpected” messages
• Supports multiple protocols within a process
Portals Reference Implementation Design

API Space

Library Space

NAL

Transport
Myrinet Kernel Implementation

Kernel Space

Library

API

User Space

NAL

RTS/CTS MCP
Cray Portals Bridge

- 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

• Kevin already told you this 😊
• Portals library currently in kernel-space
  – Interrupt-driven
  – “generic”
• Portals library moving to NIC-space
  – No interrupts
  – “accelerated”
Standard Red Storm/XT3 Performance Disclaimer

- Performance results are from a snapshot of a developer code base
- Sandia C firmware stack
- Some features that may impact performance are not implemented
  - End-to-end reliability protocol
Micro-Benchmarks

- **PtlPerf**
  - Ping-pong latency and bandwidth (uni- and bi-directional)
  - Single, persistent ME, MD, EQ
  - Best-case performance for Portals

- **NetPIPE 3.6.2**
  - Ping-pong latency and bandwidth (uni- and bi-directional)
  - Streaming bandwidth
  - Implemented Portals module
PtIPerf Latency

![Graph showing latency vs. message size for different operations](image)

- **Latency (microseconds)**
- **Message Size (bytes)**

Legend:
- **get**
- **put**
- **put-bi**
PtlPerf Bandwidth (10KB-100KB)

![Graph showing bandwidth performance for different message sizes.

- X-axis: Message Size (bytes)
- Y-axis: Bandwidth (MB/s)

Legend:
- put
- put-bi
- get

Graph indicates that bandwidth increases as message size increases, with put operations generally having higher bandwidth compared to put-bi and get operations.]
PtlPerf Bandwidth (100KB-2MB)
NetPIPE Bandwidth

![Graph showing NetPIPE Bandwidth vs Message Size (bytes)]
Conclusions

• Portals 3.3 is the lowest-level network programming interface on Red Storm
• Cray bridge abstraction allows single instance of firmware to support multiple API and Library paths
• Interrupt-driven kernel-space Library implementation achieves ~4.8 µs
• Expect NIC-space Library implementation to do better
Questions?