A NEW VINTAGE OF COMPUTING CUG 2013

Napa Valley, California • May 6–9

SeaStar Unchained: Multiplying the Performance of the Cray SeaStar Network

20 Years of Excellence in Computational Science

OLCF

Oak Ridge Leadership Computing Facility

1992–2012

Dave Dillow
HPC Systems Engineer
Technology Integration Group
Oak Ridge Leadership Computing Facility
Oak Ridge National Laboratory
dillowda@ornl.gov

May 8, 2013

Agenda

- SeaStar hardware
- Portals Overview
- Portals on SeaStar/Linux
- CCI Overview
- CCI Prototype on SeaStar
- Results

SeaStar Hardware

- Focus on the NIC
  - 500 MHz PowerPC embedded processor
    - 32 KB data cache
    - 32 KB instruction cache
  - 384 KB of local memory
- Connected via HyperTransport to host
  - 800 MHz, 16 bit interface
  - 3.2 GB/s raw, over 2 GB/s achievable

SeaStar RX DMA Engine

- 256 entry RX FIFO
  - 1.4us from empty to full
  - Allows NIC to inspect message data
- 256 CAM entries
  - Source-to-DMA program lookup
  - 7 DMA entries per CAM

Portals

- Primary goal is scalability
  - No connection state at API level
  - Requires reliability from network
- Designed for *-bypass
  - Zero-copy, messages go directly into user-buffers
  - Application not required to progress messaging
    - Allows for communication/compute overlap
  - OS bypass considered, implemented for Catamount

Portals

- Buffers owned by application
  - Described by memory descriptors (MD)
    - Akin to memory registration
- Assigned to match lists for RX
  - Per-portal ID
  - First matching entry receives data
- Messages without a matching MD dropped

Portals on SeaStar/Linux

- Almost all API calls go to kernel
- Most protocol processing in kernel-space
  - Match list processing
  - Memory descriptor translation
- Requires two interrupts per message RX
  - First to request DMA program
  - Second to notify of completion
    - Can be optimized out for small packets
  - Resource exhaustion leads to dropped packets
    - Initiator notified, must retry

Why kernel based?

- Portals header contains trusted fields
  - Message length
  - Source Process ID
- Linux virtual memory
  - Non-contiguous address space
  - Requires page-pinning
- Faster processor than NIC
  - Better, larger caches
  - Faster memory
  - Slow bus crossing
  - NIC wins up to ~30 match entries in the list

Common Communication Interface

- Common layer over a variety of interconnects
  - Sockets (TCP for WAN, UDP for LAN)
  - Verbs (InfiniBand and iWarp)
  - Cray General Network Interface (Gemini and Aries)
  - Direct-Ethernet (in progress)
  - Myricom MX (partial, not updated)
  - Cray Portals (not updated)
- Openly available
  - https://github.com/cci

CCI Concepts

- Endpoints and connections
  - Endpoints own the resources
  - Try to minimize connection state
- Simple, event-based API
- Messages
  - Aimed for small data and control messages
  - Maximum message size limited
  - Buffered sends
  - Zero-copy receives
  - Fixed number of send/receive buffers for an endpoint
- RMA for large data movement

CCI on SeaStar

- User-space messaging model
- Memory regions shared between NIC and kernel
  - Mapped into application address space
  - NIC maps with HTB translation window and TLB entries
- Allows OS-bypass once initialized
- Application bypass
  - With caveats

CCI data regions

- Placed to minimize bus traffic
- Host per-endpoint resources
  - 32 KB completion queue (up to 8192 entries)
  - 32 KB ready ring (8192 entries)
  - 32 KB ready map (8192 entries)
  - 32 16 KB transmit buffers
  - 256 16 KB receive buffers (up to 8192 allowed)
- NIC owns Command Queue
  - 4 KB of uncached memory
  - 512 64-bit commands
  - Shared by all user-space endpoints

CCI SeaStar TX

- Kernel maintains minimal trusted header
  - Message type
  - Destination endpoint
  - Message length
- Application data is copied into buffer
  - Pre-registered with NIC
- TX command placed in NIC-hosted command queue
  - Only 3 entries in DMA program
    - Set destination
    - DMA trusted header
    - DMA user-space data

CCI SeaStar RX

- Firmware demultiplexes to user-space endpoints
  - Pre-registered buffers
  - Library fills RX ready ring as buffers available
- Minimal inspection of RX FIFO
- Simple calculation of DMA destination
- Entry placed in completion queue when complete

CCI SeaStar Buffer Management

- 64 byte chunks within the 16 KB area
  - Most efficient use of HT bus
- NIC tracks RX space
  - Notifies user-space when current buffer is exhausted
  - Notes number of messages received into buffer
- User-space manages refresh
  - Tracks RX events returned for a buffer
  - Once exhausted, waits for all events to be returned
  - Posts RX area back to ready-ring for reuse

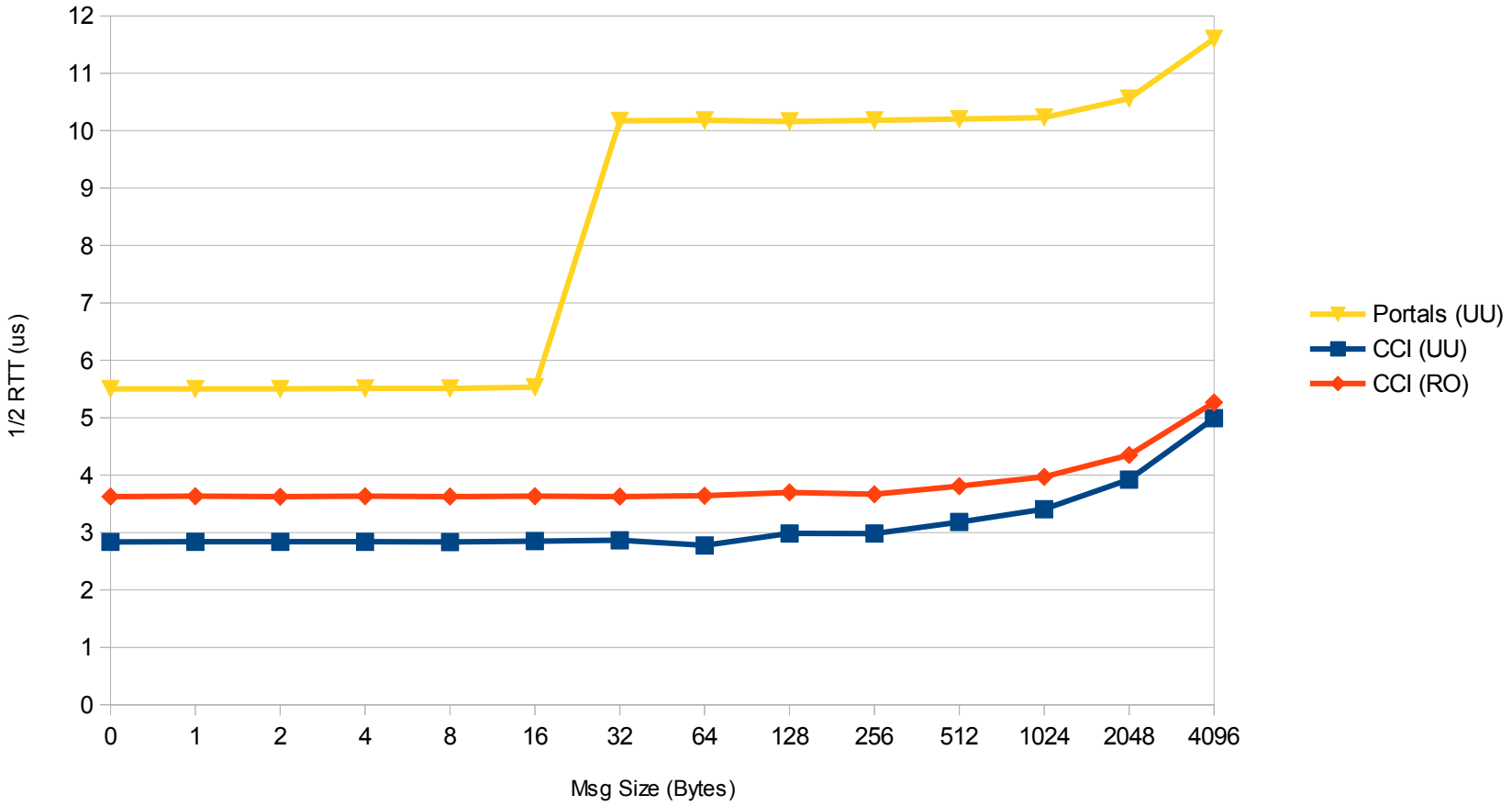
Prototype Limitations

- No RMA
- User-space endpoints only
- Command queue fairness
  - Process could be starved
  - Credit management not robust against crashed process
- Transmit scalability
  - Can only guarantee 96 sends queued per endpoint
  - May move to double-buffer, but increases copy costs
- User-driven progress
  - Mostly non-issue at present
    - Maximum received messages vs completion queue
- CAM swapping

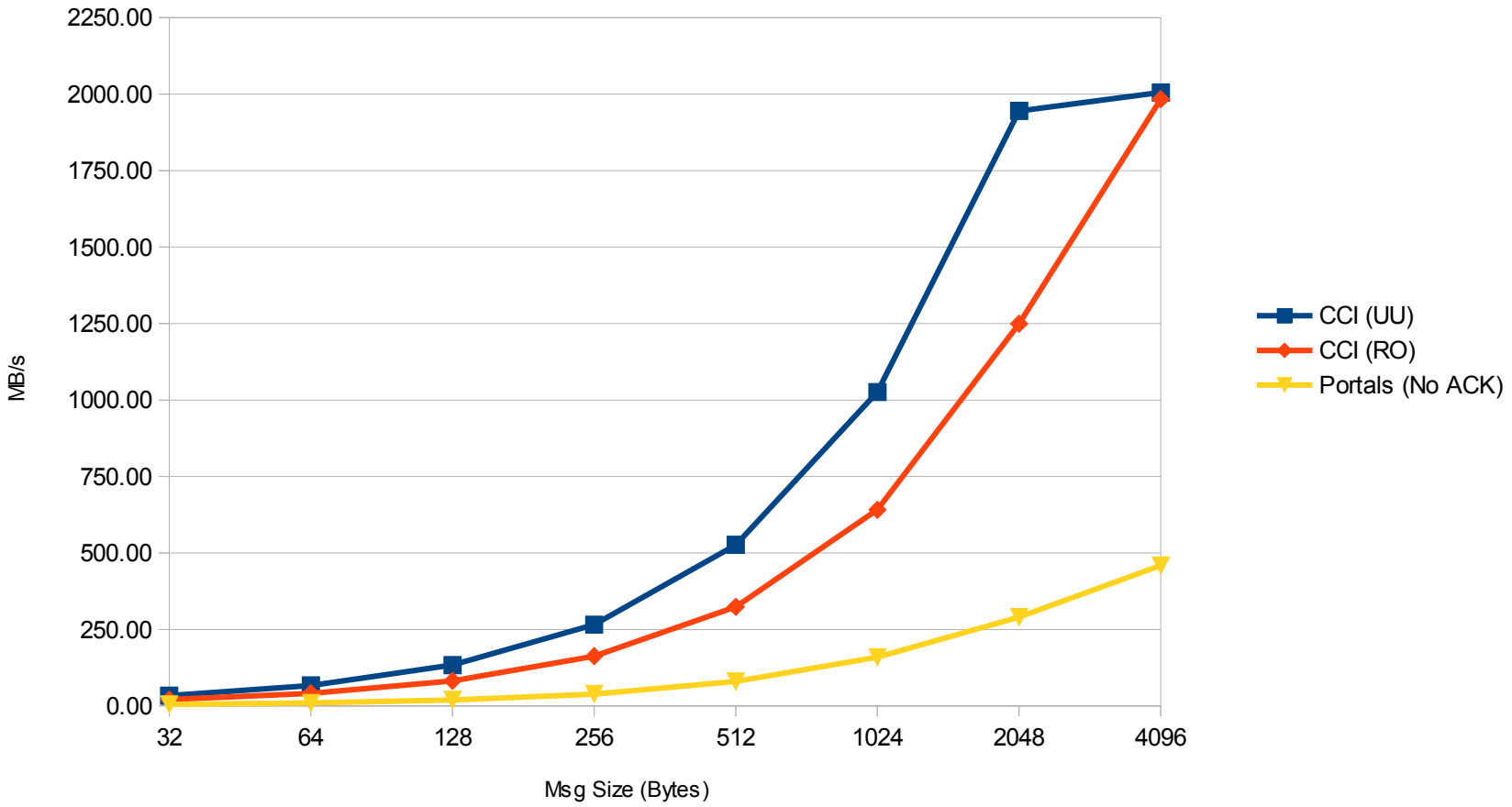
Test platform

- Cray XT5
  - 2.6 GHz 6-core Opterons
  - 16 GB DDR2 800 Mhz memory
- Portals
  - Cray CLE 2.2UP03
  - 2.6.16 kernel
- CCI
  - Custom compute image
  - 2.6.37 kernel

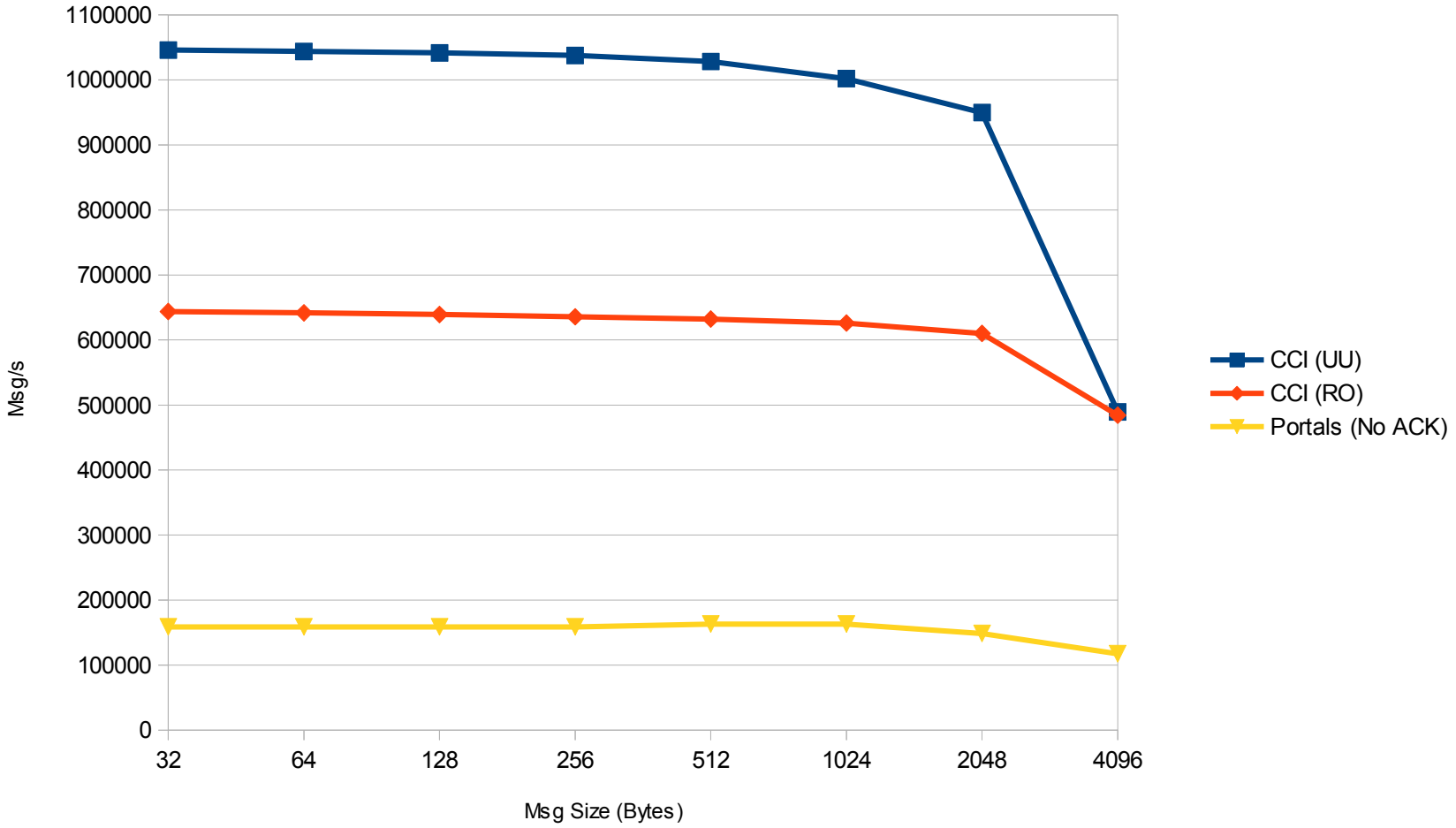
SeaStar Message Latency



SeaStar Streaming Rate



SeaStar Streaming Messages



What now?

- No further official work
  - SeaStar past end-of-life
  - OLCF no longer runs them
- But, as a hobby/educational experience...
  - Add kernel demultiplexed endpoints
  - RMA handling
  - Improve TX scalability
  - Try to optimize it further

Questions?