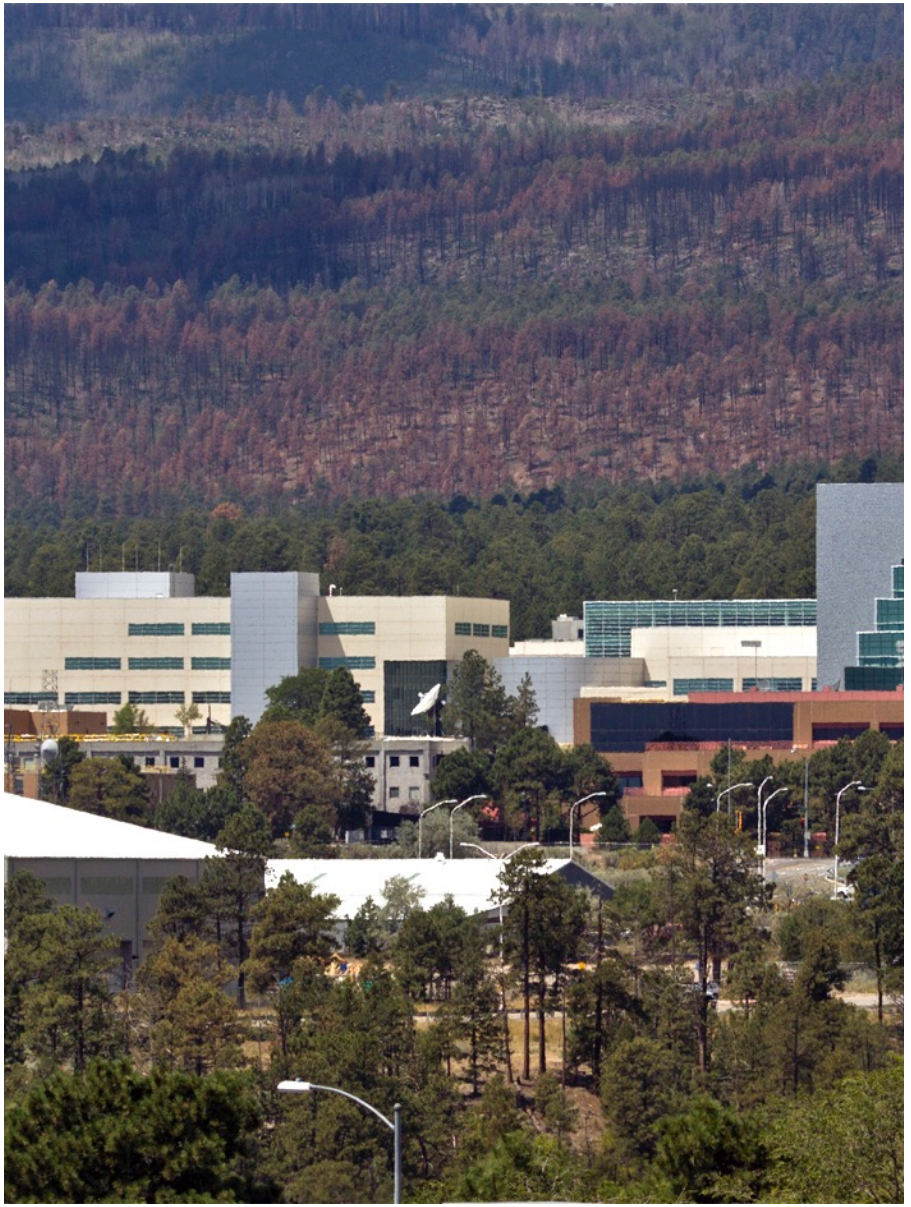
libhio: Optimizing IO on Cray XC Systems With DataWarp

May 9, 2017

Nathan Hjelm

Cray Users Group
May 9, 2017

Los Alamos
NATIONAL LABORATORY
EST. 1943



Outline

- Background
- HIO Design
- Functionality
- Performance
- Summary

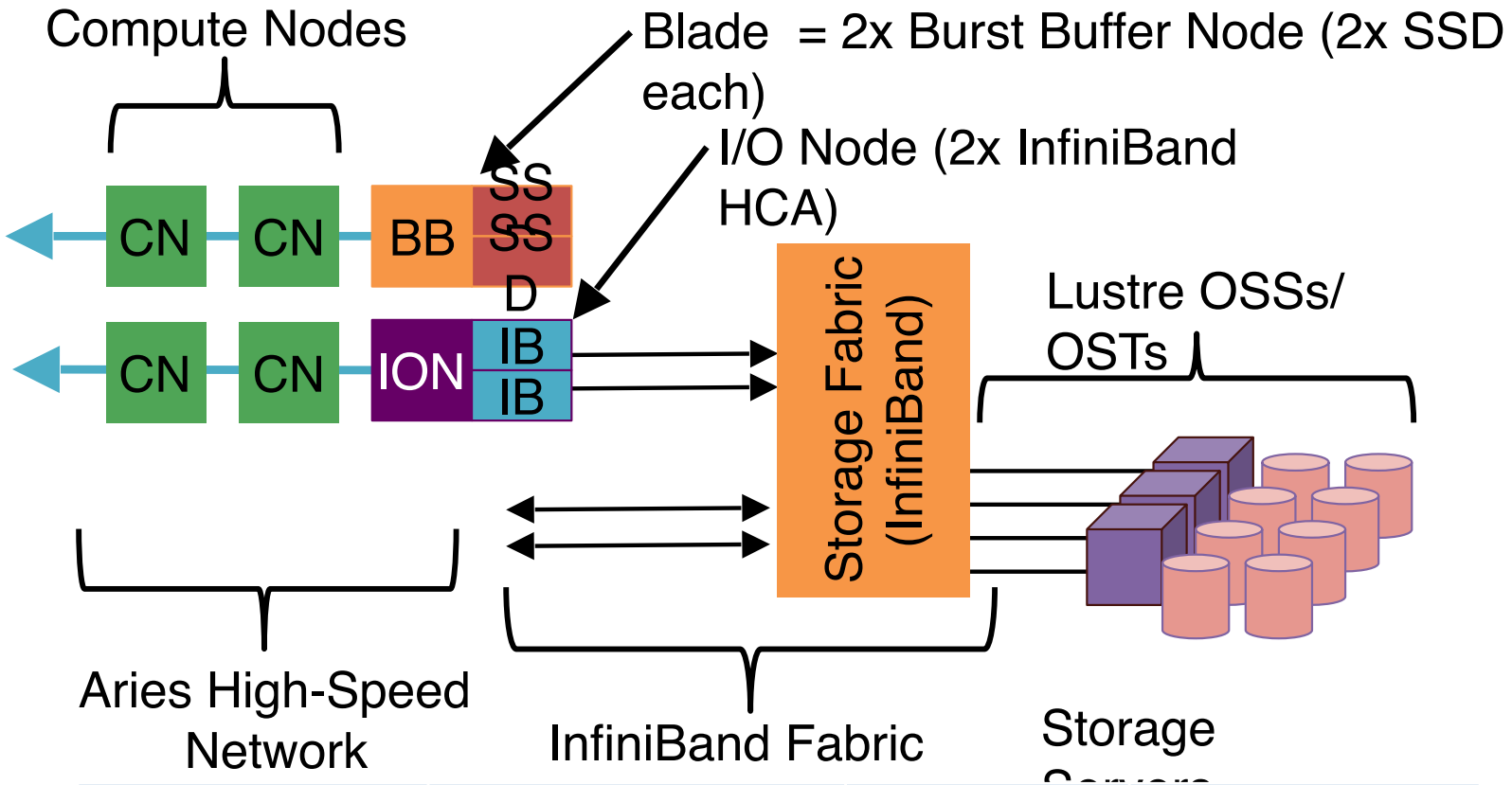
Background: Terminology

- **Burst Buffer**
  - A high-speed, low cost-per-bandwidth storage facility used to reduce the time spent on high volume IO thus improving system efficiency.
- **Cray DataWarp™**
  - A Cray SSD storage product on Trinity. It provides burst buffer (and other) function. Initially developed for Trinity and Cori.
- **Hierarchical IO Library (HIO)**
  - A LANL developed API and library which facilitates the use of burst buffer and PFS (Parallel File System) for checkpoint and analysis IO on Trinity and future systems.

Background: What Is Datawarp?

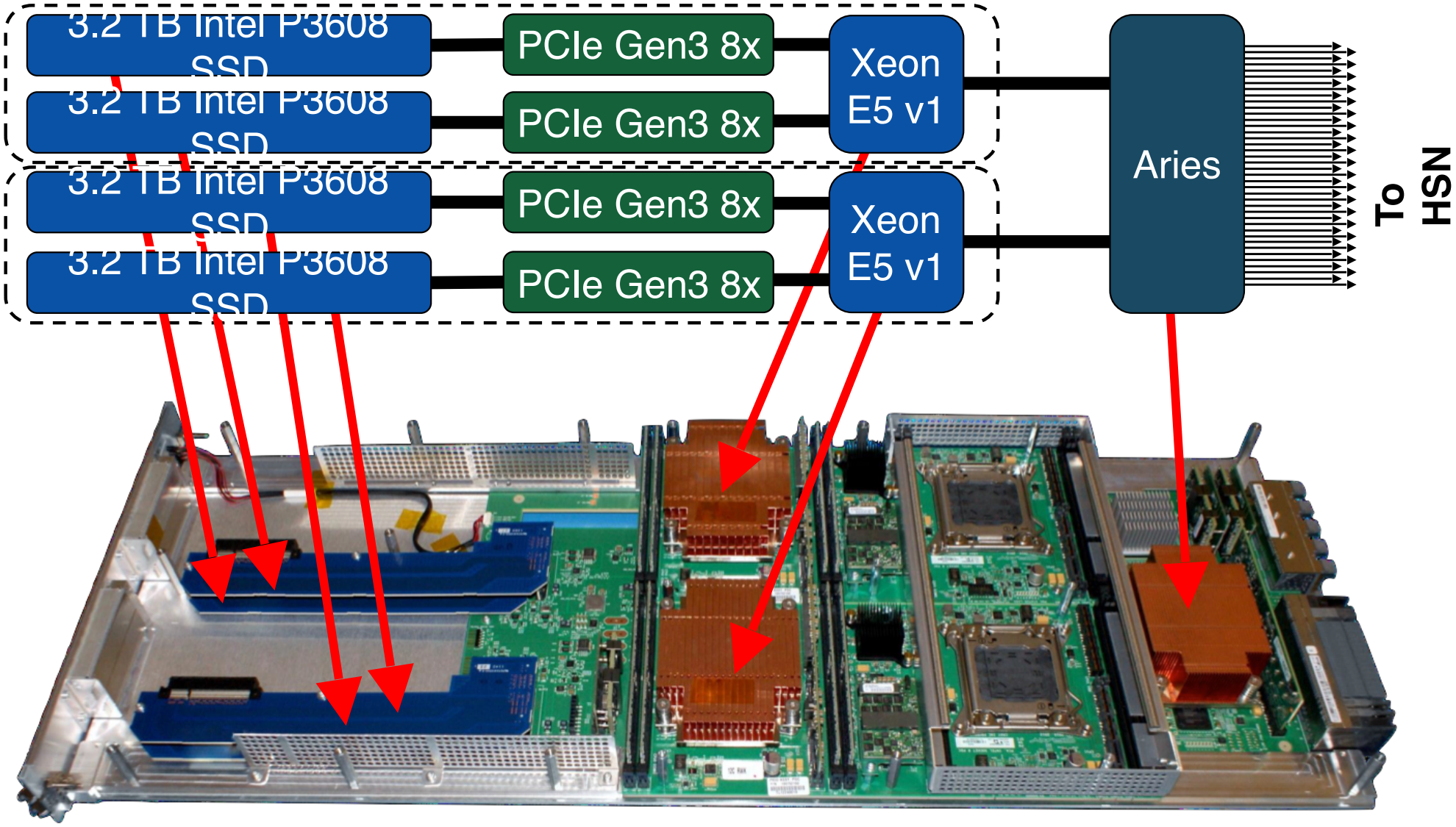
- A Cray SSD storage product developed for Trinity
- Consists of:
  - **Hardware**: service nodes connected directly to Aries network each containing two SSDs.
  - **Software**: API/library with functions to initiate stage in/ stage out and query stage state.
  - Can be configured in multiple modes using the workload manger.
    - Striped, cache, private, etc

Background: Trinity Burst Buffer Architecture



Trinity Phase	Compute Nodes	DataWarp Nodes	DataWarp Capacity
1	9500 Haswell 1000 KNL	300	1.7 PB
2	8900 KNL added	232	1.3 PB

Background: Trinity Burst Buffer Architecture



Background: Trinity Burst Buffer Architecture

Mode	Description
Private Scratch	Per node burst buffer (BB) space
Shared Scratch	Shared space, files may be striped across all BB nodes. → Used for Trinity Checkpoints ←
Shared Cache	Parallel File System (PFS) cache. Transparent and explicit options
Load Balanced Read Only Cache	PFS files replicated into multiple BB nodes to speed up widely read files

What is HIO? Design Motivation

- **Isolate applications from BB and IO technology evolution**
- Support Trinity and future burst buffer implementations
- Support Lustre and other PFS on all tri-lab systems
- Easy to incorporate into existing applications
  - Lightweight, simple to configure
- Improve checkpoint performance
  - Support N-1, N-N, N-M (and more) IO patterns
- No collective read or write calls
- Support checkpoint/restart and analysis scenarios (e.g., viz)
- Extend to bridge to future IO technologies
  - e.g., Sierra, HDF5, Object Store ?
- Provide High Performance I/0 best practices to many applications

What is HIO? API Motivation

- Codes stored multiple types of data in a restart dump
  - Provide named elements in a file
- Codes handle fallback on restart failure (corruption, missing files)
  - Provide a numeric identifier for files.
  - Provide highest identifier and last modified identifiers for open
- Need for performance metrics and tracing
- Need to handle staging between IO layers
  - Collective open/close to determine when datasets are complete

What is HIO? Structure and Function

- HIO is packaged as an independent library
- Plug-in architecture to support future storage systems
- Flexible API and configuration functionality
- Configurable diagnostic and performance reporting
- Supports Cray DataWarp Burst Buffer on Trinity
  - Staging to PFS, space management, striping
  - Automatically handles staging of data.
  - Automatic handling of space.
- Supports other PFS on Trinity
  - Lustre specific support (interrogation, striping)
  - Add others (GPFS on Sierra) as needed

HIO Concepts

- HIO is centered around an abstract name space
- An HIO “file” is known as a “dataset”
- An HIO dataset supports
  - Named binary elements
  - Shared or unique-per-rank element offset space
    - Offset space is per-named element
    - This provides traditional N-1 and N-N functionality
  - Version or level identifier
  - A notion of completeness or correctness
- POSIX on-disk structure currently is a directory and its contents
- Internal on-disk structure is not specified – to preserve flexibility for future optimization
- Backward compatibility across HIO versions
  - Read any prior dataset formats
  - Support any prior API definitions

HIO IO Modes: Basic Mode

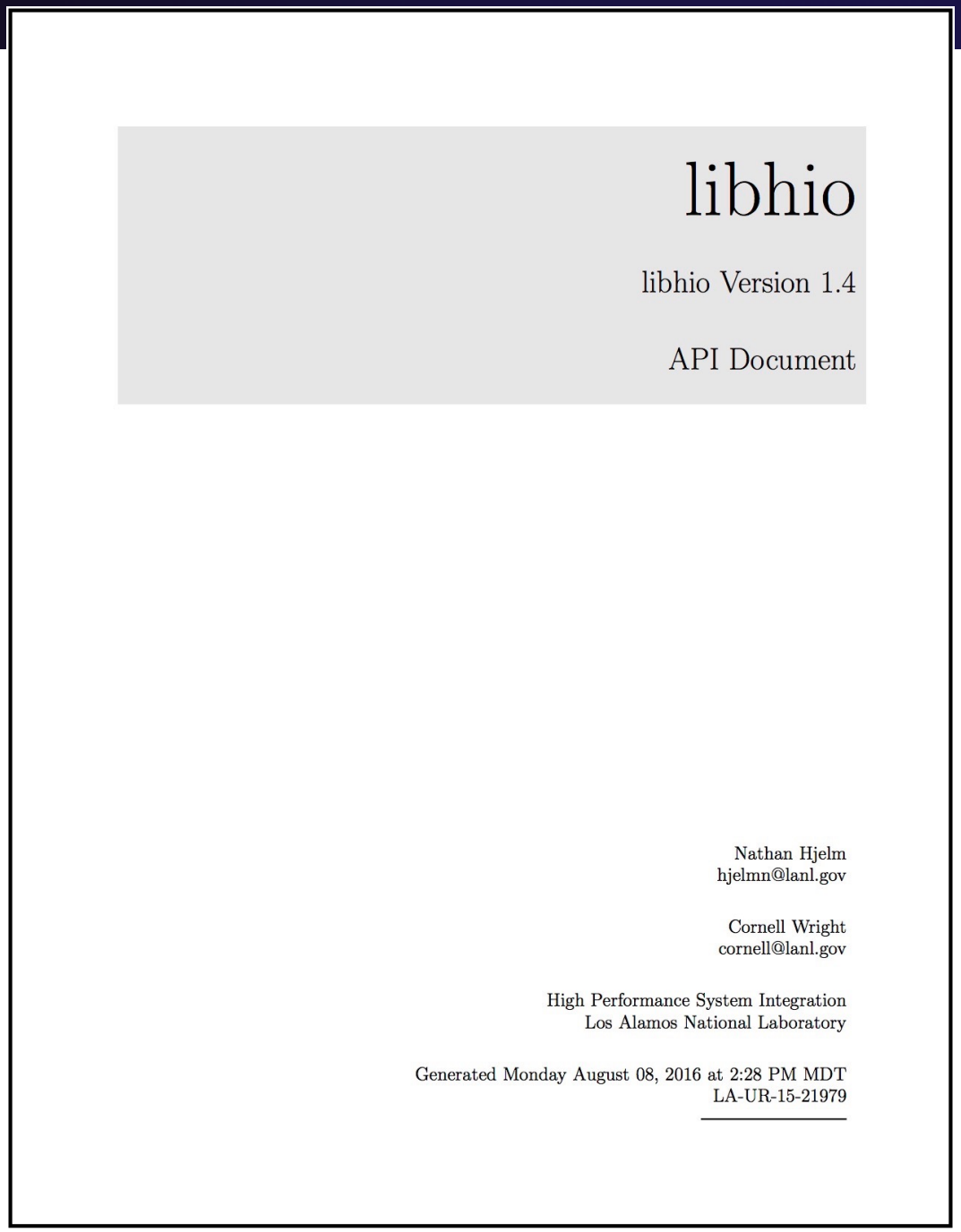
- Provided early access to API
- Now provides fall back
- Translates directly to POSIX or C streaming IO (stdio)
  - Unique element addressing translates directly to a file per element per process
  - Shared element addressing translates directly to a single shared file per element

HIO IO Modes: File Per Node

- Targeted originally for Lustre
- Based on ideas from plfs
  - Turns most writes into contiguous writes
- Does not support read-write mode
- N-1 (element_shared) application view maintained
- Different read vs. write node count supported
- Reduces number of files by 32:1 or 68:1
- Less metadata load than traditional N → N
- Less file contention than traditional N → 1
- Cross-process data lookup supported with MPI-3 Remote Memory Access (RMA)
- Typically better performance for shared file access

HIO Development Status

- API fully documented with Doxygen
- Version 1.4.0.0 released on GitHub (open source)
  - ~7.5 KLOC
- HDF5 plugin under development
- Extensive HIO / DataWarp / Lustre regression test suite
- Ongoing work:
  - Import / Export utility
  - Checkpoint interval advice
  - Performance enhancements
  - Instrumentation enhancements



libhio

libhio Version 1.4

API Document

Nathan Hjelm
hjelmn@lanl.gov

Cornell Wright
cornell@lanl.gov

High Performance System Integration
Los Alamos National Laboratory

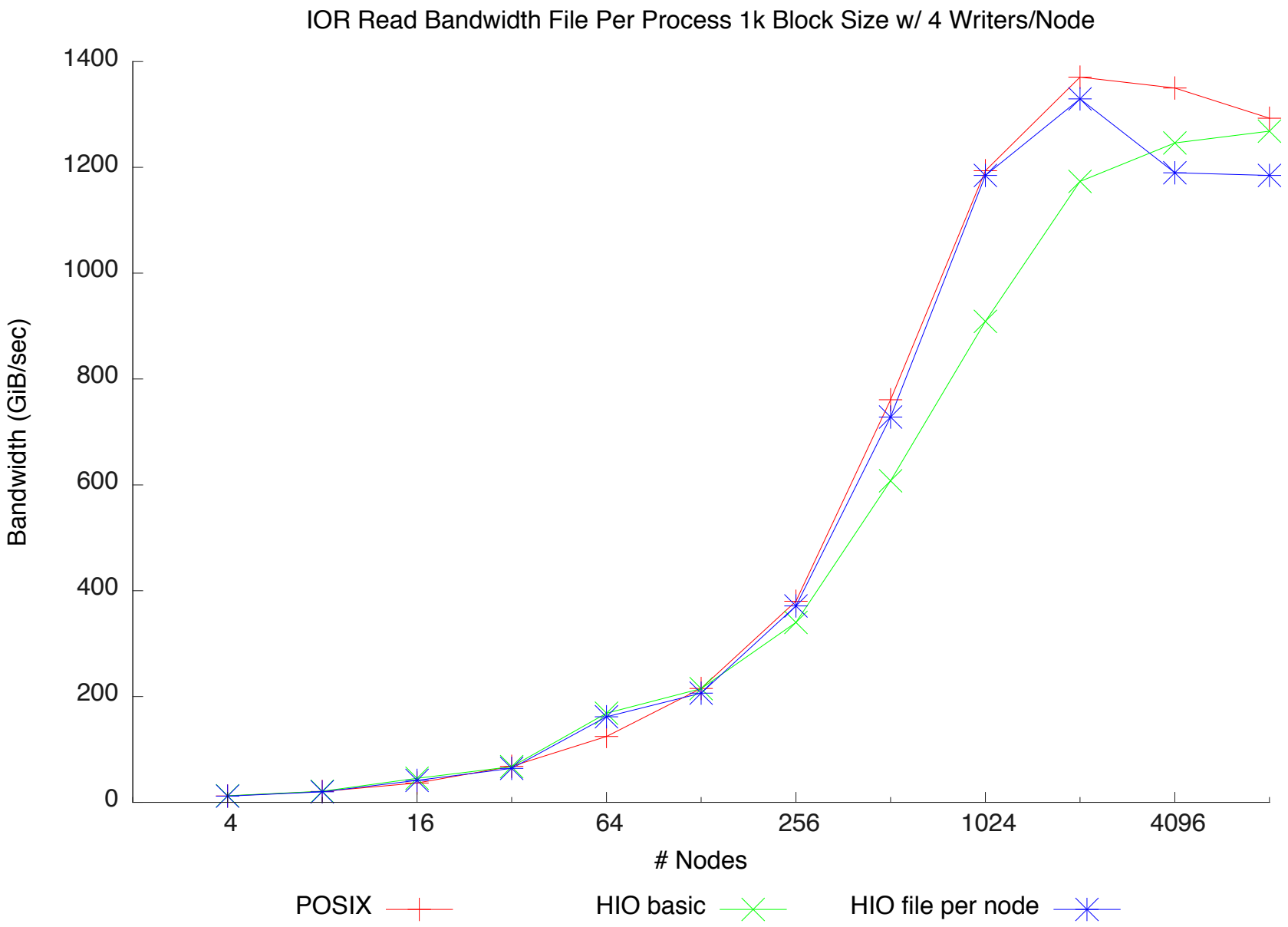
Generated Monday August 08, 2016 at 2:28 PM MDT
LA-UR-15-21979

- libhio Document
  - API Document
  - User's Guide
- Library description
- HIO APIs
- Configuration variables
- 40 Pages

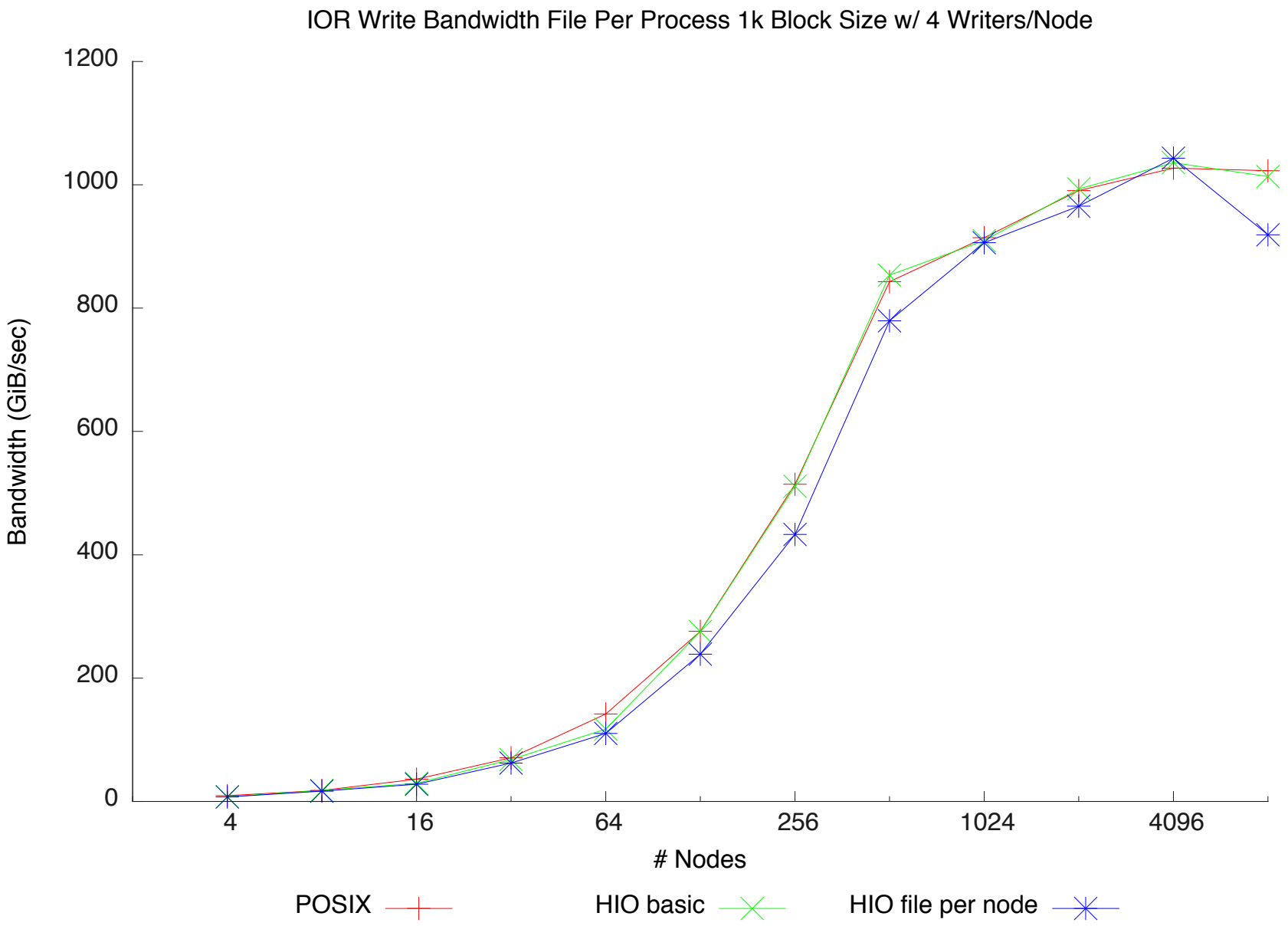
Performance: DataWarp

- **Trinity Phase 2**
  - Cray XC-40
  - 8192 Intel KNL nodes
    - 68 Cores w/ 4 hypertheads/core
  - 234 Cray DataWarp nodes
    - 2 x 4 TB Intel P3608 SSDs
    - Aggregate bandwidth of ~ 1200 GiB/sec
  - Open MPI master hash *6886c12*
- **Modified ior benchmark**
  - Added backend for HIO
  - 1 MiB block size
  - 8 GiB / MPI process
  - 4 MPI processes/node

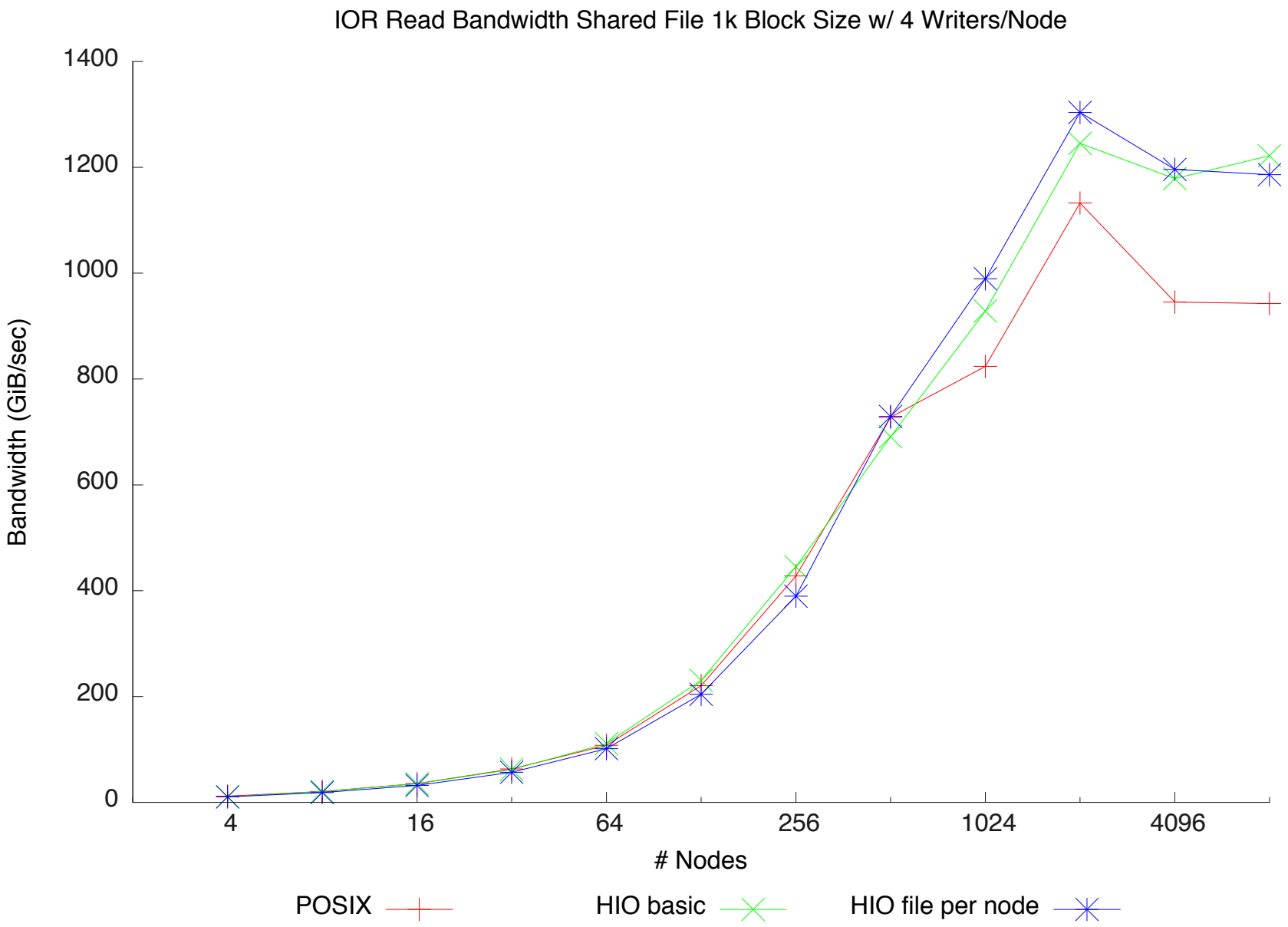
Performance: DataWarp - File Per Process - Read



Performance: DataWarp - File Per Process - Write



Performance: DataWarp - Shared File - Read



In Summary . . .

- HIO provides:
  - High Performance I/O
  - That isolates applications from I/O technology shifts
  - And improves application performance and reliability
  - While reducing development costs

- HIO is ready to use now:
  - Flexible API set
  - Documented
  - Tested
  - Well structured for future enhancement and extensions

Questions ?

Thank You