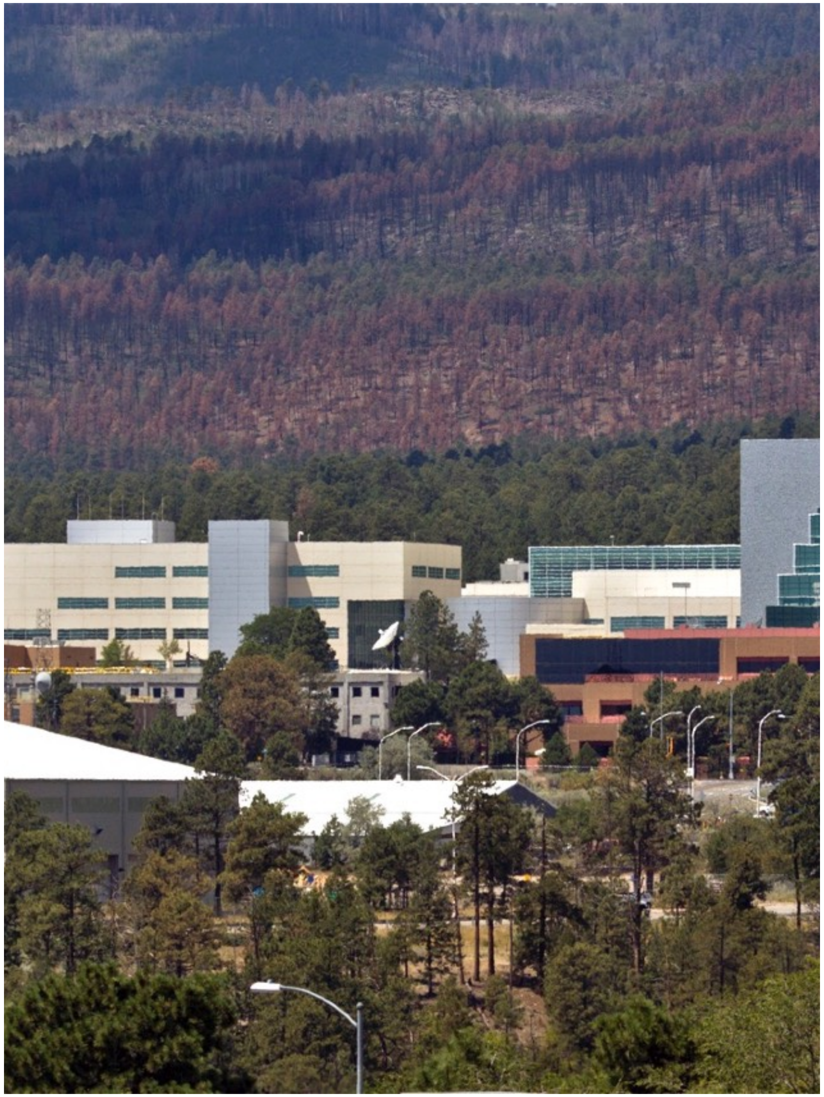
API and Usage of libhio on XC-40 Systems

May 24, 2018

Nathan Hjelm

Cray Users Group
May 24, 2018



Outline

- Background
- HIO Design
- HIO API
- HIO Configuration
- ~~Performance~~
- Summary

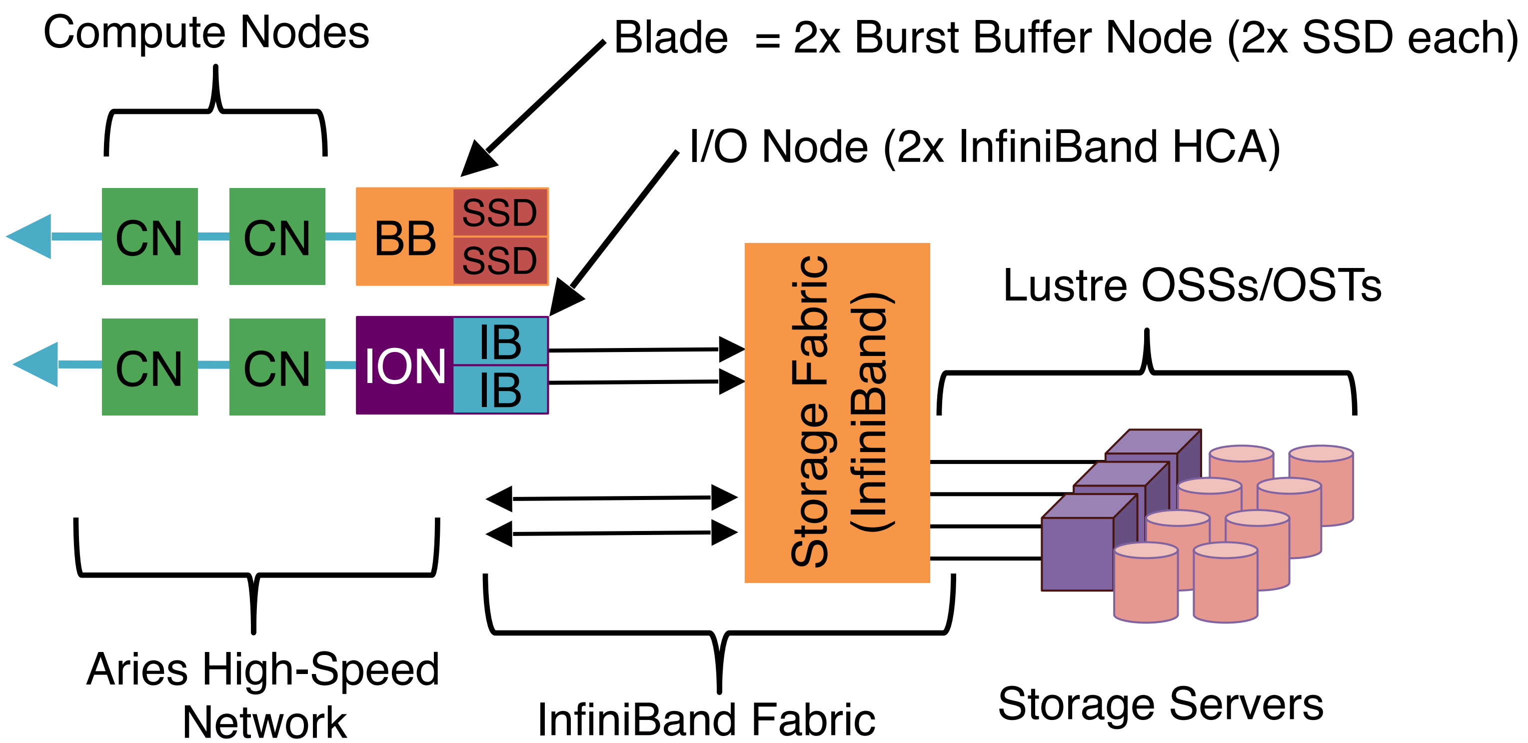
Background: Terminology

- **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 (LANL) and Cori (NERSC).
- **Hierarchical IO Library (HIO)**
  - A LANL developed API and implementation 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?

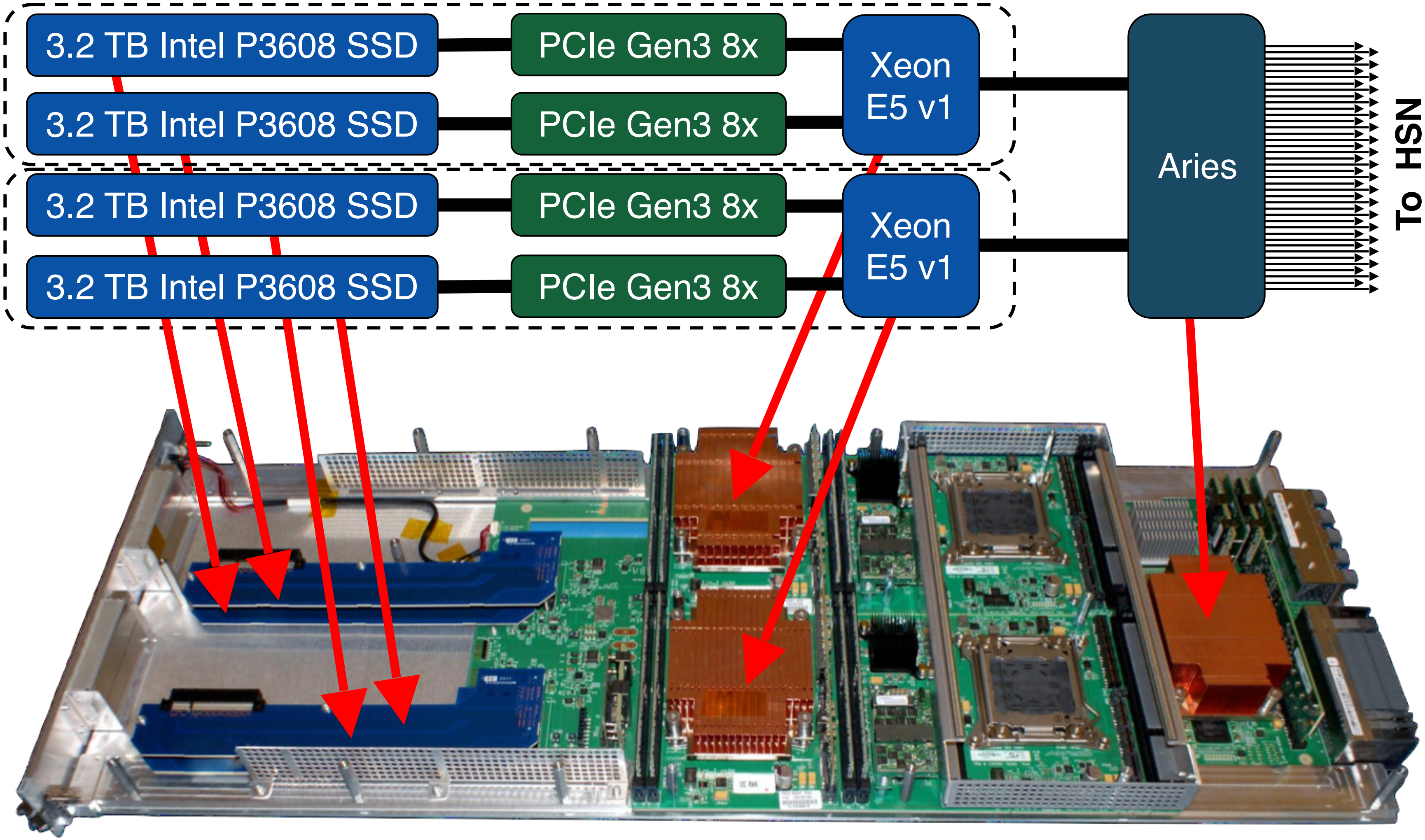
- A Cray SSD storage product on Trinity.
- Consists of:
  - **Hardware**: service nodes connected directly to Aries network each containing two SSDs.
  - **Software**: API/library with functions to initiate stage in/stage out and query stage state.
  - **WLM**: Can be configured in multiple modes using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

Background: Trinity Burst Buffer Architecture



Storage System	Nodes	Capacity	Bandwidth
DataWarp	576	3.7 PB	3.3 TB/sec
Lustre	222	78 PB	1.45 TB/sec

Background: Trinity Burst Buffer Architecture



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? 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 Lustre on Trinity
  - Lustre specific support (interrogation, striping)
  - Add others (GPFS on Sierra) as needed

HIO Concepts

- HIO is centered around a hierarchical abstract name space
- Each level is represented by an HIO object: hio_object_t
- POSIX on-disk structure currently 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 Configuration Interface

- Supports configuration via environment, file, or API calls
- File configuration is specified at context creation time
- Variables can apply globally or to specific HIO objects
- Environment Configuration
  - HIO_*variable_name* - Applies globally.
  - HIO_context_*context_name_variable_name* - Applies to named context
  - HIO_dataset_*dataset_name_variable_name* - Applies to named dataset

Configuration APIs

```
hio_return_t    hio_config_get_count     (hio_object_t object, int *count);
hio_return_t    hio_config_get_info      (hio_object_t object, int index, char **name,
                                          hio_config_type_t *type, bool *read_only);
hio_return_t    hio_config_get_value     (hio_object_t object, char *variable, char **value);
hio_return_t    hio_config_set_value     (hio_object_t object, const char *variable,
                                          const char *value);
```

HIO Hierarchy: Context

- C-language type: hio_context_t
- Encompasses all interaction between an application and libhio
- MPI and local context creation
- All APIs are collective over supplied MPI communicator
- Targets a set of data locations known as “data roots”
  - Posix directories: posix:/some/path
  - Per-job DataWarp: datawarp
  - Persistent DataWarp: datawarp-name

Context APIs

```
hio_return_t    hio_init_single   (hio_context_t *new_context, const char *config_file,
                                    const char *config_file_prefix, const char *context_name);
hio_return_t    hio_init_mpi      (hio_context_t *new_context, MPI_Comm *comm,
                                    const char *config_file, const char *config_file_prefix,
                                    const char *name);
hio_return_t    hio_fini          (hio_context_t *context);
```

HIO Hierarchy: Dataset

- C-language type: hio_dataset_t
- Can have unique (HIO_SET_ELEMENT_UNIQUE) or shared (HIO_SET_ELEMENT_SHARED) offset mode
- Equivalent to a file (or set of files for unique offset space)
- Contains 0+ named elements
- Dataset instance is a context, dataset, dataset id triple
- Collective open/close
- Flags specify read (HIO_FLAG_READ), write (HIO_FLAG_WRITE), create (HIO_FLAG_CREATE), and truncate (HIO_FLAG_TRUNC)

Dataset APIs

```
hio_return_t    hio_dataset_alloc  (hio_context_t context, hio_dataset_t *set_out,
                                     const char *name, int64_t set_id, int flags,
                                     hio_dataset_mode_t mode);
hio_return_t    hio_dataset_open   (hio_dataset_t dataset);
hio_return_t    hio_dataset_close  (hio_dataset_t dataset);
hio_return_t    hio_dataset_free   (hio_dataset_t *dataset);
hio_return_t    hio_dataset_unlink (hio_context_t context, const char *name,
                                     int64_t set_id, hio_unlink_mode_t mode);
```

HIO Hierarchy: Element

- C-language type: hio_element_t
- Named binary data within a dataset
- No collective calls within element APIs
- Contiguous and strided (read scatter, write gather)
- Blocking and non-blocking (request based)

```
Element APIs
hio_return_t    hio_element_open    (hio_dataset_t dataset, hio_element_t *element_out,
                                     const char *name, int flags);
hio_return_t    hio_element_flush   (hio_element_t element, hio_flush_mode_t mode);
hio_return_t    hio_element_close   (hio_element_t *element);
hio_return_t    hio_element_write   (hio_element_t element, off_t offset,
                                     unsigned long reserved0, const void *ptr, size_t count,
                                     size_t size);
hio_return_t    hio_element_read    (hio_element_t element, off_t offset,
                                     unsigned long reserved0, void *ptr, size_t count,
                                     size_t size);
```

*note: flags, reserved0 must be 0

HIO Configuration: General

- Multiple data layouts: dataset_file_mode
  - basic: recommended for DataWarp
  - file_per_node: recommended for Lustre when using shared offset mode
- Control for directory sub-directory naming: dataset_posix_directory_mode
  - hierarchical: context_name.hio/dataset_name/dataset_id
  - single: context_name.dataset_name.dataset_id.hiod

dataset_file_mode	Data layout to use: basic (default), file_per_node
dataset_posix_directory_mode	Directory naming scheme: hierarchical (default), single
dataset_block_size	Block size to use for file_per_node mode
posix_file_api	POSIX file APIs to use: posix (read/write), stdio (fread/fwrite), pposix (pread/pwrite)
data_roots	Comma-delimited list of storage targets: posix paths, datawarp

HIO Configuration: Data Layout

- Dataset configuration variable: dataset_file_mode
- Two data layouts supported: basic, file_per_node

- Basic mode:
  - Provided early access to API
  - Translates directly to POSIX or C streaming IO (stdio)

- File_per_node:
  - Provides good performance for Lustre with all workloads
  - Appends all writes from node-local processes to same file
  - Uses Lustre group locking if available
  - 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
  - Lower filesystem metadata load than traditional N → N

HIO XC-40 Usage: DataWarp

- Automatically detected by default
- Since v1.4.1 supports both persistent and per-job DataWarp allocations
- Per-job DataWarp has priority
- Automatically manages space
- Stage-out location is taken from next data root
  - Ex: HIO_data_roots=datawarp,posix:/lustre/scratch1/user

datawarp_stage_out_destination	PFS state-out destination
datawarp_stage_out_stripe_count	Stripe count for data files on Lustre
datawarp_stage_out_stripe_size	Stripe size for data files on Lustre
datawarp_stage_mode	Stage mode: auto (default), immediate, end_of_job, disable
datawarp_keep_last	Number of dataset instances to keep (default: 1)

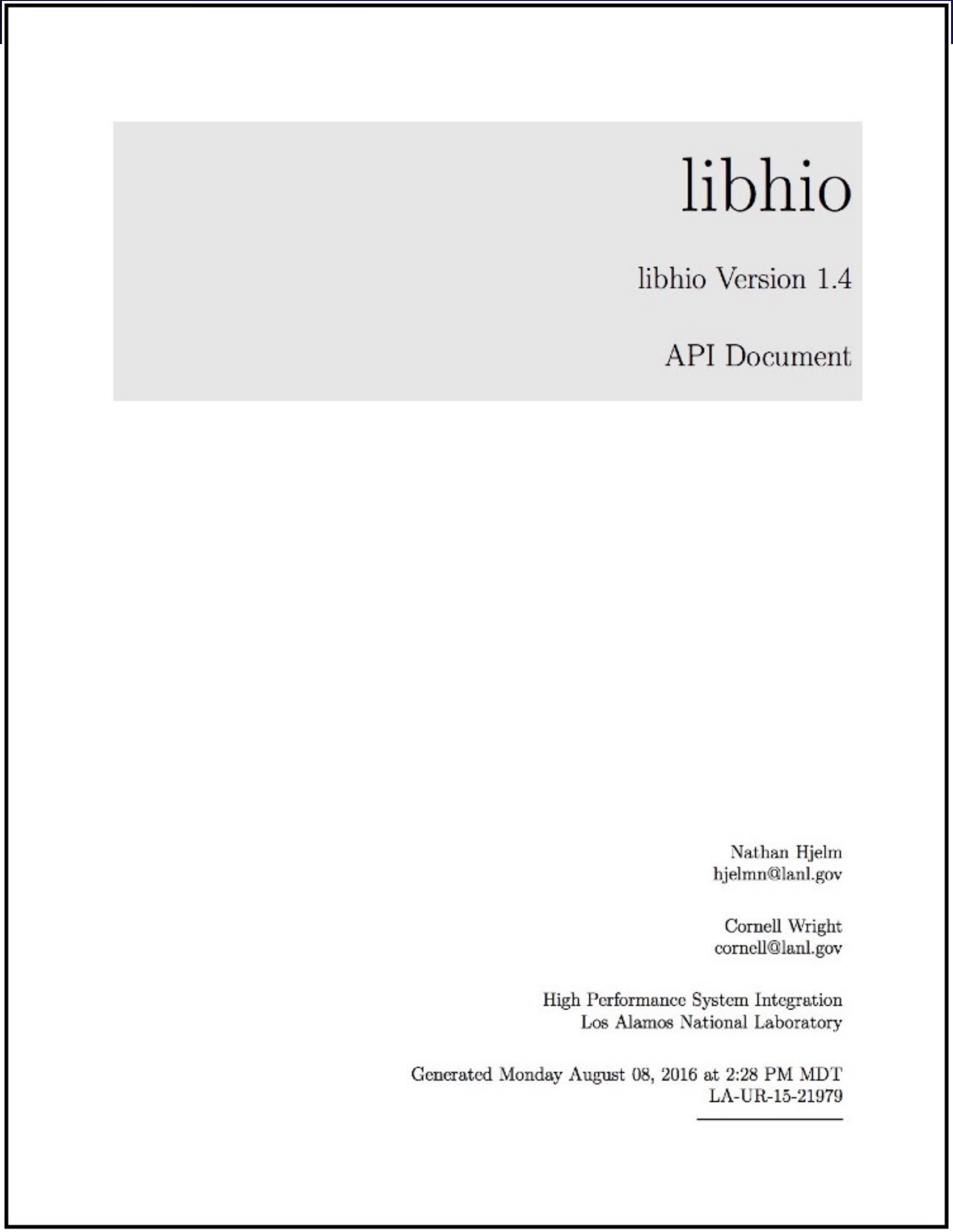
HIO Configuration: Striping

- Provides support for setting filesystem stripe size and count
- Supports Lustre (DataWarp needs CLE 6.0 UP06)
- Applies only to data files in dataset
- Support for group locking (default for file_per_node mode)

stripe_size	Stripe size to use for data files.
stripe_count	Stripe count to use for data files (number of OSTs on Lustre)
lock_mode	Lock mode to use with Lustre: default, group, disable, no-expand

HIO Development Status

- API fully documented with Doxygen
- Version 1.4.1.2 released on GitHub (open source)
  - http://github.com/hpc/libhio
  - ~7.5 KLOC
- HDF5 plugin complete
  - Looking at improving plugin
- Extensive HIO / DataWarp / Lustre regression test suite
- Ongoing work:
  - Import / Export utility
  - Checkpoint interval advice
  - Performance enhancements
  - Instrumentation enhancements
  - Support for node-local burst-buffers



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

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

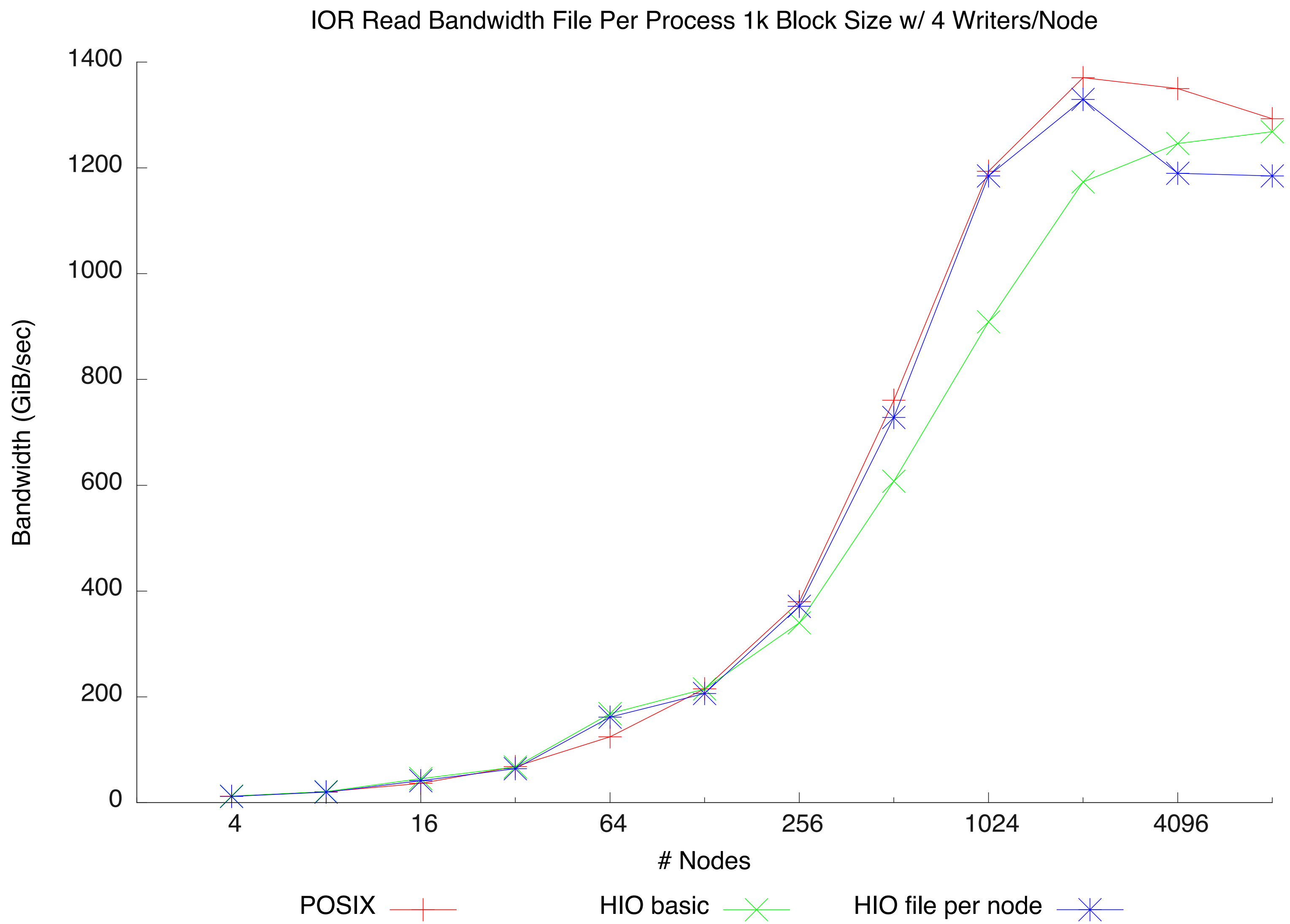
Questions ?

Thank You

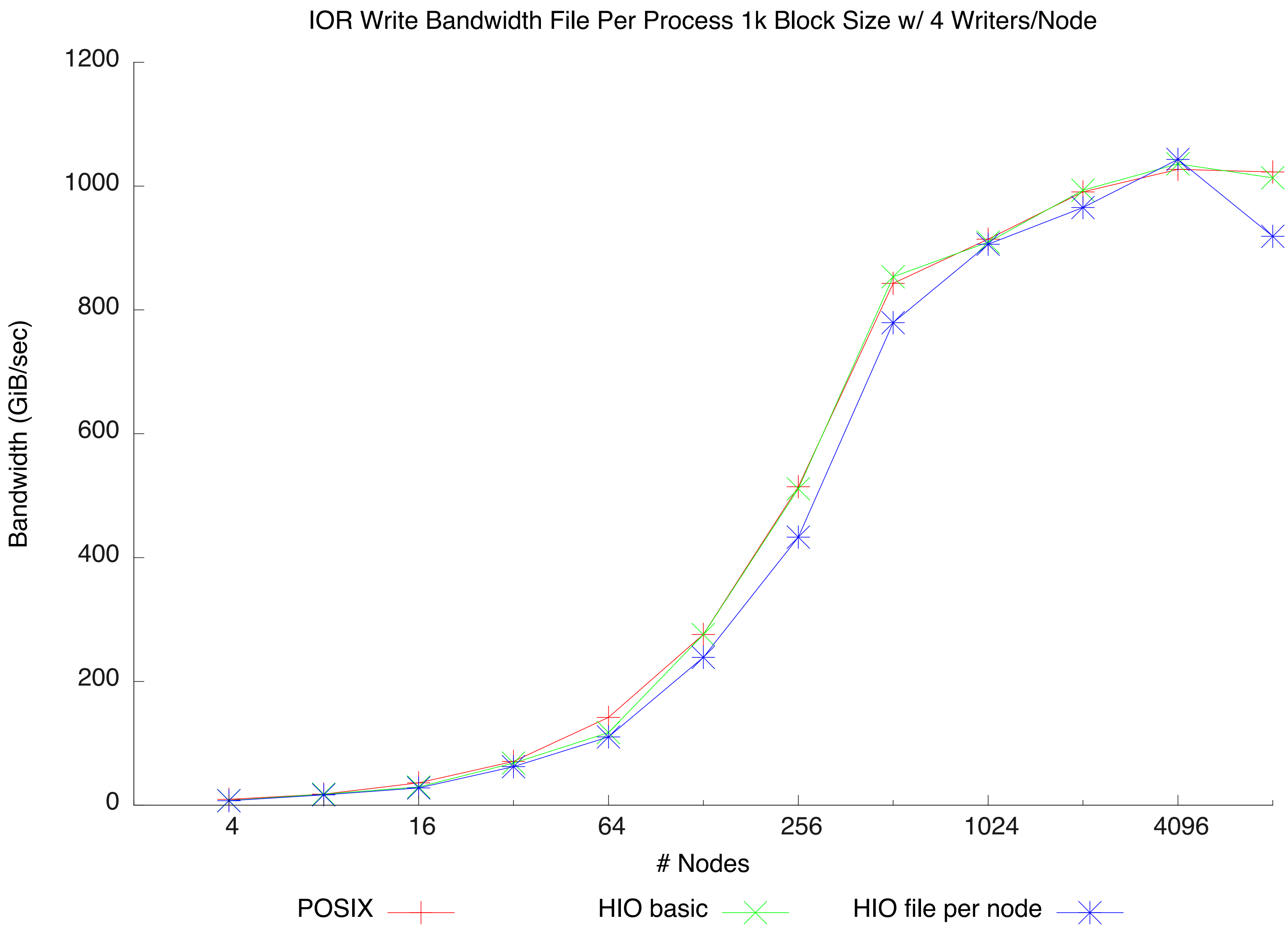
Experimental Setup

- **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

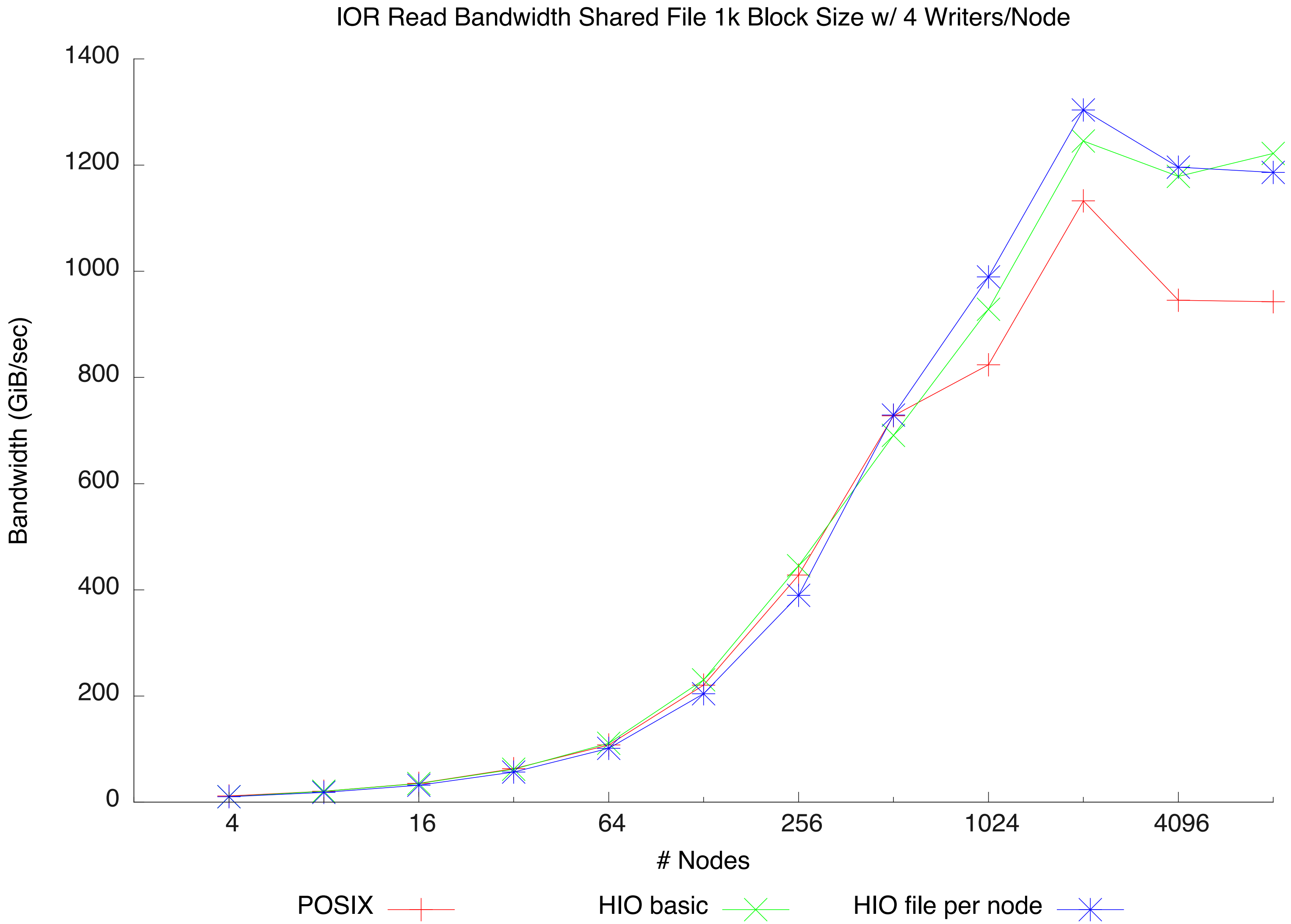
Performance: File Per Process - Read



Performance: File Per Process - Write



Performance: Shared File - Read



Performance: Shared File - Write