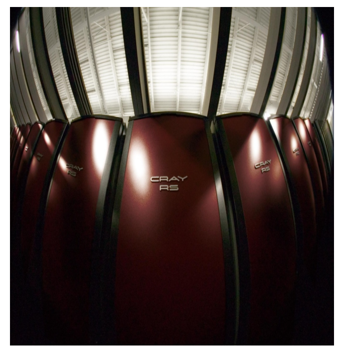
Exceptional service in the national interest

Sandia National Laboratories






Portals 4: Enabling Application/Architecture Co-Design for High-Performance Interconnects

Ron Brightwell, Technical Manager

Scalable System Software Department

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Outline

- History of Portals
- Building block approach
- Matching semantics between API and hardware
- Portals 4
  - Reference implementation
  - Triggered operations

Portals Network Programming Interface

- Network API developed by Sandia, U. New Mexico, Intel
- Previous generations of Portals deployed on several production massively parallel systems
  - 1993: 1800-node Intel Paragon (SUNMOS)
  - 1997: 10,000-node Intel ASCI Red (Puma/Cougar)
  - 1999: 1800-node Cplant cluster (Linux)
  - 2005: 10,000-node Cray Sandia Red Storm (Catamount)
  - 2009: 18,688-node Cray XT5 – ORNL Jaguar (Linux)
- Focused on providing
  - Lightweight “connectionless” model for massively parallel systems
  - Low latency, high bandwidth
  - Independent progress
  - Overlap of computation and communication
  - Scalable buffering semantics
  - Protocol building blocks to support higher-level application protocols and libraries and system services

Basic Assumptions

- A single low-level API is needed
  - Compute node OS doesn't have TCP/IP
  - Compute node application should own all network resources
- Applications will use multiple protocols simultaneously
  - Can't focus on just MPI
- Need to support communication between unrelated processes
  - Client/server communication between application processes and system services
- Need to support general-purpose interconnect capabilities
  - Can't assume special collective network hardware

What Makes Portals Different?

- One-sided communication with optional matching
- Provides elementary building blocks for supporting higher-level protocols well
- Allows key data structures to be placed where optimal for hardware
  - User-space, kernel-space, or NIC-space
- Allows for zero-copy and OS-bypass implementations
- Scalable buffering of MPI unexpected messages
- Supports multiple upper-level protocols (ULPs) within a process
- Run-time system independent
- Well-defined failure semantics

Design Philosophy – Don't Constrain

- Connectionless
  - Easy to do connections over connectionless
  - Impossible to do vice-versa
- One-sided
  - Easy to do two-sided over one-sided
  - Hard to do vice-versa
- Matching
  - Needed to enable flexible independent progress
  - Otherwise matching and progress must be done above
- Offload
  - Straightforward to onload API designed for offload
  - Hard to do vice-versa (see TOE)
- Progress
  - Must be implicit

ULPs Supported

- Application services
  - MPI-1, MPI-2, MPI-3 (send/recv, collective, one-sided)
    - MPICH, MPI/Pro, ChaMPIon/Pro, MPICH2, OpenMPI
  - PGAS
    - Cray SHMEM, OpenSHMEM, GASNet, ARMCI
  - MultiProcessor Computing (MPC)
  - CCI
- OS/Runtime services
  - Parallel job launch
    - Yod
  - File system and I/O
    - Fyod, Lustre
  - System calls
    - Remote procedure calls
  - IP
  - Qthreads runtime

Building Blocks Approach

- Define basic objects and operations that can be combined to simultaneously support multiple ULPs
  - Alternative approach is to define functions
  - Both approaches attempt to meet the semantics of the ULP as well as possible
- Pros
  - Supports a wider variety of upper-level protocols
  - Encapsulates important structures and functions
  - Enables specific hardware optimization opportunities
- Cons
  - More difficult to optimize for a single ULP
  - Can create interesting corner cases when combining objects and functions
  - Potential performance penalty for composability
  - Exposes implementation details

Mismatch Between Layers

RDMA

- One-sided
  - Allows process to read/write remote memory implicitly
- Zero-copy data transfer
  - No need for intermediate buffering in host memory
- Low CPU overhead
  - Decouples host processor from network
- Fixed memory resources
  - No unexpected Messages
- Supports unstructured, non-blocking data transfer
  - Completion is a local event

MPI Point-to-Point

- Two-sided
  - Short messages are copied
  - Long messages need rendezvous
- CPU involved in every message
  - Message matching
- Unexpected messages
  - Need flow control
- Completion may be non-local
  - Need control messages

How to Implement MPI over RDMA

1. Mvapich-Aptus: Scalable High-Performance Multi-Transport MPI over InfiniBand, Int'l Conference on Parallel and Distributed Computing, Miami, FL, Apr. 2008
2. Designing Passive Synchronization for MPI-2 One-Sided Communication to Maximize Overlap, Int'l Conference on Parallel and Distributed Computing, Miami, FL, Apr. 2008
3. MPI-2 One Sided Usage and Implementation for Read Modify Write operations: A case study with HPCC, EuroPVM/MPI 2007, Sept. 2007.
4. Zero-Copy Protocol for MPI using InfiniBand Unreliable Datagram, IEEE International Conference on Cluster Computing (Cluster'07), Austin, TX, September 2007.
5. High Performance MPI over iWARP: Early Experiences, Int'l Conference on Parallel Processing, XiAn, China, September 2007.
6. High Performance MPI Design using Unreliable Datagram for Ultra-Scale InfiniBand Clusters, 21st Int'l ACM Conference on Supercomputing, June 2007.
7. Reducing Connection Memory Requirements of MPI for InfiniBand Clusters: A Message Coalescing Approach, Int'l Symposium on Cluster Computing and the Grid (CCGrid), Rio de Janeiro - Brazil, May 2007
8. Hot-Spot Avoidance With Multi-Pathing Over InfiniBand: An MPI Perspective, Int'l Symposium on Cluster Computing and the Grid (CCGrid), Rio de Janeiro - Brazil, May 2007
9. High Performance MPI on IBM 12x InfiniBand Architecture, International Workshop on High-Level Parallel Programming Models and Supportive Environments (HIPS), held in conjunction with IPDPS '07, March 2007.
10. High-Performance and Scalable MPI over InfiniBand with Reduced Memory Usage: An In-Depth Performance Analysis, SuperComputing (SC 06), November, 2006.
11. Efficient Shared Memory and RDMA based design for MPI_Allgather over InfiniBand, EuroPVM/MPI, September 2006.
12. Memory Scalability Evaluation of the Next-Generation Intel Bensley Platform with InfiniBand, Hot Interconnect (HOTI 06), August, 2006.
13. MPI over uDAPL: Can High Performance andPortability Exist Across Architectures?, Int'l Sympsoium on Cluster Computing and the Grid (CCGrid), Singapore, May 2006.
14. Shared Receive Queue based Scalable MPI Design for InfiniBand Clusters, Int'l Parallel and Distributed Processing Symposium (IPDPS '06), April 2006, Rhode Island, Greece.
15. Adaptive Connection Management for Scalable MPI over InfiniBand , International Parallel and Distributed Processing Symposium
16. Efficient SMP-Aware MPI-Level Broadcast over InfiniBand's Hardware Multicast , Communication Architecture for Clusters (CAC) Workshop, to be held in conjunction with Int'l Parallel and Distributed Processing Symposium (IPDPS '06), April 2006, Rhode Island, Greece.
17. RDMA Read Based Rendezvous Protocol for MPI over InfiniBand: Design Alternatives and Benefits , Symposium on Principles and Practice of Parallel Programming,
18. High Performance RDMA Based All-to-all Broadcast for InfiniBand Clusters, International Conference on High Performance Computing (HiPC 2005)
19. Supporting MPI-2 One Sided Communication on Multi-Rail InfiniBand Clusters: Design Challenges and Performance Benefits , International Conference on High Performance Computing (HiPC 2005), December 18-21, 2005, Goa, India.
20. Designing a Portable MPI-2 over Modern Interconnects Using uDAPL Interface, EuroPVM/MPI 2005, Sept. 2005.
21. Efficient Hardware Multicast Group Management for Multiple MPI Communicators over InfiniBand, EuroPVM/MPI 2005, Sept. 2005.
22. Design Alternatives and Performance Trade-offs for Implementing MPI-2 over InfiniBand, EuroPVM/MPI 2005, Sept. 2005.
23. Can Memory-Less Network Adapters Benefit Next-Generation InfiniBand Systems?, Hot Interconnect (HOTI 05), August, 2005.
24. Analysis of Design Considerations for Optimizing Multi-Channel MPI over InfiniBand , Workshop on Communication Architecture on Clusters
25. Scheduling of MPI-2 One Sided Operations over InfiniBand, Workshop on Communication Architecture on Clusters (CAC 05) in conjunction with International Parallel and Distributed Processing Symposium
26. Building Multirail InfiniBand Clusters: MPI-Level Design and Performance Evaluation. SuperComputing Conference, Nov 6-12, 2004, Pittsburgh, Pennsylvania.
27. Efficient Barrier and Allreduce on IBA clusters using hardware multicast and adaptive algorithms, IEEE Cluster Computing 2004, Sept. 20-23 2004, San Diego, California.
28. Zero-Copy MPI Derived Datatype Communication over InfiniBand, EuroPVM/MPI 2004, Sept. 19-22 2004, Budapest, Hungary.
29. Efficient Implementation of MPI-2 Passive One-Sided Communication on InfiniBand Clusters , EuroPVM/MPI 2004, Sept. 19-22 2004, Budapest, Hungary.
30. Efficient and Scalable All-to-All Exchange for InfiniBand-based Clusters. International Conference on Parallel Processing (ICPP-04), Aug. 15-18, 2004, Montreal, Quebec, Canada.
31. Design and Implementation of MPICH2 over InfiniBand with RDMA Support. Int'l Parallel and Distributed Processing Symposium (IPDPS 04), April, 2004.
32. Fast and Scalable MPI-Level Broadcast using InfiniBand's Hardware Multicast Support. Int'l Parallel and Distributed Processing Symposium (IPDPS 04), April, 2004.
33. High Performance Implementation of MPI Datatype Communication over InfiniBand. Int'l Parallel and Distributed Processing Symposium (IPDPS 04), April, 2004.
34. Implementing Efficient and Scalable Flow Control Schemes in MPI over InfiniBand. Workshop on Communication Architecture for Clusters (CAC 04)
35. High Performance MPI-2 One-Sided Communication over InfiniBand. IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 04), April, 2004.
36. Fast and Scalable Barrier using RDMA and Multicast Mechanisms for InfiniBand-Based Clusters. Euro PVM/MPI Conference, September 29-Oct 2, 2003, Venice, Italy.
37. High Performance RDMA-Based MPI Implementation over InfiniBand. 17th Annual ACM International Conference on Supercomputing. San Francisco Bay Area. June, 2003.
38. Impact of On-Demand Connection Management in MPI over VIA , Cluster '02, Sept. 2002.

PSM Example (Todd Rimmer's slides from 2011 OFA Workshop)

A Bit of InfiniBand History

OPENFABRICS ALLIANCE

Early 2000's

- InfiniBand Inception
  - Designed for the enterprise data center market and an IO paradigm
  - Backbone network as a replacement for Ethernet and Fibre Channel
  - Incorporate best data center features of all interconnects and protocols
  - Cluster sizes: 100s of nodes
  - Performance Req: 1M IOPs

IB Verbs Design Point

Mid-2000's

- InfiniBand Finds Its Niche
  - High Performance Computing Clusters market
  - Low-Latency / High Bandwidth advantages
  - Primary message paradigm – MPI
  - Cluster sizes: 1000s of nodes
  - Performance Req: tens of millions of messages per second

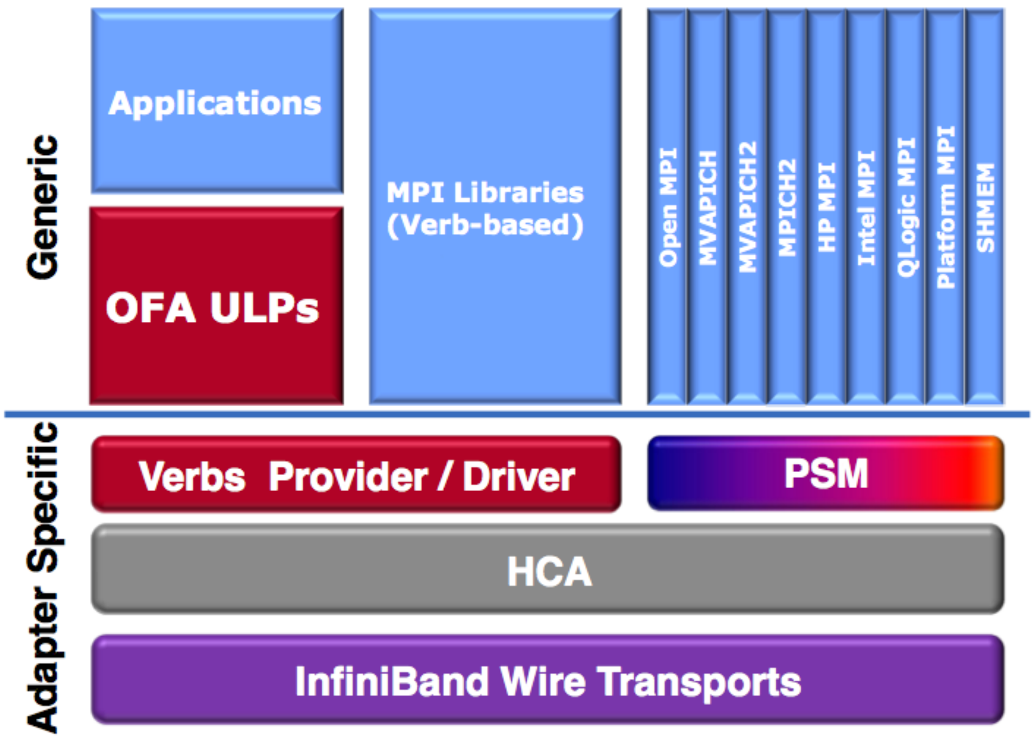
PSM Design Point

April 2011 4

PSM Design Focus

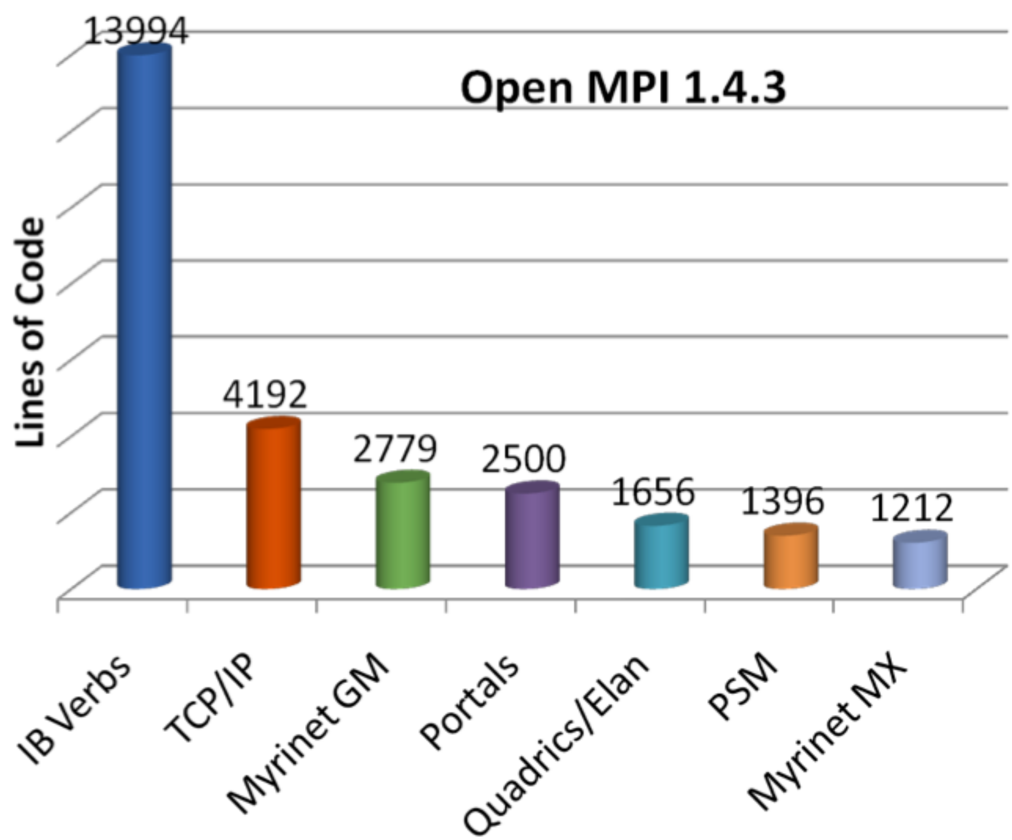
- **Focus on the needs of MPI and HPC Compute**
- **Design for very high HPC messaging rate, scalable latency**
  - Allow full support of all MPI collective implementations
- **Maintain a minimal memory footprint**
  - No adapter state per connection, no caches in adapter
  - Minimal memory footprint per end point
  - Scale out to large job size
- **Provide the high degree of resiliency needed by HPC**
  - More sophisticated retry/timeout mechanisms
  - More persistence
  - (IBTA Verbs limited to 7 retries at fixed timeout interval)
- **Overcome weaknesses in IO oriented IB transport**
  - Out of order packet handling, dispersive send of individual messages, etc
  - While interoperating with standard IB link and network layers
- **Centralize implementation of sophisticated features**
  - Allow all MPIs to easily take advantage

Integration of PSM with Open Fabrics



Comparison of Impedance Match OpenMPI MTL and BTL sizes

- Interfacing to PSM is comparable to HPC focused interconnects
  - Such as Quadrics, Myrinet
- Relative sizes are similar for other MPIs
  - mvapich, mvapich2, etc



Qlogic PSM

- Verbs didn't look like MPI at all
- Designed PSM after looking at how MPI implementation was architected
- Peer to verbs – use verbs for I/O
- Tag matching in library
- Latency and message rate at heart of collectives
  - Not progress

- What if you need to support more than MPI and SHMEM?
- What if MPI-3 adds non-blocking collectives?

Fixing the Mismatch

- Adapt the interface to the ULP
  - Myrinet
    - GM -> MX
  - InfiniBand
    - Verbs -> PSM
  - Portals
    - 0 -> 1.0 -> 2.0 -> 3.0 -> 4.0
- Adapt the hardware to the ULP
  - InfiniBand
    - +SRQ
    - +XRC
    - +collective offload
    - +MXM

Portals 4.0: Applying lessons learned from Cray SeaStar

- Allow for higher message rate
  - Atomic search and post for MPI receives required round-trip across PCI
  - Eliminate round-trip by having Portals manage unexpected messages
- Provide explicit flow control
  - Encourages well-behaved applications
    - Fail fast
    - Identify application scalability issues early
  - Resource exhaustion caused unrecoverable failure
  - Recovery doesn't have to be fast
  - Resource exhaustion will disable Portal
  - Subsequent messages will fail with event notification at initiator
  - Applies back pressure from network
    - Performance for scalable applications
    - Correctness for non-scalable applications

Portals 4.0 (cont'd)

- Enable a better hardware implementation
  - Designed for intelligent or programmable NICs
  - Arbitrary list insertion is bad
  - Remove unneeded symmetry on initiator and target objects
- New functionality for one-sided operations
  - Eliminate matching information
    - Smaller network header
    - Minimize processing at target
  - Scalable event delivery
    - Lightweight counting events
  - Triggered operations
    - Chain sequence of data movement operations
    - Asynchronous collective operations
      - Mitigate OS noise effects
    - Triggered rendezvous protocol
      - Enables progress without bandwidth penalty

Portals 4 Reference Implementation

- OpenFabrics Verbs
  - Provided by System Fabric Works
  - Provides a high-performance reference implementation for experimentation
  - Help identify issues with API, semantics, performance, etc.
  - Independent analysis of the specification
- Shared memory
  - Offers consistent and understandable performance characteristics
  - Provides ability to accurately measure instruction count for Portals operations
  - Better characterization of operations that impact latency and message rate
  - Evaluation of single-core onloading performance limits

http://code.google.com/p/**portals4**/

Reference implementation issues

- Extra layer of software between ULP and hardware
  - Impacts latency performance
  - More software layers is never good (unless you're IBM)
- Needs a progress thread
  - Impacts latency performance
  - Issues when ULP wants a progress thread too

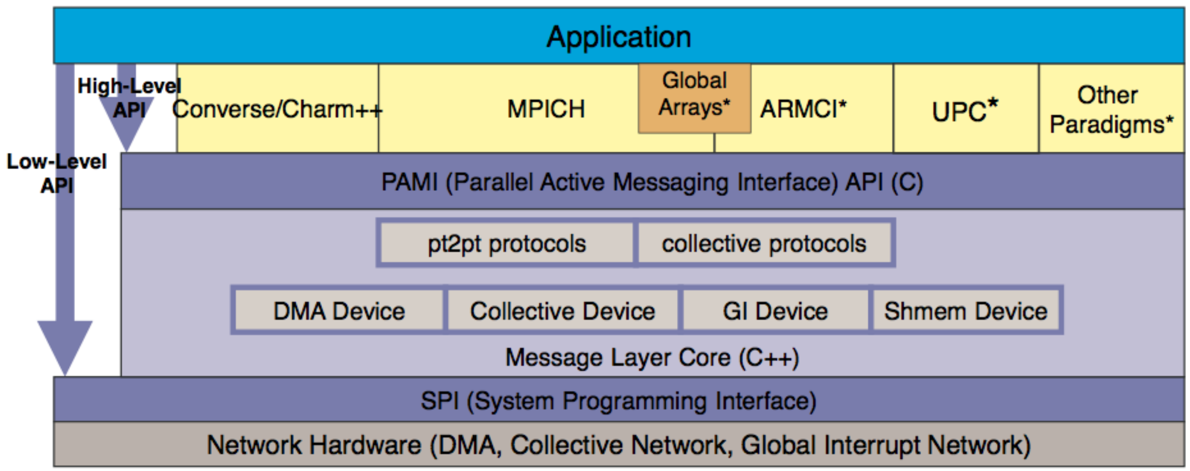
- Do we modify API for hardware we have or continue to design for hardware we need?
- Will two-node 0-byte ping-pong ever become irrelevant?

More Layers = More Performance?

IBM System Technology Group

IBM

Parallel Active Message Interface



- **Message Layer Core has C++ message classes and other utilities to program the different network devices**
- **Support many programming paradigms**
- **PAMI runtime layer allows uniformity across IBM HPC platforms**

***describes capability not necessarily product support**

MPI Messaging: Latency & Bandwidth

- Cielo has anisotropic link characteristics with X&Z having twice the width of the Y dimension
- Therefore messaging has to be characterized across two separate dimensions

Cielo's Gemini

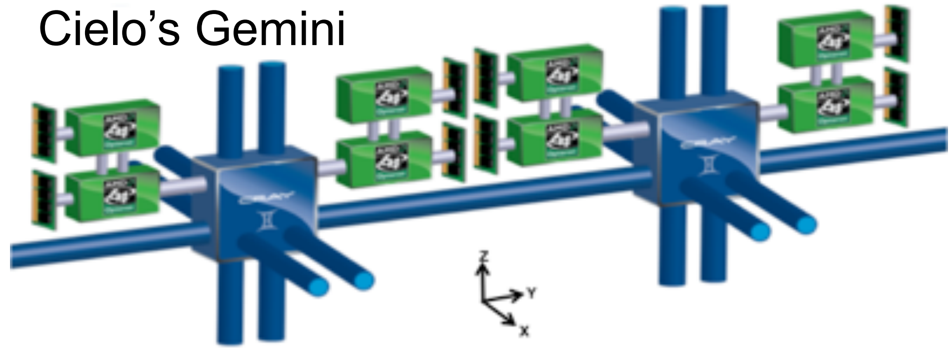
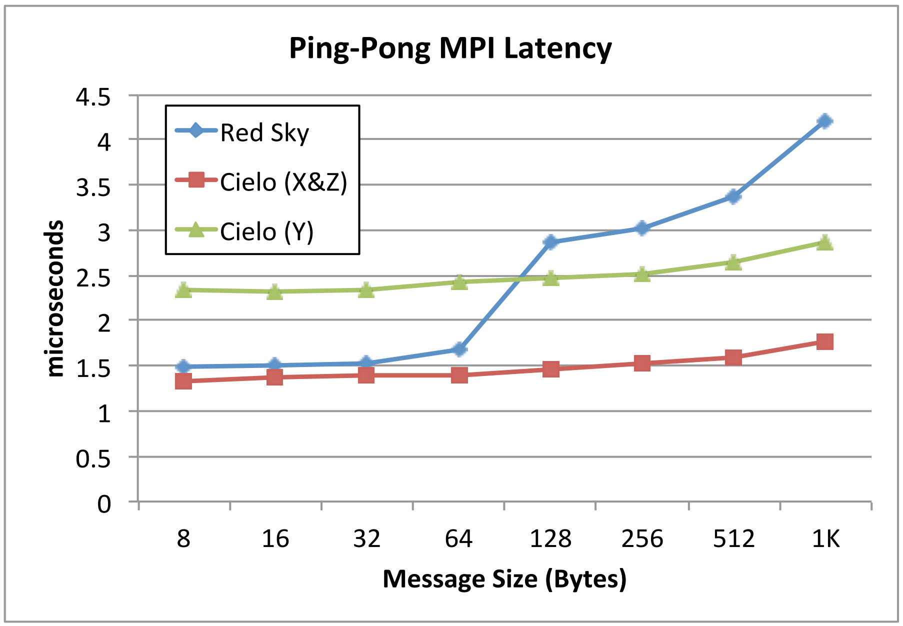
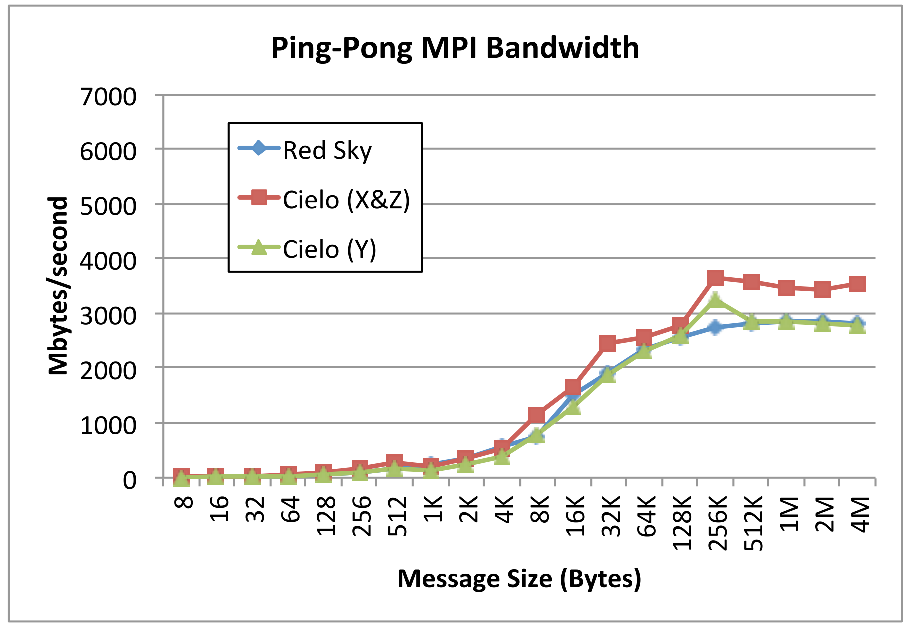
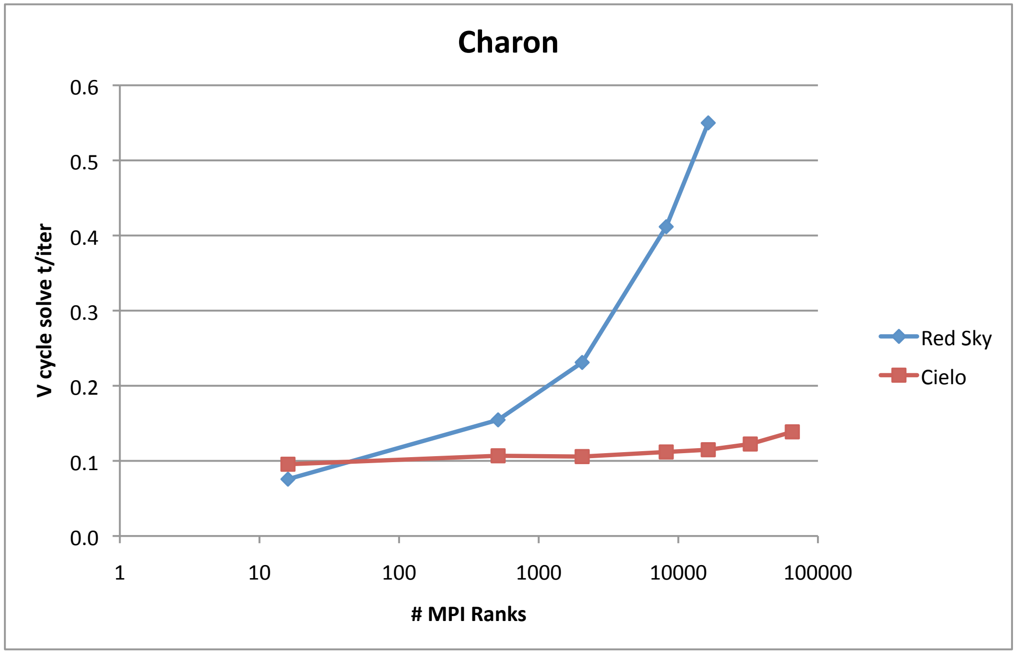


Image Courtesy of Cray, Inc.



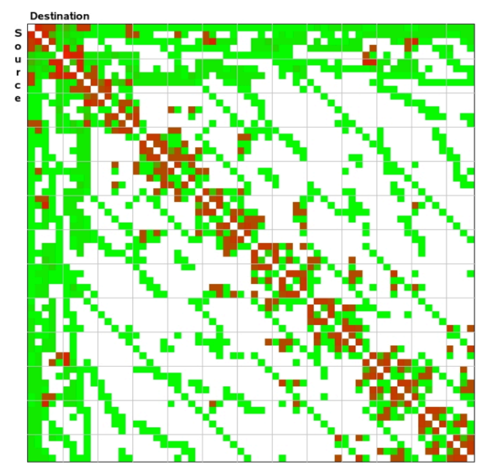


Cielo 6x application: SNL's Charon; Weak Scaling; (lower better)



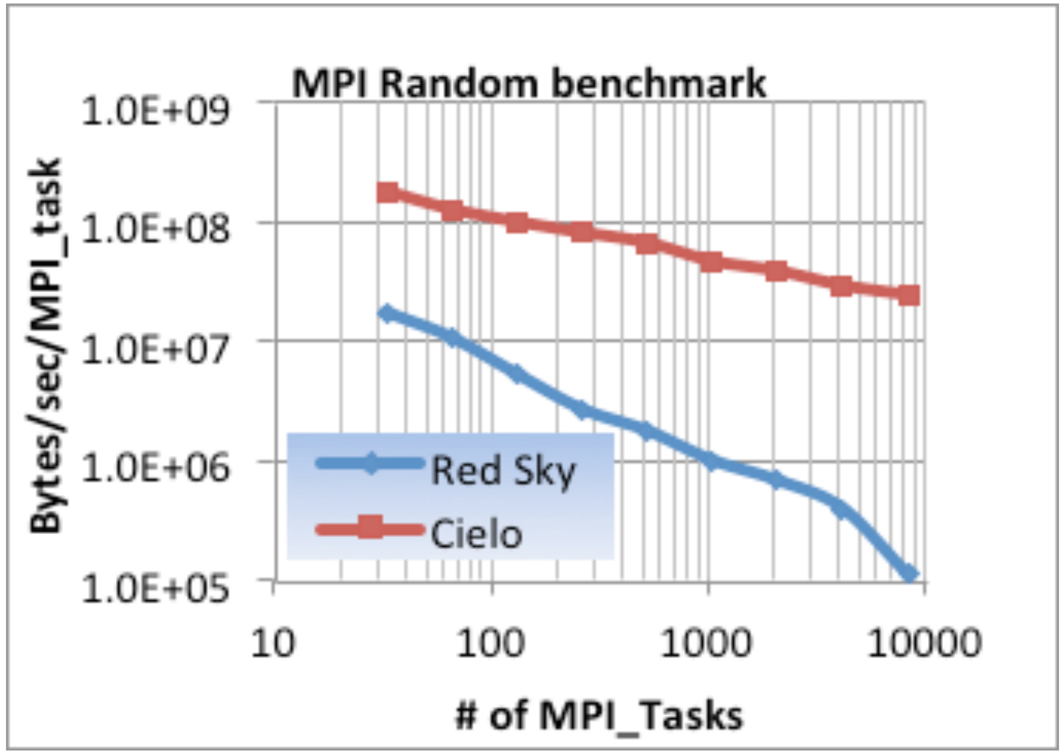
- Semiconductor device simulation code
- Finite element discretization produces a sparse, strongly coupled nonlinear system.
- A fully-coupled implicit Newton-Krylov solver is used with a multigrid preconditioner
- Performance dominated by small message transfers, avg 900 bytes, and small message Allreduce at scale

64 PE CrayPat plot



MPI Random Messaging Benchmark

- 100 B to 1 KB messages are sent to random MPI rank destinations
- Average message rate for all ranks is reported
- Red Sky is a factor of 10 slower at 32 PEs, factor of 220 slower at 8K PEs
- Here's one of our first indications of a deficiency in Red Sky's communication fabric
- What correlation is there with a real application?



Challenge areas for HPC networks

- The traditional “big three”
  - Bandwidth
  - Latency
  - Message Rate (Throughput)

- Other important areas for “real applications” versus benchmarks
  - Allowable outstanding messages
  - Host memory bandwidth usage
  - Noise (threading, cache effects)
  - Synchronization
  - Progress
  - Topology
  - Reliability

MPI Will Likely Persist Into Exascale Era

- Number of network endpoints will increase significantly (5-50x)
- Memory and power will be dominant resources to manage
  - Networks must be power and energy efficient
  - Data movement must be carefully managed
  - Memory copies will be very expensive
- Impact of unexpected messages must be minimized
  - Eager protocol for short messages leads to receive-side buffering
  - Need strategies for managing host buffer resources
  - Flow control will be critical
  - N-to-1 communication patterns will (continue to be) disastrous
- Must preserve key network performance characteristics
  - Latency
  - Bandwidth
  - Message rate (throughput)
  - Progress
  - Overlap

Portals Triggered Operations

- Lightweight events are counters of various network transactions
  - One counter can be attached to multiple different operations or even types of operations
  - Fine grained control of what you count is provided
- Portals operation is "triggered" when a counter reaches a threshold specified in the operations
  - Various types of operations can be triggered
  - Triggered counter update allows chaining of local operations

Motivation

- Collectives are important to a broad array of applications
  - As node counts grow, it becomes hard to keep collective time low
- Offload provides a mechanism to reduce collective time
  - Eliminates portion of Host-to-NIC latency from the critical path
  - Relatively complex collective algorithms are constantly refined and tuned
- Building blocks provide a better
  - Allow algorithm research and implementation to occur on the host
  - Provides a simple set of hardware mechanisms to implement
- A general purpose API is needed to express the building blocks

Generality of Triggered Operations

- Numerous collectives have been implemented so far
  - Allreduce
  - Bcast
  - Barrier
- Numerous algorithms have been implemented for multiple collectives
  - Binary tree
  - k-nomial tree
  - Pipelined broadcast
  - Dissemination barrier
  - Recursive doubling

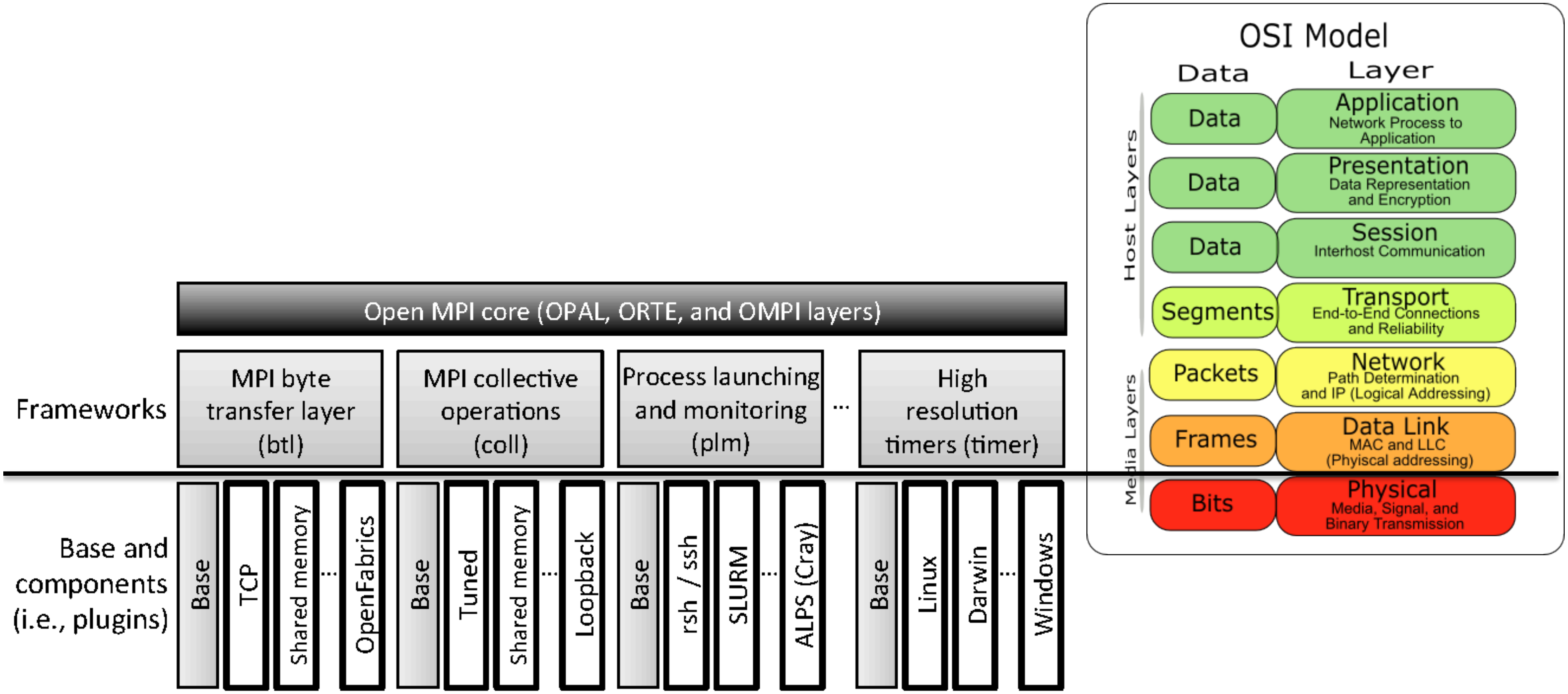
Brian Barrett, Ron Brightwell and Keith Underwood."A Low Impact Flow Control Implementation For Offload Communication Interfaces," in Proceedings of the 2012 European MPI Users' Group Conference, September 2012.

RECEIVER-MANAGED FLOW CONTROL

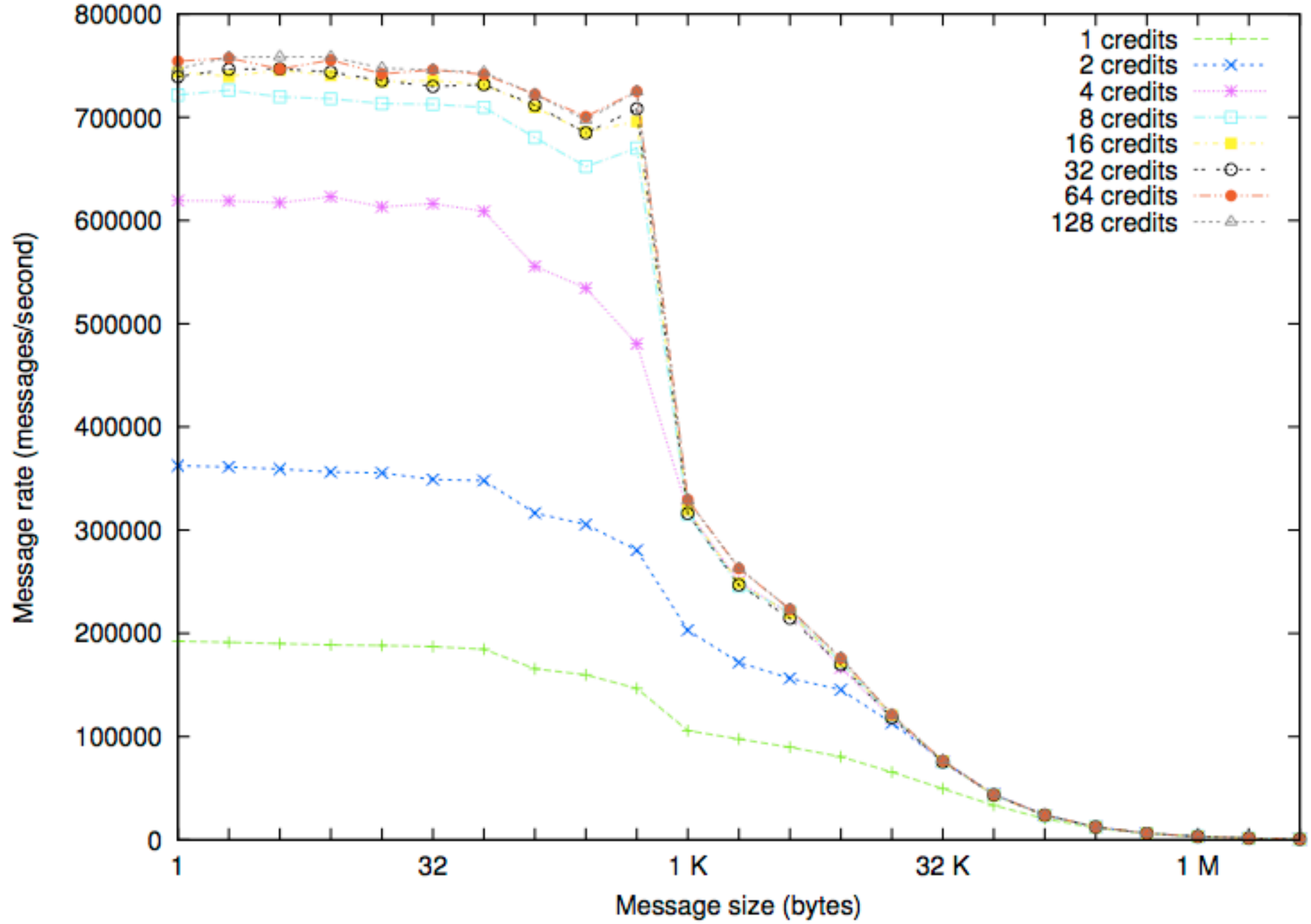
MPI Requires Flow Control

- MPI-1 Standard mandates that senders cannot overwhelm receivers with too many unexpected messages
- Two main strategies for providing flow control
  - Credit-based
    - A credit is needed to send a message
    - Credits given out at initialization or connection setup
    - Credits can be static or dynamic based on message intensity
    - Credits exchanged through explicit or piggyback messages
    - Potential performance penalty for well-behaved applications
  - Acknowledgment-based
    - Wait for receiver to acknowledge message reception
    - ACKs can be explicit or piggyback messages
    - Performance penalty for every application
- Both strategies assume senders need to be constrained
- Flow control implemented at user-level inside MPI library
- Network transport usually has its own flow control mechanism
  - No mechanism for back pressure from host resources to network
- Our approach is to recover rather than constrain
  - Emphasize performance for well-designed applications
  - Provide correctness for poorly-designed applications

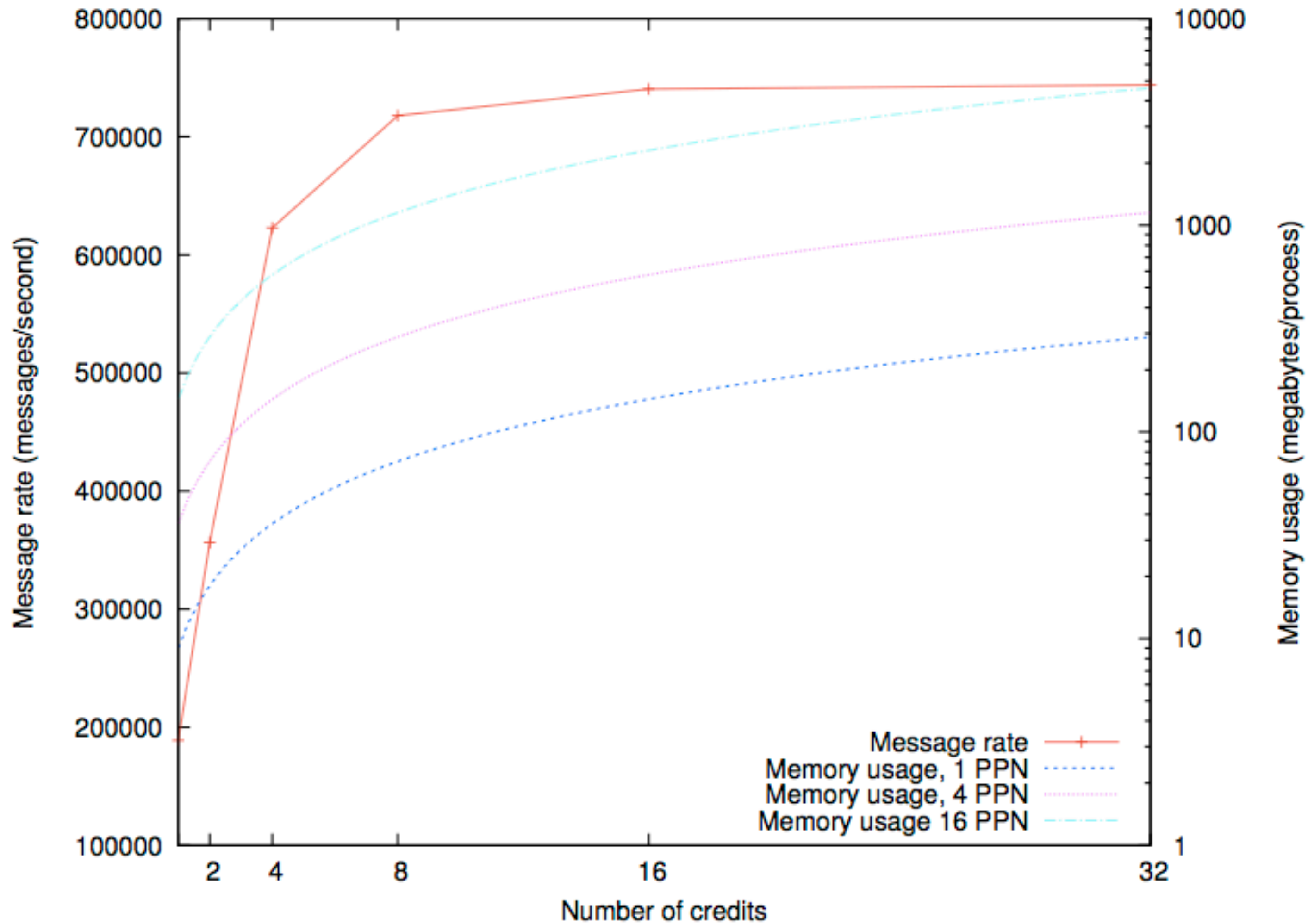
User-level Networks Duplicate Much of The Stack



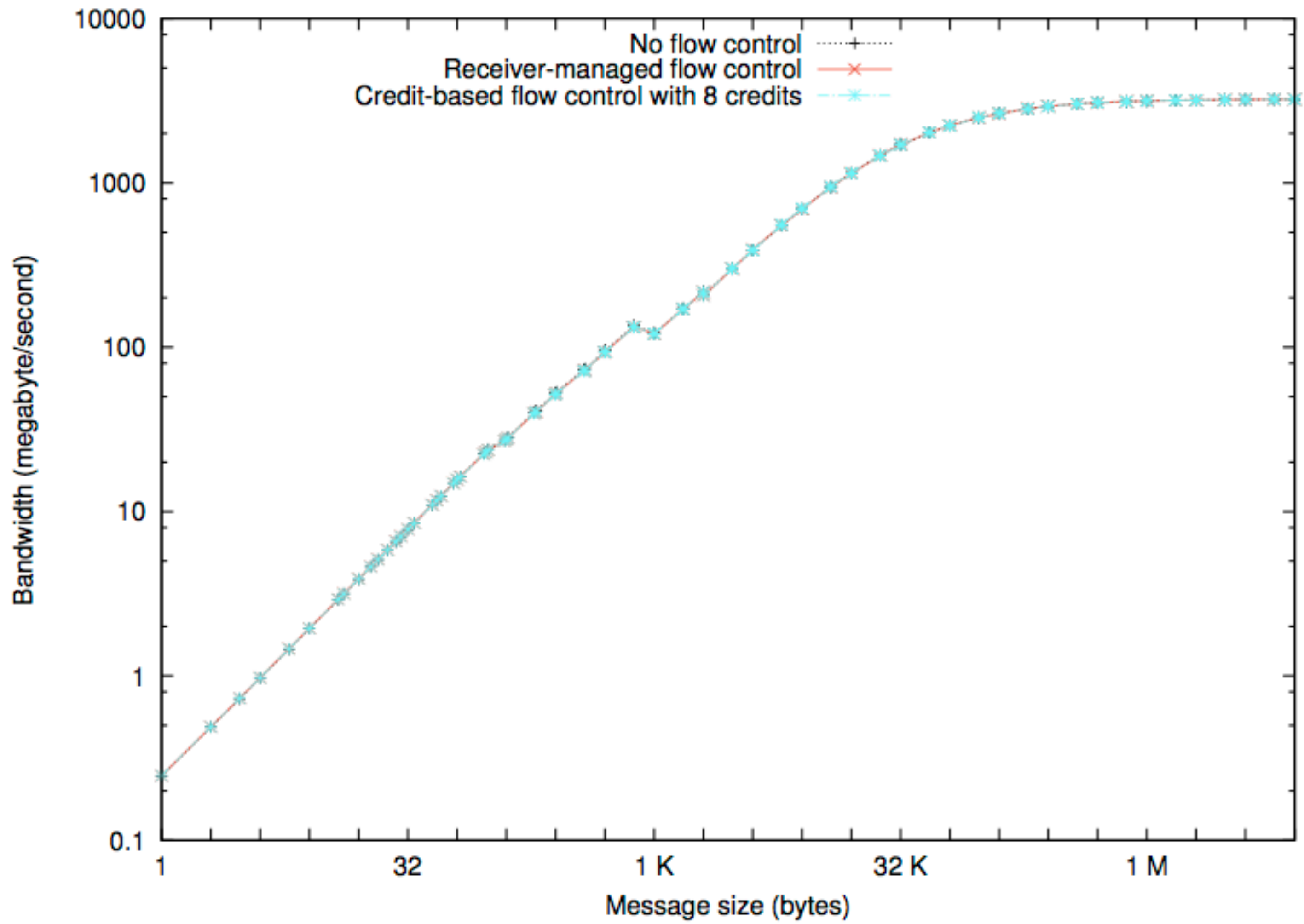
Too few credits can reduce message rate



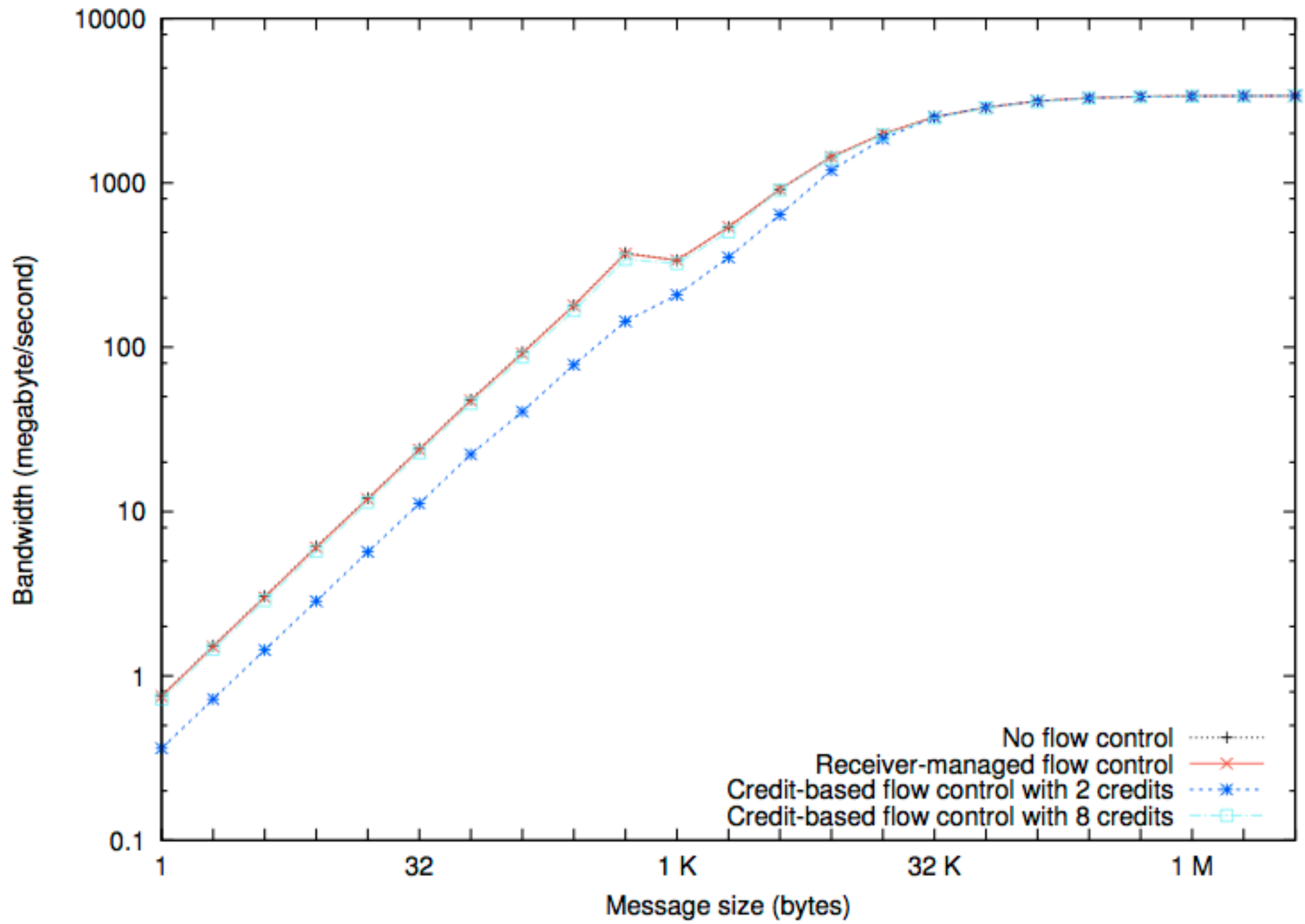
Too many credits wastes memory



Ping-pong bandwidth is unimpacted



Too few credits degrades streaming bandwidth



Summary

- Portals 4 provides building blocks that support many ULPs
- Encapsulates required semantics in a single API
- Design decisions based on least constraints
- Reference implementation available
  - Trying to figure out how to not be just another layer of software
- Triggered operations can implement
  - Non-blocking collective operations
  - Efficient rendezvous protocol for long messages
  - Recovery-based flow control for MPI

Acknowledgments

- Sandia
  - Brian Barrett
  - Scott Hemmert
  - Kevin Pedretti
  - Mike Levenhagen
  - Kyle Wheeler
- Intel
  - Keith Underwood
  - Jerrie Coffman
  - Roy Larsen
- System Fabric Works
  - Bob Pearson
  - Frank Zago

http://www.cs.sandia.gov/Portals