Sandia National Laboratories

Exceptional service in the national interest

THE LAST DIFFERENTIATING CAPABILITY OF HPC SYSTEMS IS FADING

Ron Brightwell

Scalable System Software Department

Center for Computing Research

Multicore World – February 15, 2024

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

FOLLOWING ANDREW'S LEAD

A

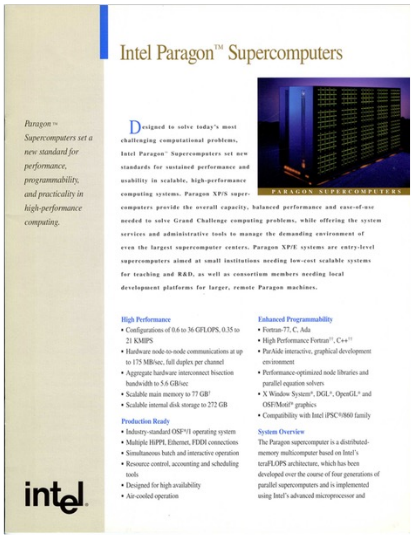
B

C

GOALS FOR THIS TALK

- Share my interconnect hardware/software co-design journey
- Provide a perspective on how HPC interconnect ecosystem has been impacted by cloud
- Tell some stories
- Take credit for co-design success that occurred mostly through serendipity
- Discuss challenges that need to be addressed to ensure interconnect technologies continue to meet the requirements of extreme-scale systems
- Re-use as many slides as possible (mine and others)

INTEL PARAGON NODE (CIRCA 1993)



Intel Paragon™ Supercomputers

Paragon™ Supercomputers set a new standard for performance, programmability, and practicality in high-performance computing.

Designed to solve today's most challenging computational problems, Intel Paragon™ Supercomputers set new standards for sustained performance and usability in scalable, high-performance computing systems. Paragon XP/S supercomputers provide the overall capacity, balanced performance and ease-of-use needed to solve Grand Challenge computing problems, while offering the system services and administrative tools to manage the demanding environment of even the largest supercomputer centers. Paragon XP/E systems are entry-level supercomputers aimed at small institutions needing low-cost scalable systems for teaching and R&D, as well as consortium members needing local development platforms for larger, remote Paragon machines.

PARAGON SUPERCOMPUTERS

High Performance
- Configurations of 0.6 to 36 GFLOPS, 0.35 to 21 KMIPS
- Hardware node-to-node communications at up to 175 MB/sec, full duplex per channel
- Aggregate hardware interconnect bisection bandwidth to 5.6 GB/sec
- Scalable main memory to 77 GB†
- Scalable internal disk storage to 272 GB

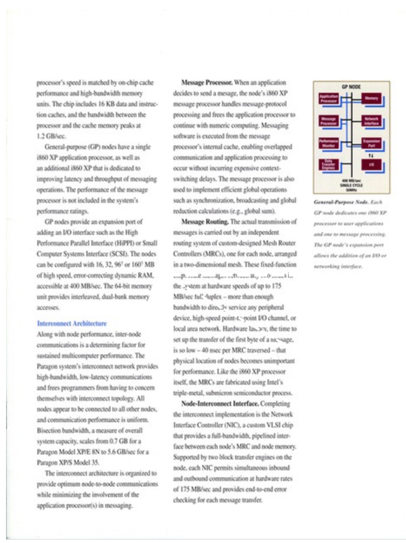
Production Ready
- Industry-standard OSF*/1 operating system
- Multiple HiPPI, Ethernet, FDDI connections
- Simultaneous batch and interactive operation
- Resource control, accounting and scheduling tools
- Designed for high availability
- Air-cooled operation

Enhanced Programmability
- Fortran-77, C, Ada
- High Performance Fortran††, C++††
- ParAide interactive, graphical development environment
- Performance-optimized node libraries and parallel equation solvers
- X Window System*, DGL*, OpenGL* and OSF/Motif* graphics
- Compatibility with Intel iPSC®/860 family

System Overview

The Paragon supercomputer is a distributed-memory multicomputer based on Intel's teraFLOPS architecture, which has been developed over the course of four generations of parallel supercomputers and is implemented using Intel's advanced microprocessor and

intel.



processor's speed is matched by on-chip cache performance and high-bandwidth memory units. The chip includes 16 KB data and instruction caches, and the bandwidth between the processor and the cache memory peaks at 1.2 GB/sec.

General-purpose (GP) nodes have a single i860 XP application processor, as well as an additional i860 XP that is dedicated to improving latency and throughput of messaging operations. The performance of the message processor is not included in the system's performance ratings.

GP nodes provide an expansion port of adding an I/O interface such as the High Performance Parallel Interface (HiPPI) or Small Computer Systems Interface (SCSI). The nodes can be configured with 16, 32, 96† or 160† MB of high speed, error-correcting dynamic RAM, accessible at 400 MB/sec. The 64-bit memory unit provides interleaved, dual-bank memory accesses.

Interconnect Architecture

Along with node performance, inter-node communications is a determining factor for sustained multicomputer performance. The Paragon system's interconnect network provides high-bandwidth, low-latency communications and frees programmers from having to concern themselves with interconnect topology. All nodes appear to be connected to all other nodes, and communication performance is uniform. Bisection bandwidth, a measure of overall system capacity, scales from 0.7 GB for a Paragon Model XP/E 8N to 5.6 GB/sec for a Paragon XP/S Model 35.

The interconnect architecture is organized to provide optimum node-to-node communications while minimizing the involvement of the application processor(s) in messaging.

Message Processor. When an application decides to send a mesage, the node's i860 XP message processor handles message-protocol processing and frees the application processor to continue with numeric computing. Messaging software is executed from the message processor's internal cache, enabling overlapped communication and application processing to occur without incurring expensive context-switching delays. The message processor is also used to implement efficient global operations such as synchronization, broadcasting and global reduction calculations (e.g., global sum).

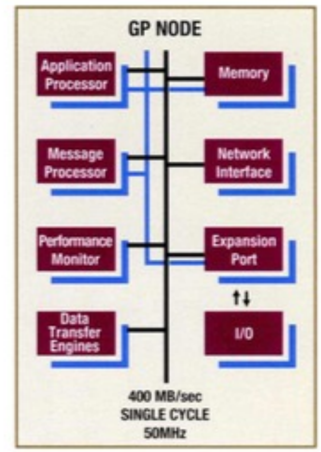
Message Routing. The actual transmission of messages is carried out by an independent routing system of custom-designed Mesh Router Controllers (MRCs), one for each node, arranged in a two-dimensional mesh. These fixed-function [illegible] the system at hardware speeds of up to 175 MB/sec full-duplex – more than enough bandwidth to directly service any peripheral device, high-speed point-to-point I/O channel, or local area network. Hardware latency, the time to set up the transfer of the first byte of a message, is so low – 40 nsec per MRC traversed – that physical location of nodes becomes unimportant for performance. Like the i860 XP processor itself, the MRCs are fabricated using Intel's triple-metal, submicron semiconductor process.

Node-Interconnect Interface. Completing the interconnect implementation is the Network Interface Controller (NIC), a custom VLSI chip that provides a full-bandwidth, pipelined interface between each node's MRC and node memory. Supported by two block transfer engines on the node, each NIC permits simultaneous inbound and outbound communication at hardware rates of 175 MB/sec and provides end-to-end error checking for each message transfer.

General-Purpose Node. Each GP node dedicates one i860 XP processor to user applications and one to message processing. The GP node's expansion port allows the addition of an I/O or networking interface.

Message Processor. When an application decides to send a mesage, the node's i860 XP message processor handles message-protocol processing and frees the application processor to continue with numeric computing. Messaging software is executed from the message processor's internal cache, enabling overlapped communication and application processing to occur without incurring expensive context-switching delays. The message processor is also used to implement efficient global operations such as synchronization, broadcasting and global reduction calculations (e.g., global sum).

Message Routing. The actual transmission of messages is carried out by an independent routing system of custom-designed Mesh Router Controllers (MRCs), one for each node, arranged in a two-dimensional mesh. These fixed-function



General-Purpose Node. Each GP node dedicates one i860 XP processor to user applications and one to message processing. The GP node's expansion port allows the addition of an I/O or networking interface.

INTEL ASCI RED (TFLOPS) COMPUTE NODE

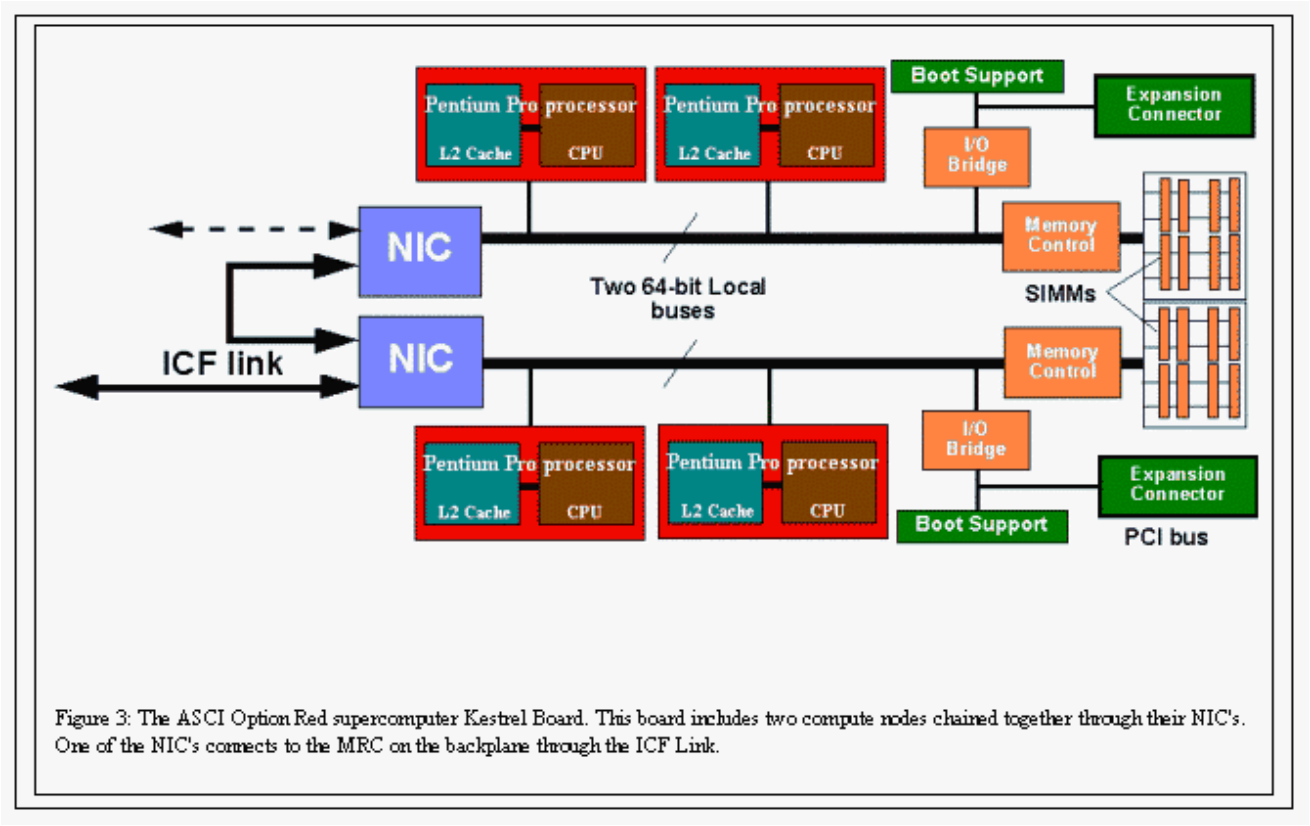


Figure 3: The ASCI Option Red supercomputer Kestrel Board. This board includes two compute nodes chained together through their NIC's. One of the NIC's connects to the MRC on the backplane through the ICF Link.

PARAGON/TFLOPS NETWORK INTERFACE CONTROLLER (NIC)

- Attached to the memory bus
- Cache coherent with the processor(s)
- Programmed by the operating system (OS)
  - Device driver was embedded in the OS
  - Driver consisted of programming DMA engines and memory-mapped registers
- Interrupt-driven
  - An interrupt would be generated for:
    - Arrival of an incoming message
    - Completion of an incoming message
    - Completion of an outgoing message
- Messages initiation via system call trap
- Source-routed, circuit-switched, wormhole routed network
  - Message header contained route to destination
  - Message body was one contiguous block

PORTALS INTERCONNECT PROGRAMMING INTERFACE

- Developed primarily by Sandia, U. New Mexico, Intel
- 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

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

BASIC ASSUMPTIONS ABOUT NETWORKING FOR HPC

- A single low-level network API is needed
  - Compute node may not have a TCP/IP stack
  - System is space-shared
    - Compute node application should own all network resources
- Applications will use multiple protocols simultaneously
  - Can't focus on just MPI
  - Runtime system, system call forwarding, I/O protocols too
- 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
- Interconnect hardware limitations can't be fixed in software

KEY NETWORK CAPABILITIES

- Independent progress
  - Data should move without requiring polling from user-level library
  - Adhere to the strong progress rule interpretation of MPI
- Overlap
  - Decouple the host processors from the network as much as possible
  - Enable overlap of computation and communication as well as communication and communication
- Scalable use of memory resources
  - Buffer space for MPI unexpected messages
  - Memory use should be independent of the number of peers
- High performance
  - Maximize bandwidth by avoiding memory-to-memory copies
  - Minimize latency by avoiding OS interaction

MPI - WEAK VERSUS STRONG PROGRESS

Rank 0

- MPI_Irecv()
- [compute for a long time]
- MPI_Wait()
- .....

- MPI_Recv()
- [compute for a long time]

Rank 1

- MPI_Send()
- [compute for a long time]

- .....

- MPI_Isend()
- [compute for a long time]
- MPI_Wait()

MORE ON STRONG VERSUS WEAK PROGRESS

- Progress was initially viewed as a quality of implementation issue
- Weak progress proponents
  - Non-blocking peer communication operations existed solely for correctness
  - Mostly an implementation issues, users are unlikely to care
- Strong progress proponents
  - Applications will structure their code based on how and when data will actually move
  - Frequency of calls into MPI will impact code structure
- MPI 2.2 Standard:
  - *Different MPI implementations reflect these different interpretations. While this ambiguity is unfortunate, it does not seem to affect many real codes. The MPI forum decided not to decide which interpretation of the standard is the correct one, since the issue is very contentious, and a decision would have much impact on implementors but less impact on users.*
- MPI 3.1 Standard:
  - *Different MPI implementations reflect these different interpretations. While this ambiguity is unfortunate, the MPI Forum decided not to de- fine which interpretation of the standard is the correct one, since the issue is contentious.*

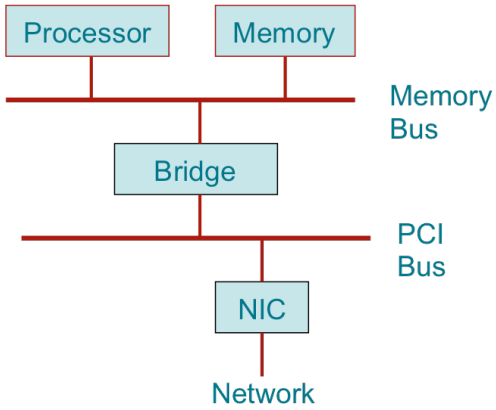
ISSUES FOR WEAK PROGRESS

- OS Jitter
  - Application process must be scheduled and be making MPI call to make progress
  - Progress thread can also be impacted
    - Can be offloaded to a dedicated core
- Non-blocking collective operations
  - Delay in one stage of collective can impact entire operation
- Composition and interoperability with other programming systems
  - Coordinate potentially multiple progress threads
  - Ideally, a single thread would handle multiple ULPs
- Creates implementation complexity for transports that have strong progress

- Impact of weak progress is pervasive

PROGRAMMABLE USER-LEVEL NETWORKS ENABLED API EXPLORATION

- Myrinet (~1994)
  - First commercially available Gb/s standalone network
  - Based on technology developed for Intel MPP networks
  - Initially available for Sun SPARC SBus, later for PCI-based PCs
  - Custom embedded MIPS-based programmable processor (LANai)
  - Myrinet Control Program (MCP) software development environment
  - Destination routed, maximum message size (packets)
  - Numerous APIs and MCPs: AM, FM, GM, PM, MX
- Quadrics QSNet (~2001)
  - Outgrowth of technology developed for Meiko MPP networks
  - Offered several different APIs for user-level networking
  - Provided a development environment for running user-level functions on NIC



FIXING SEMANTIC MISMATCH BETWEEN LAYERS

Majority of interconnect software R&D is spent on dealing with the semantic mismatch between what the upper-layer protocols need and what the low-level network software and the underlying hardware provide

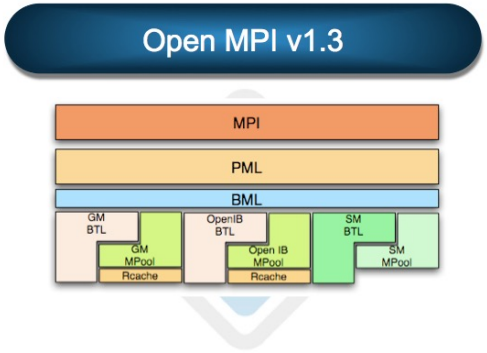
RDMA (e.g. InfiniBand Verbs)

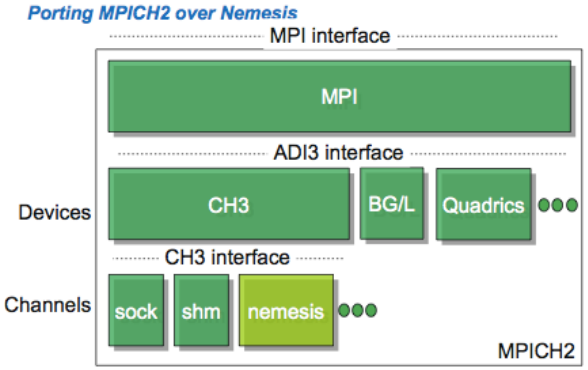
- RDMA (e.g. InfiniBand Verbs)
- One-sided
  - Allows process to read/write remote memory implicitly
- Zero-copy data transfer
  - No need for intermediate buffering in host memory
- Low CPU overhead
  - Decouples host processor from network
- Fixed memory resources
  - No unexpected Messages
- Supports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

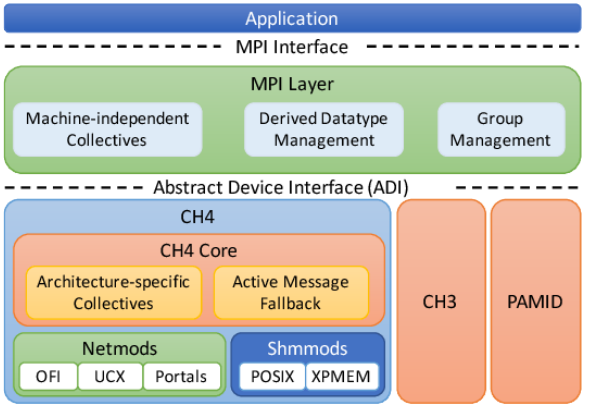
MPI Point-to-Point

- 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

NETWORK PORTABILITY ABSTRACTION LAYERS ABOUND

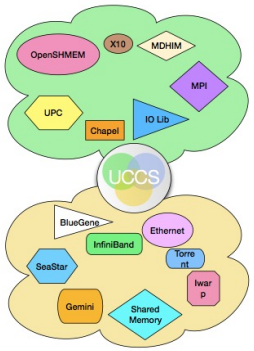






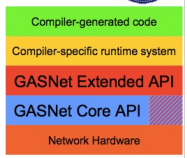
Goals

- Provide a common low-level scalable, robust, portable, simple and performance driven communication API for multiple parallel programming models over modern network interfaces
- Increasing code reusability and reducing development effort
- Include performance/power measurement capabilities in a central location



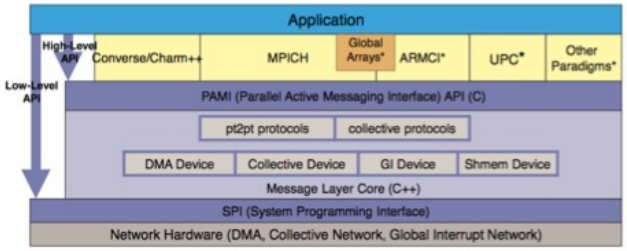
GASNet Design Overview: System Architecture

- Two-Level architecture is mechanism for portability
- GASNet Core API
  - Most basic required primitives, narrow and general
  - Implemented directly on each network
  - Based on Active Messages lightweight RPC paradigm
- GASNet Extended API
  - Wider interface that includes higher-level operations
    - puts and gets w/ flexible sync, split-phase barriers, collective operations, etc
  - Have reference implementation of the extended API in terms of the core API
  - Directly implement selected subset of interface for performance
    - leverage hardware support for higher-level operations



IBM System Technology Group

Parallel Active Message Interface



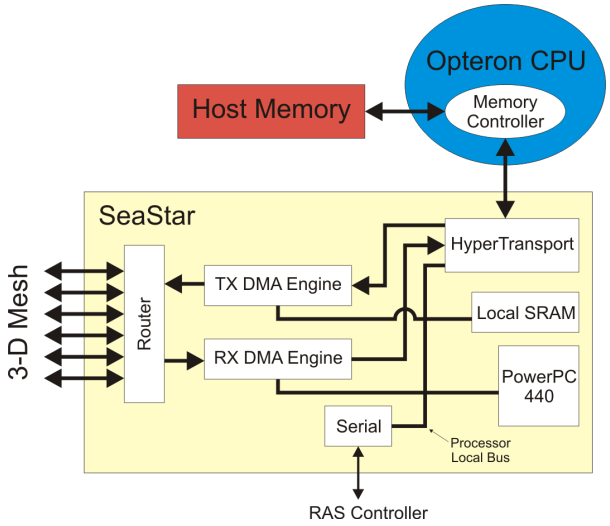
MOTIVATION FOR LOW-LEVEL TRANSPORT APIS

- Targeting a single programming model target is too limiting (top down approach)
  - MPI - MPICH, OpenMPI
  - PGAS - GasNet, OpenSHMEM
  - I/O
  - Big Data
- Desire to reduce development costs
  - Provide one network abstraction for all ULPs
  - Large porting effort is a strong indication of the semantic mismatch
- “Thin is in”
  - Optimize to a semantic mismatch
  - Get as close to the functionality you don’t really want as possible
    - Communication as well as memory management (page pinning)
- Vendor differentiation
  - Which really defeats the portability goal

2005 - CRAY SEASTAR NIC/ROUTER

Hardware

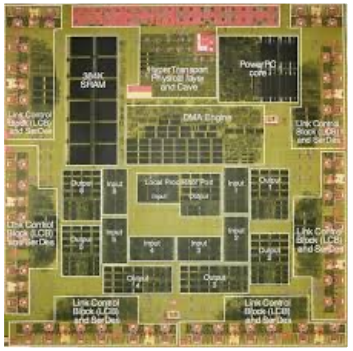
- IBM 0.13u ASIC process
- 16 1.6 Gb/s HyperTransport to Opteron
- 500 MHz embedded PowerPC 440
- 384 KB on-board scratch RAM
- Seven-port router
- Six 12-channel 3.2 Gb/s high-speed serial links



PowerPC

- Message preparation
- Message demultiplexing
  - MPI matching
  - Native IP packets
- End-to-end reliability protocol
- System monitoring

- Independent send/recv cache-coherent DMA engines between Opteron memory and network
- Message-based
  - DMA engines handle packetization
- Attempts to minimizes host overhead
- Supports reception of multiple simultaneous messages
- Delivers boot code to Opteron



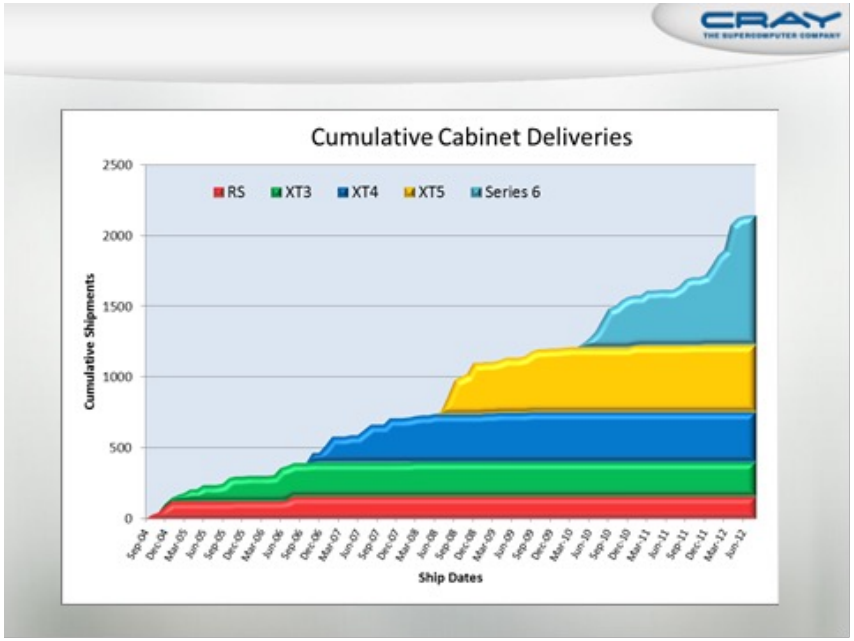
SEASTAR PERFORMANCE ISSUES

- HyperTransport IP that Cray purchased was not optimized
- Memory latency between PowerPC and SRAM memory was 2x over design
- 500 MHz PowerPC was too slow to perform simple tag matching and list processing
  - Interrupt-driven message arrival
  - Slows down message rate
- Not able to DMA from host memory
  - OS trap needed to initiate put operation
  - Slows down message rate even more
- Only 384k of local memory
- No facility for virtual-to-physical address translation

RED STORM – CRAY XT{3,4,5} IS CRAY'S BEST SELLING PLATFORM

- "Without Red Storm I wouldn't be here in front of you today. Virtually everything we do at Cray - each of our three business units - comes from Red Storm. It spawned a company around it, a historic company struggling as to where we would go next. Literally, this program saved Cray."

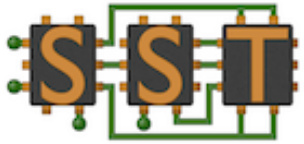
  – Pete Ungaro, Cray CEO – 2018



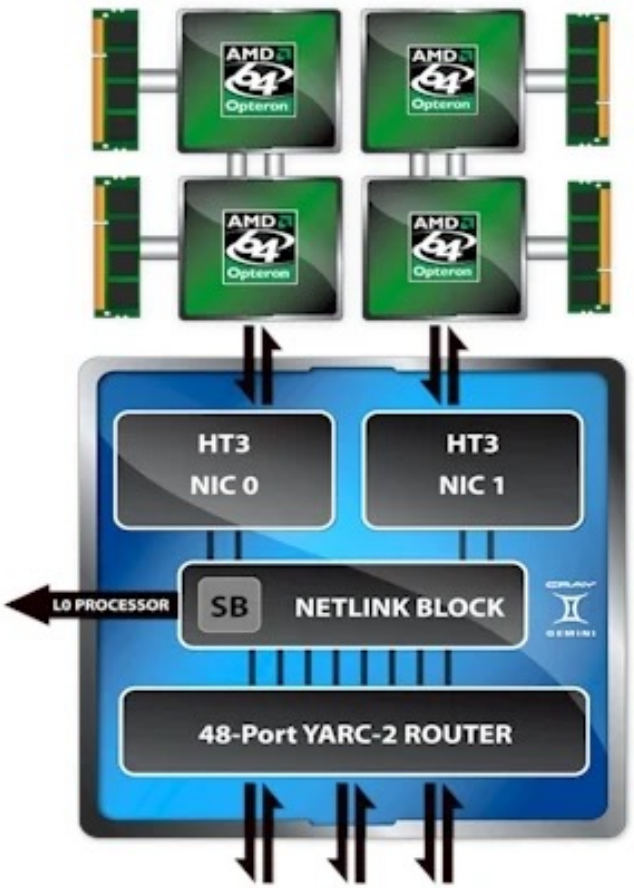
"After the Storm: The supercomputer that saved Cray," https://www.datacenterdynamics.com/en/analysis/after-the-storm-the-supercomputer-that-saved-cray/

NEXT-GENERATION SEASTAR

- Sandia lobbied Cray to build on the Seastar technology
- Fix the hardware issues
  - Improve latency and message rate
- Stay on the physical layer signaling curve
  - Keep up with InfiniBand advancements in data rate
- Developed a simulation capability to demonstrate the benefits
  - Validated the simulator against existing Seastar performance
  - Used the simulator to show the impact of hardware fixes and enhancements
- Provided a convincing argument that Seastar 3 would be ideal for HPC



2010 – CRAY GEMINI INTERCONNECT



CUSTOM NIC HARDWARE FOR HPC – IPDPS 2005

A Hardware Acceleration Unit for MPI Queue Processing

Keith D. Underwood, K. Scott Hemmert, Arun Rodrigues, Richard Murphy, and Ron Brightwell
Sandia National Laboratories*
P.O. Box 5800, MS-1110
Albuquerque, NM 87185-1110
{kdunder, kshemme, afrodri, rcmurph, rbbrigh}@sandia.gov

Abstract

With the heavy reliance of modern scientific applications upon the MPI Standard, it has become critical for the implementation of MPI to be as capable and as fast as possible. This has led some of the fastest modern networks to introduce the capability to offload aspects of MPI processing to an embedded processor on the network interface. With this important capability has come significant performance implications. Most notably, the time to process long queues of posted receives or unexpected messages is substantially longer on embedded processors. This paper presents an associative list matching structure to accelerate the processing of moderate length queues in MPI. Simulations are used to compare the performance of an embedded processor augmented with this capability to a baseline implementation. The proposed enhancement significantly reduces latency for moderate length queues while adding virtually no overhead for extremely short queues.

1. Introduction

In the mid-1990's, message passing became the dominant mechanism for programming massively parallel processor systems. By the late-1990's, the majority of message passing programs leveraged the MPI Standard [14]. In the intervening years, billions of dollars have been invested in developing application codes using MPI. Thus, it has become critically important to insure that new systems implement MPI as efficiently as possible.

Many approaches have been taken to characterizing the efficiency of MPI. The most common (and least useful) is to evaluate the ping-pong latency and bandwidth of the network. While these are necessary first order measures, models such as LogP [11] and LogGP [1] are more useful. Early work with these models [13] indicates that the most important thing for applications was to minimize the overhead (the time the application processor is involved in the communication). As a result, some of the highest performing networks have chosen to offload much of the work of sending and receiving MPI messages onto the network interfaces [2, 17, 16].

Unfortunately, the second largest impact on application performance is gap (the inverse of the message rate). Recent work [7, 3] has indicated that applications tend to traverse a significant number of entries in the two primary queues managed by MPI: the posted receive queue and the unexpected message queue. For networks that use embedded processors to traverse these queues, traversing long queues increases gap. Thus, a compromise has been made to decrease overhead while increasing gap in some scenarios.

This paper proposes a unique hardware structure to augment the microprocessor to accelerate list traversal and matching. The proposed hardware uses associative matching structures similar in concept to those in ternary content addressable memories (TCAMs) to perform high-performance parallel matching. These structures are enhanced with list management capabilities to support the unique combination of ordering semantics and high list entry turnover needed to support MPI point-to-point message passing.

To better understand basic properties of the design, a prototype has been created in FPGA hardware. The prototype provides an idea of both the clock rate that can be achieved and the timing that should be expected. It also serves as an avenue to explore and refine issues with the control interface. Unfortunately, this implementation would be difficult to integrate into a "real" environment. Thus, system-level simulation was used to demonstrate the usefulness of the proposed hardware. An MPI implementation was created that leverages the hardware acceleration unit. Using simulation, this MPI implementation was compared to a baseline

* Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)
1530-2075/05 $ 20.00 IEEE

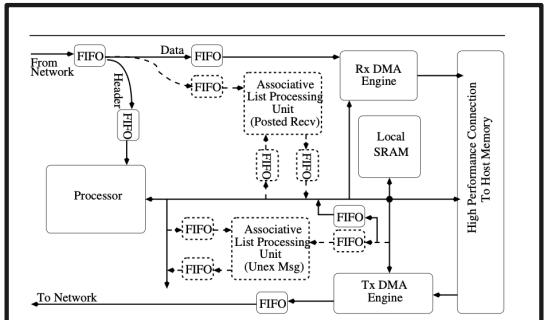


Figure 1. The proposed NIC architecture (new features shown in dashed lines)

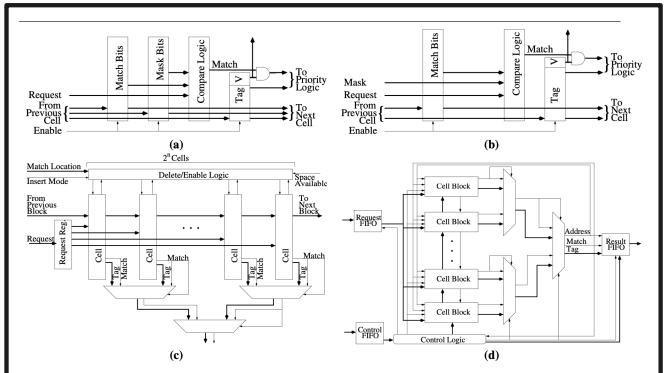


Figure 2. (a) A cell with a single match unit for the posted receive queue; (b) The corresponding cell a for the unexpected message queue; (c) A block of cells; (d) The associative match engine

Parameter	CPU	NIC Processor
Fetch Q	4	2
Issue Width	8	4
Commit Width	4	4
RUU Size	64	16
Integer Units	4	2
Memory Ports	3	1
L1 Caches	64K 2-way	32K 64-way
L2 Cache	512K	none
Clock Speed	2Ghz	500Mhz
Lat. To Main Memory	80-85ns	115-120ns
ISA	PowerPC	PowerPC
Network Wire Lat.	200 ns	

Table 2. Processor Simulation Parameters

measuring latency in that it includes the time to post the receive for the latency measuring message as part of the latency. This better reflects the way that MPI is actually used by applications, which typically have some number of iterations and post receives in each iteration.

5.2. Simulation Environment

System-level simulation of the matching structure used a simulator based on Enkidu [19], a component-based discrete event simulation framework. To simulate the CPU and NIC processors, `sim-outorder` from the SimpleScalar [10] tool suite was integrated into this framework. Components representing a simple network, DMA engines, a memory controller, and DRAM chips were added. The memory hierarchy was modeled to include contention for open rows on the DRAM chips.

The main processor was parameterized to be similar to a modern high-performance processor. The NIC processor was parameterized to be similar to a processor in higher end network cards, such as the PowerPC 440 (see table 2). A simple bus on the NIC connected the main processor with the DMA engine, SRAM, and matching structure. This bus was simulated with a 20ns delay. The SRAM was modeled with a 3ns delay.

5.4. FPGA Prototype

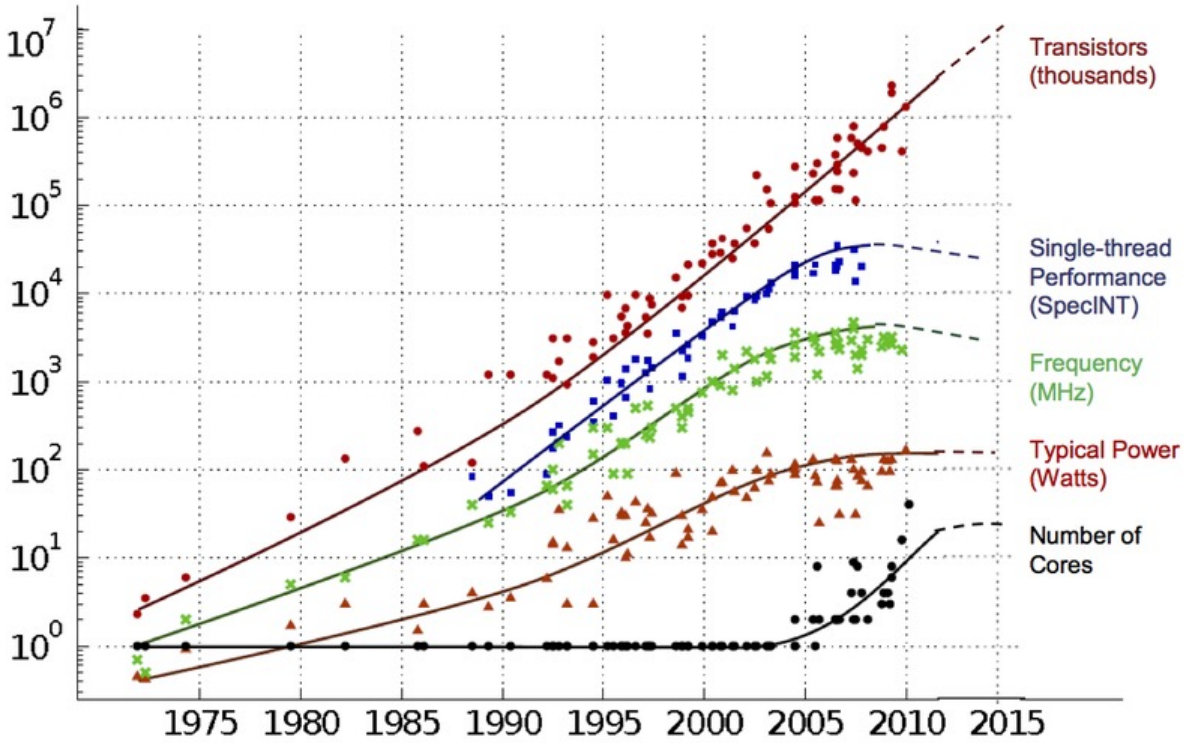
To provide a reasonable estimate of the size and operating frequency of the ALPU, a prototype implementation was created, targeting Xilinx Virtex 2 and Virtex 2 Pro FPGAs. The ALPU was designed using JHDL [12], a structural design tool that provides fine-grained control over the placement of logic on the FPGA. The final design is parameterized to allow different match and tag widths, as well as different combinations of the total number of cells and the number of cells in each block.

When designing the unit, the top priorities were small area, high speed and regularity in placement. To allow for higher operating frequencies, the ALPU requires multiple clock cycles to complete a match (6 or 7 depending on the size of the ALPU and the blocking factor). If desired, it is possible to overlap execution of the first and last cycles. The simulation results assume a 7 cycle pipelining latency with

HOST ONLOAD VERSUS NIC OFFLOAD ARGUMENT

- Why design a custom NIC for offload?
- Just dedicate a core
  - A 3 GHz Xeon will outperform a 500+ MHz embedded processor on network protocol processing
  - A custom ASIC is way too expensive (especially for the small HPC market)
- Cost will go down as core count increases
- Cores won't be getting slower, right?

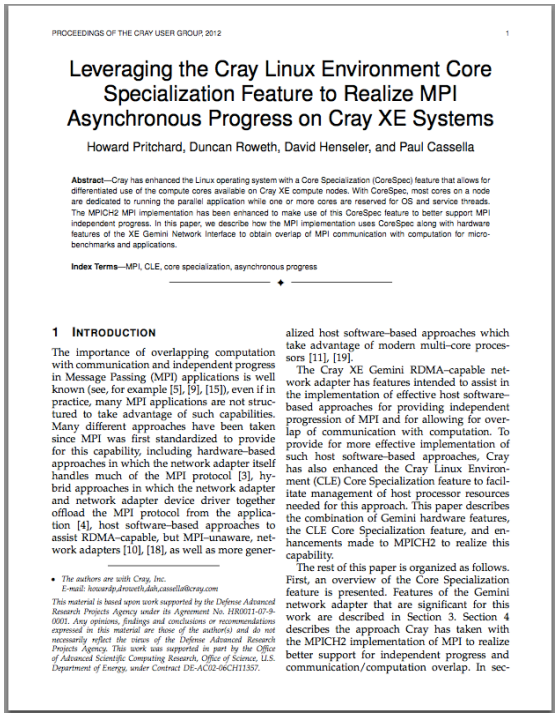
CPU CORE CLOCK FREQUENCY STALLED IN ~2007



Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten
Dotted line extrapolations by C. Moore

CRAY CORE SPECIALIZATION

- Dedicate “OS” cores to handle MPI progress
  - MPI progress threads run on a dedicated set of cores



PROCEEDINGS OF THE CRAY USER GROUP, 2012 1

Leveraging the Cray Linux Environment Core Specialization Feature to Realize MPI Asynchronous Progress on Cray XE Systems

Howard Pritchard, Duncan Roweth, David Henseler, and Paul Cassella

Abstract—Cray has enhanced the Linux operating system with a Core Specialization (CoreSpec) feature that allows for differentiated use of the compute cores available on Cray XE compute nodes. With CoreSpec, most cores on a node are dedicated to running the parallel application while one or more cores are reserved for OS and service threads. The MPICH2 MPI implementation has been enhanced to make use of this CoreSpec feature to better support MPI independent progress. In this paper, we describe how the MPI implementation uses CoreSpec along with hardware features of the XE Gemini Network interface to obtain overlap of MPI communication with computation for micro-benchmarks and applications.

Index Terms—MPI, CLE, core specialization, asynchronous progress

1 Introduction

The importance of overlapping computation with communication and independent progress in Message Passing (MPI) applications is well known (see, for example [5], [9], [15]), even if in practice, many MPI applications are not structured to take advantage of such capabilities. Many different approaches have been taken since MPI was first standardized to provide for this capability, including hardware–based approaches in which the network adapter itself handles much of the MPI protocol [3], hybrid approaches in which the network adapter and network adapter device driver together offload the MPI protocol from the application [4], host software–based approaches to assist RDMA–capable, but MPI–unaware, network adapters [10], [18], as well as more generalized host software–based approaches which take advantage of modern multi–core processors [11], [19].

The Cray XE Gemini RDMA–capable network adapter has features intended to assist in the implementation of effective host software–based approaches for providing independent progression of MPI and for allowing for overlap of communication with computation. To provide for more effective implementation of such host software–based approaches, Cray has also enhanced the Cray Linux Environment (CLE) Core Specialization feature to facilitate management of host processor resources needed for this approach. This paper describes the combination of Gemini hardware features, the CLE Core Specialization feature, and enhancements made to MPICH2 to realize this capability.

The rest of this paper is organized as follows. First, an overview of the Core Specialization feature is presented. Features of the Gemini network adapter that are significant for this work are described in Section 3. Section 4 describes the approach Cray has taken with the MPICH2 implementation of MPI to realize better support for independent progress and communication/computation overlap. In sec-

• The authors are with Cray, Inc.
E-mail: howardp,droweth,dah,cassella@cray.com

This material is based upon work supported by the Defense Advanced Research Projects Agency under its Agreement No. HR0011-07-9-0001. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency. This work was supported in part by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357.

S3D Time Step Summary

# Application Threads	Progression disabled	Progression enabled
14	4.77	3.93
15	4.68	4.05
16	4.59	4.06

MILC Run Time Summary(secs)

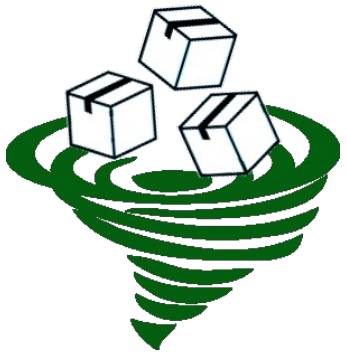
# Run Type	4096 ranks	8192 ranks
No progression	2165	1168
Progression (phase 1)	2121	1072
Progression (phase 2)	3782	2138
Progression (phase 1) no reserved cores	3560	2210
Progression (phase 1) reserve core but no corespec	2930	2070

2017 – MORE NIC HARDWARE DESIGN

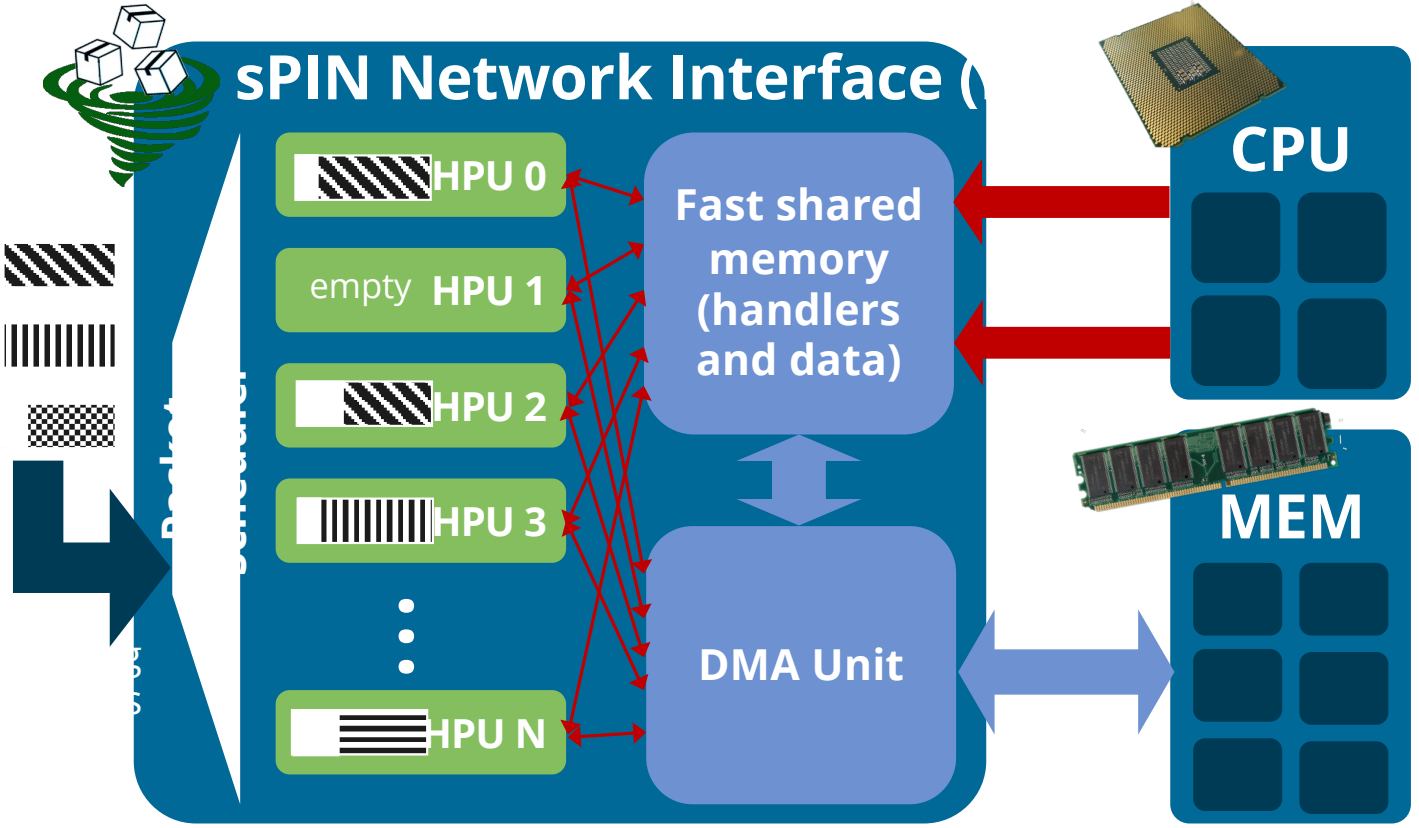
- Cores are slower, more energy-efficient
  - Modern cores require 10-20 ns to access L3 cache
- Terabit per second networks are coming
  - 400 Gib/s can deliver a 64-byte message every 1.2 ns
- Need to remove processor from network processing path (offload)
- RDMA only supports data transfer between virtual memory spaces
  - Data is placed blindly into memory
  - Need varying levels of steering the data at the target

STREAM PROCESSING IN THE NETWORK (SPIN)

- Augment RDMA and matching tasks with simple processing tasks
- Programming interface for specifying packet processing kernels
  - Similar to the way CUDA and OpenCL work for compute accelerators
- Tasks that can be performed on incoming packets
  - Limited state
  - Small fast memory on NIC
  - Execute short user-defined functions
- Handlers are compiled for the NIC and passed down at initialization time
- Vendor-independent interface would enable strong collaborative open-source environment similar to MPI



SPIN ARCHITECTURE OVERVIEW



SPIN APPROACH

- Handlers are executed on NIC Handler Processing Units (HPUs)
- Simple runtime manages HPUs
- Each handler owns shared memory that is persistent for the lifetime of a message
  - Handlers can use this memory to keep state and communicate
- NIC identifies all packets belonging to the same message
- Three handler types
  - Header handler – first packet in a message
  - Payload handler – all subsequent packets
  - Completion handler – after all payload handlers complete
- HPU memory is managed by the host OS
- Host compiles and offloads handler code to the HPU
- Handler code is only a few hundred instructions

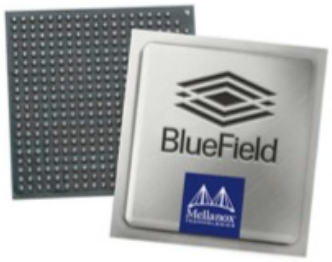
SPIN APPROACH (CONT'D)

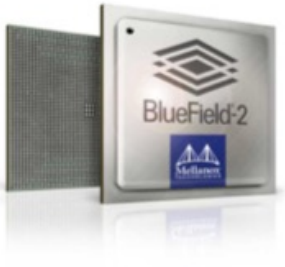
- Handlers are written in standard C/C++ code
- No system calls or complex libraries
- Handlers are compiled to the specific Network ISA
- Handler resources are accounted for on a per-application basis
  - Handlers that run too long may stall NIC or drop packets
- Programmers need to ensure handlers run at line rate
- Handlers can start executing within a cycle after packet arrival
  - Assuming an HPU is available
- Handlers execute in a sandbox relative to host memory
  - They can only access application's virtual address space
  - Access to host memory is via DMA

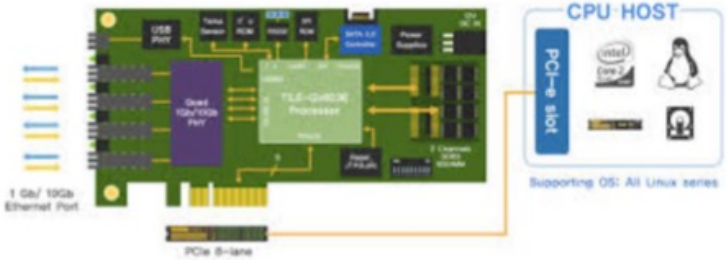
1997 – 2017: PROGRAMMABLE NICS ARE NOT VIABLE

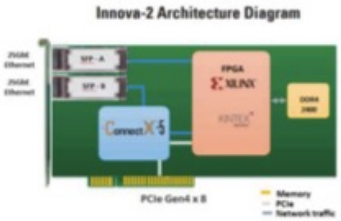
- Programmable network interface controllers (NICs) that support offload for HPC aren't commercially viable
  - Custom NIC design is too expensive
  - Market/volume is too small to justify development cost
  - 2004 – Quadrics QsNet III dies
  - 2013 – Myricom finally goes out of business
- Offloading for improving network performance doesn't make sense
  - TCP offload engines never quite worked
  - Use dedicated host cores for network protocol processing
  - 3 GHz Xeon will outperform any embedded NIC processor
  - Overhead will go down as host core count increases

2019: SMARTNICS TAKE OVER THE WORLD









CN2350-210SVPN

CN7890-240NV

SMARTNICS/DPUS

- Designed to offload network services to support virtualization
  - Application computation offload
  - Other services, such as I/O
- Lots of papers on offloading applications and services operations to SmartNICs
- What else are they good for?
  - Edge computing?
  - AI/ML?
- Don't seem to address MPI/GPU performance challenges
  - Data movement and synchronization from accelerator hardware
- What's the programming model?
  - Packet-based rather than message based
  - Deposit entire message into NIC memory for inspection
  - Not really Active Message semantics HPC may want
  - No help for GPU communication integration
  - Still need integrated NICs?

Panel:
Is Exascale End-of-the-Line for Commodity Networks?

Ron Brightwell
Scalable System Software
Sandia National Laboratories
Albuquerque, NM, USA

IEEE Conference on Cluster Computing

September 27, 2011

Sandia is a Multiprogram Laboratory Operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy Under Contract DE-ACO4-94AL85000.

NNSA

Sandia National Laboratories

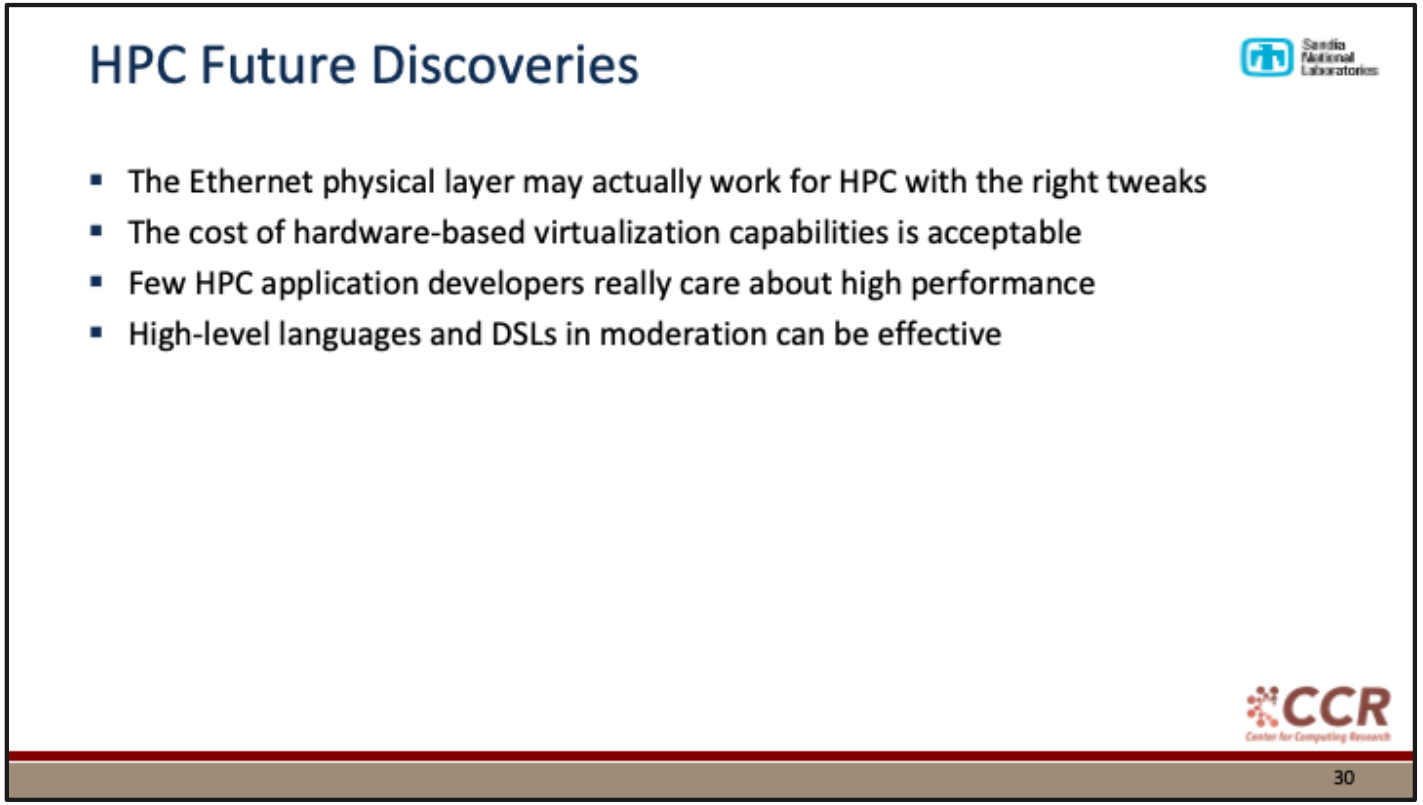
Answers to Panel Questions

- Are commodity networks a lost cause?
  - Yes. A balanced exascale system will require a non-commodity solution.
- Will the constraints of Exascale systems require proprietary solutions?
  - Exascale power, performance and resiliency requirements require tighter integration between network interface, cores, and memory
- Can commodity networks sustain at/beyond Exascale?
  - Maybe. It depends on how much imbalance is tolerable.
- Will they be able to meet the stringent Exascale requirements?
  - No.
- Will the non-HPC market ever require commoditization of proprietary Exascale network technologies?
  - Eventually, yes.
- Time prediction - Will commodity networks eventually catch up with proprietary solutions (many years later) or will they be at the leading forefront (that is, one of the first 5 Exascale machines)?
  - The technology push needed for exascale requires non-commodity solutions, but these technologies will accelerate commodity solutions for general-purpose computing.

NNSA

Sandia National Laboratories

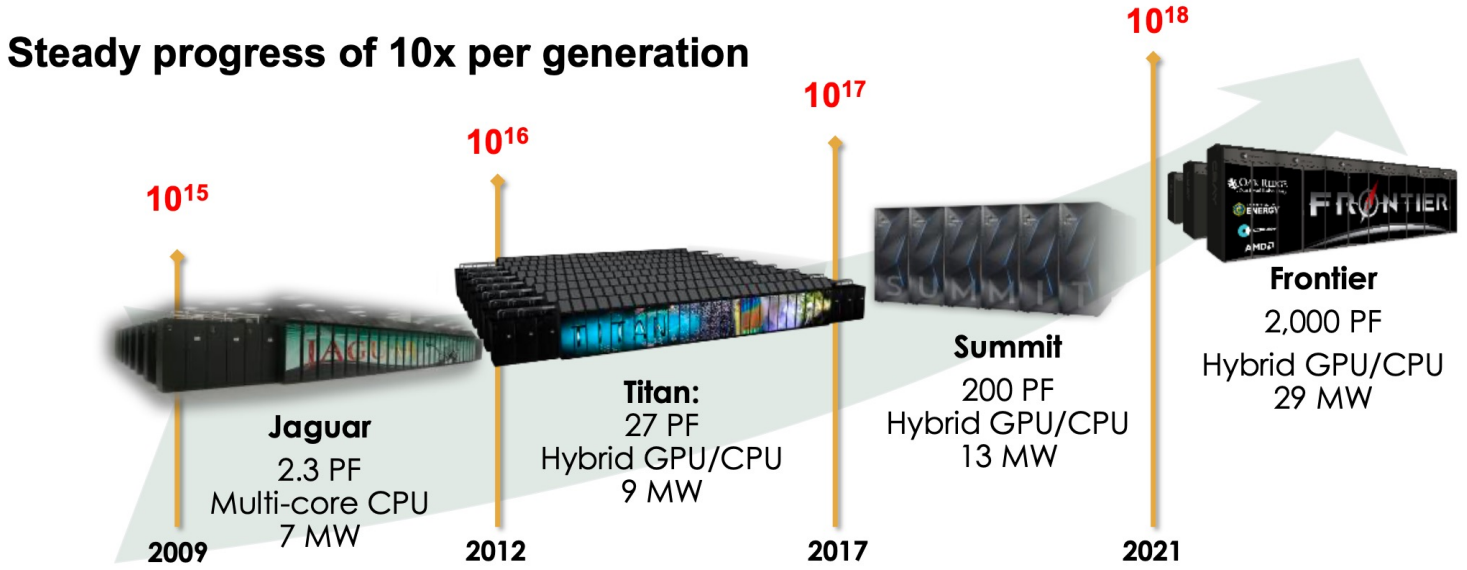
EUROMPI 2019 KEYNOTE



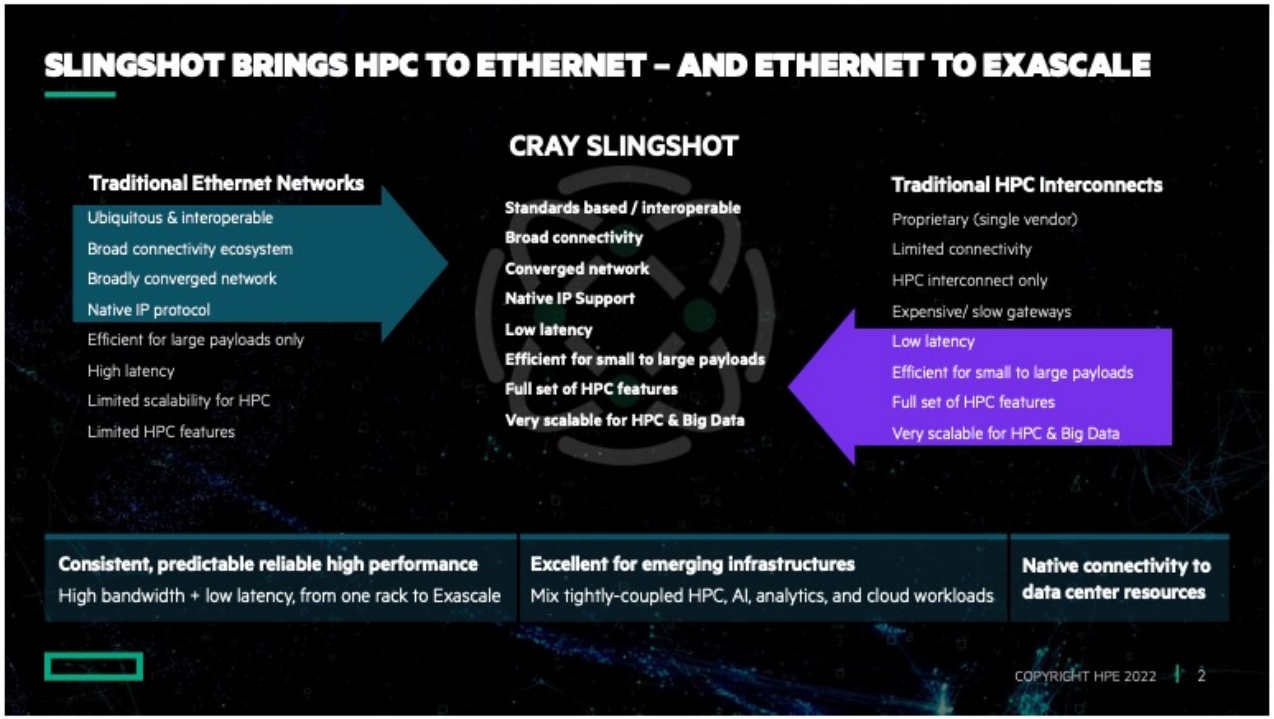
EXASCALE IS HERE

Oak Ridge National Laboratory's Journey from Petascale to Exascale

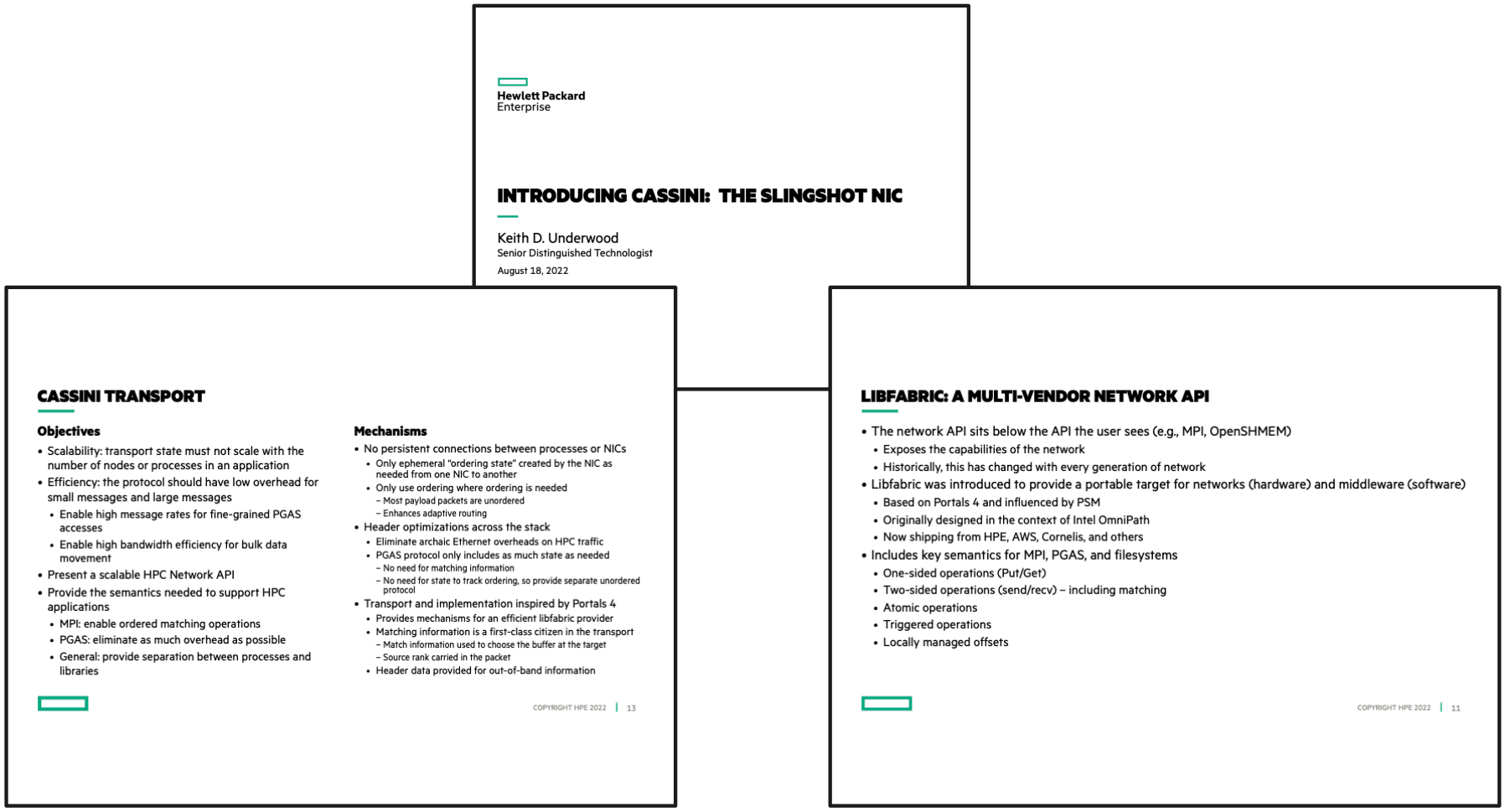
Mission: Providing world-class computational resources and specialized services for the most computationally intensive global challenges	Vision: Deliver transforming discoveries in energy technologies, materials, biology, environment, health, etc.



EXASCALE USES ETHERNET



PORTALS INFLUENCE ON SLINGSHOT



Hewlett Packard
Enterprise

INTRODUCING CASSINI: THE SLINGSHOT NIC

Keith D. Underwood
Senior Distinguished Technologist
August 18, 2022

CASSINI TRANSPORT

Objectives

- Scalability: transport state must not scale with the number of nodes or processes in an application
- Efficiency: the protocol should have low overhead for small messages and large messages
  - Enable high message rates for fine-grained PGAS accesses
  - Enable high bandwidth efficiency for bulk data movement
- Present a scalable HPC Network API
- Provide the semantics needed to support HPC applications
  - MPI: enable ordered matching operations
  - PGAS: eliminate as much overhead as possible
  - General: provide separation between processes and libraries

Mechanisms

- No persistent connections between processes or NICs
  - Only ephemeral "ordering state" created by the NIC as needed from one NIC to another
  - Only use ordering where ordering is needed
    - Most payload packets are unordered
    - Enhances adaptive routing
- Header optimizations across the stack
  - Eliminate archaic Ethernet overheads on HPC traffic
  - PGAS protocol only includes as much state as needed
    - No need for matching information
    - No need for state to track ordering, so provide separate unordered protocol
- Transport and implementation inspired by Portals 4
  - Provides mechanisms for an efficient libfabric provider
  - Matching information is a first-class citizen in the transport
    - Match information used to choose the buffer at the target
    - Source rank carried in the packet
  - Header data provided for out-of-band information

COPYRIGHT HPE 2022 | 13

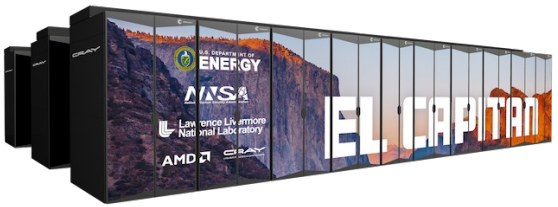
LIBFABRIC: A MULTI-VENDOR NETWORK API

- The network API sits below the API the user sees (e.g., MPI, OpenSHMEM)
  - Exposes the capabilities of the network
  - Historically, this has changed with every generation of network
- Libfabric was introduced to provide a portable target for networks (hardware) and middleware (software)
  - Based on Portals 4 and influenced by PSM
  - Originally designed in the context of Intel OmniPath
  - Now shipping from HPE, AWS, Cornelis, and others
- Includes key semantics for MPI, PGAS, and filesystems
  - One-sided operations (Put/Get)
  - Two-sided operations (send/recv) – including matching
  - Atomic operations
  - Triggered operations
  - Locally managed offsets

COPYRIGHT HPE 2022 | 11

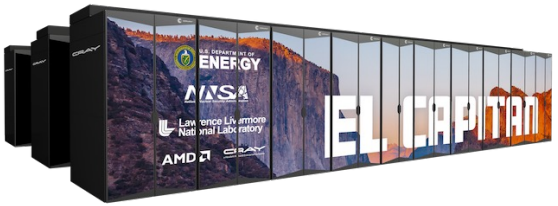
“SANDIA INSIDE” ALL THREE DOE EXASCALE SYSTEMS

- “**Portals 4 provided inspiration for the Cassini line of NICs in the HPE Cray Slingshot** network, the interconnect enabling the first Exascale system on the Top500 list. The network transport between Cassini NICs draws on Portals 4 for how to provide buffer addressing, buffer authentication, and flow control. For the Cassini NIC architecture itself, looking at Portals 4 provided an initial guideline for what should be a feature of software and what features the NIC should incorporate. […] **the visionary work done by the Portals team significantly accelerated the architecture definition phase of Cassini**. As Cray and Intel looked for a network API capable of meeting the demands of Exascale, **Portals 4 became the inspiration for the OFIWG libfabric API**, which exploits many of its concepts. Libfabric ships with the HPE Cray Slingshot network today and is gaining industry traction, notably through its use in the AWS EFA NIC.”
  - Mike Vildibill, Vice President and General Manager, HPE
- “[…] the **Portals 4 specification was a key technology that informed and guided the evolution of Intel's OmniPath Architecture**. Further, Portals 4 was very influential in the design of the network interface hardware and the semantics of the Open Fabrics Interface API (OFI). Portals 4 was a critical proof point that the industry could develop tailored networking technology while maintaining a vendor-neutral software interface, and it is arguably **the most successful hardware/software co-design tool for high performance networking** I encountered during my years of technical leadership.”
  - Bill Magro, Chief Technologist - High-Performance Computing, Google Cloud



“SANDIA INSIDE” ALL THREE DOE EXASCALE SYSTEMS

- “Cray had always performed extensive system scale simulation using home grown tools. As the Slingshot project started, the Cray team decide to adopt an open-source simulator. We selected SST for two main reasons. Firstly, we knew that we would need to scale to large system sizes, thousands of switches with tens of thousands of endpoints or NICs. Secondly, the DOE Labs community was developing motifs and miniApps that characterized their applications and integrating them into SST. The Cray team worked with developers at Sandia introducing a device interface that allowed for interchange of open source and proprietary device models. Cray implemented models of the Rosetta switch and Cassini NIC that used this interface. The resulting devices are being used in all three of the US exascale systems. **We regard the work on SST that Cray undertook with Sandia as an excellent example of vendor-labs codesign, one that we will build upon in future projects**.”
- Duncan Roweth, HPE Senior Distinguished Technologist



SIGNIFICANT VENDOR IMPACT OF SANDIA'S PORTALS NETWORKING TECHNOLOGY

All of these production vendor-supported systems used Portals as the network hardware programming interface.

Portals 0

Portals 2

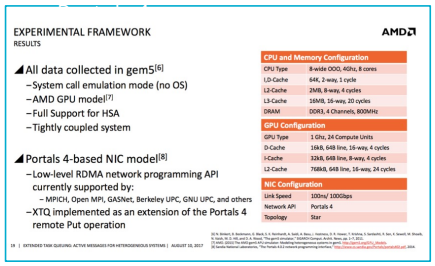
Portals 3

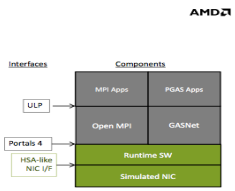
Portals 3

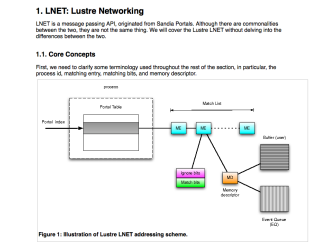
Portals 4

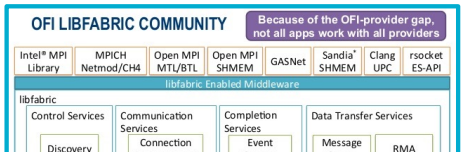
portals

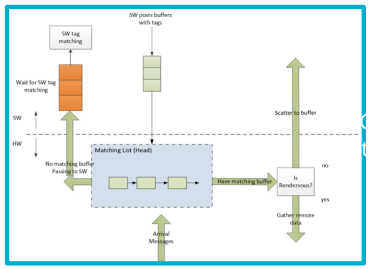
Unlike other low-level network programming interfaces, Portals is intended to enable co-design rather than serve as a portability layer.

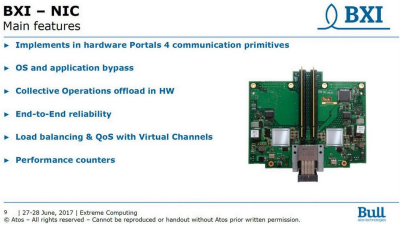




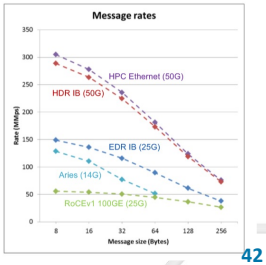








- Slingshot speaks standard Ethernet at the edge, and optimized HPC Ethernet on internal links
- Reduced minimum frame size
  - Remove Ethernet's 64B minimum frame size
  - Target a 40B frame rate but allow 32B frames + sideband
- Removed inter-packet gap
- Optimized header
  - Reduced preamble
  - IPv4 and IPv6 packets can be sent without an L2 header
  - Portals uses modified IPv4 header without an L2 header
- Credit-based flow control
- Protocol also provides resiliency benefits
  - Low-latency FEC (see 25Gbit Ethernet Consortium)
  - Link level retry to tolerate transient errors
  - Lane degrade to tolerate hard failures



ULTRA ETHERNET

Ultra Ethernet Consortium | Jul 19, 2023

Leading Cloud Service, Semiconductor, and System Providers Unite to Form Ultra Ethernet Consortium

UEC to deliver on Ethernet-based open, interoperable, high-performance full-communications stack architecture to meet the growing network demands of AI & HPC at scale. SAN FRANCISCO – July 19, 2023 – Announced today, Ultra Ethernet Consortium (UEC) is bringing together leading companies for industry-wide cooperation to build a complete Ethernet-based communication stack architecture for...

- Working groups
  - Physical layer
  - Link layer
  - Transport layer
  - Software
  - Storage
  - Compliance
  - Management
  - Performance and debug

VENDOR PARTICIPATION

WHAT DOES THIS MEAN FOR HPC?

- Potentially large number of network vendors providing HPC-capable network hardware
  - Significantly expands current HPC network providers
    - HPE, Nvidia/Mellanox, Cornelis, BXI
  - Potentially limits the number of future HPC network providers
    - HPE
- Multiple potentially competing goals
  - HPC, data center, and AI/ML
  - Opportunity for tighter integration of AI/ML accelerator hardware?
- May impact customization/specialization capabilities
  - Not clear where vendor differentiation will occur
    - Network programming API has been the traditional differentiator

FUTURE CHALLENGES AND POTENTIAL SOLUTIONS

- Most pressing HPC interconnect challenges
  - Latency
  - Parallelism in the network interface
  - Resiliency
  - Flow control
  - Active message semantics
  - New types of communication – runtime, workflows, analytics
  - Event-driven capability – for resilience and task-based programming models
- What hardware mechanisms can help?
  - Integrated NIC
  - More network contexts
  - Virtual addressing of endpoints
  - Flow control protocols that understand endpoint resources
  - Mechanisms for network isolation
- What else can help?
  - Instrumentation to help understand application resource usage (security implications)
  - Canonical complex workflows/workloads
  - Benchmarks that target specific desired capabilities

ACKNOWLEDGMENTS

- Several people who contributed to Portals over the years
  - Brian Barrett
  - Ryan Grant
  - Torsten Hoefler
  - Barney Maccabe
  - Kevin Pedretti
  - Rolf Riesen
  - Keith Underwood
- People from whom I stole slides
  - Al Geist
  - Keith Underwood