Interfacing HDF5 with A Scalable Object-centric Storage System on Hierarchical Storage

May 8, 2019

Jingqing Mu*, Jerome Soumagne*, Suren Byna†, Quincey Koziol†, Houjun Tang†, Richard Warren*

*The HDF Group
†Lawrence Berkeley National Laboratory
Outline

- Challenges and Motivation
- Proactive Data Containers (PDC)
- HDF5 Virtual Object Layer (VOL) and PDC
- Performance Evaluation
- Conclusion and Future work
Outline

- Challenges and Motivation
- Proactive Data Containers (PDC)
- HDF5 Virtual Object Layer (VOL) and PDC
- Performance Evaluation
- Conclusion and Future work
Challenges and Motivation

- Upcoming HPC systems: extreme parallelism, heterogenous memory hierarchy and rapidly increasing amount of data
- Several challenges for I/O libraries
  - Limitation of POSIX I/O w.r.t. extreme parallelism
  - Inability to move data btw different levels of storage transparently and efficiently
- Solutions and directions
  - Object-based storage semantics
  - Storage abstraction and asynchronous I/O
- Is it possible to prevent application code changes?
Outline

- Challenges and Motivation
- Proactive Data Containers (PDC)
- HDF5 Virtual Object Layer (VOL) and PDC
- Performance Evaluation
- Conclusion and Future work
Proactive Data Containers (PDC)

- Explore next generation storage systems and interfaces
- Object centric storage
- Autonomous data management
  - Proactive use of memory hierarchy
- Support for extracting information from data
  - Information management
  - Simulation time analytics
  - Interaction among multiple datasets
Proactive Data Containers (cont’d)

PDC organizes data as a set of *objects* within a *Container*.

*Object* is a generic term to describe byte streams in an abstract manner.

*Region* is the basic and fine-grain unit for data movement operations in PDC.
No explicit data movement

Object mapping
- (Transformation)
- Data movement operations *implicit*
  - Similar to `mmap()`

Concurrent access
- Explicit lock operation per region
- Unlocked region = data movement can occur from/to that region

Primitives: `map/unmap` `lock/unlock`
Proactive Data Containers (cont’d)

- **Client / server architecture**
- **Data movement and I/O realized **asynchronously**
  - Further transfers to deeper levels of the storage hierarchy can be handled by PDC server
  - Overlap computation with I/O operations
- **Application’s buffers, when mapped, can only be used and modified once a lock is acquired**
Proactive Data Containers (cont’d)

- Deployment challenges
  - Ease of deployment for application user
    - Cannot require root privileges
  - Resource management
    - Must co-exist with application

- Multiple deployment options
  - Shared-mode (co-located within the same node / run)
  - Dedicated mode (as a separate allocation)
Proactive Data Containers (cont’d)

Shared Mode

Dedicated Mode

5/8/2019
Cray User Group 2019
Outline

- Challenges and Motivation
- Proactive Data Containers (PDC)
- HDF5 Virtual Object Layer (VOL) and PDC
- Performance Evaluation
- Conclusion and Future work
HDF5

- Well-established I/O middleware package
  - Scientific and industrial applications
- File / Groups / Objects are base components of data model
  - Example: 1 group + 1 dataset: “/G1/D1”

- Parallel HDF5 uses “native” HDF5 on top of MPI I/O
  - However tied to POSIX I/O

File on Disk

<table>
<thead>
<tr>
<th>File Superblock</th>
<th>Object header</th>
<th>Object data</th>
</tr>
</thead>
</table>

Contiguously mapped memory space to disk
HDF5 Virtual Object Layer (VOL)

- Allows developers to redefine the HDF5 I/O API calls
- Re-route to corresponding VOL connector backend seamlessly
- Connector responsible for I/O calls

API

H5Dcreate()

VOL

H5VL_dataset_create()

Connector

Connector Dataset Create Callback

Perform operation on storage

Record info
HDF5 VOL and PDC

- Currently only implements a subset of the HDF5 API
  - HDF5 files mapped to PDC containers
  - HDF5 datasets mapped to PDC objects
  - PDC regions similar to HDF5 selections

- File create, open and close operations are a direct match to PDC container operations
HDF5 VOL and PDC

- Dataset R/W operations require some extra handling
  - No explicit read and write semantics in PDC
  - Similar to `mmap()` way of writing to a file
    - `mmap()` + `memcpy()`
- Not a good match for PDC
  - Do not want to `map()` / `unmap()` in every call
  - Missing API functionality from HDF5

```c
static herr_t
H5VL_pdc_dataset_write(void * dset,
hid_t mem_type_id, hid_t mem_space_id,
hid_t file_space_id, hid_t dxpl_id,
const void *buf, ...) {
    // HDF5 selection to PDC region
    ...
    PDCbuf_obj_map((void *)buf, mem_type,
        mem_reg, dset->obj.obj_id, obj_reg);
    PDCreg_lock(dset->obj.obj_id, obj_reg, WRITE, NOBLOCK);
    ...
    PDCreg_unlock(dset->obj.obj_id,
        obj_reg, WRITE);
    PDCbuf_obj_unmap((void *)buf, mem_reg,
        dset->obj.obj_id, obj_reg);
}
```
HDF5 VOL and PDC

- Applications can benefit from PDC with a single line code change

- Which benefits?
  - Asynchronous I/O to subsequent levels of storage
  - Object oriented storage semantics
    - Storage and extraction of metadata
  - Analysis and transforms on PDC server (not demonstrated here)

```
/* Register PDC VOL */
pdc_vol_id = H5VLregister_connector(&H5VL_pdc_g, H5P_DEFAULT);

/* Create a new file access property */
fapl_id = H5Pcreate(H5P_FILE_ACCESS);

/* Set the VOL */
H5Pset_vol(fapl_id, pdc_vol_id, NULL);

/* Create file */
file_id = H5Fcreate(argv[1], H5F_ACC_TRUNC, H5P_DEFAULT, fapl_id);

/* Close property */
H5Pclose(fapl_id);

/* Close file */
H5Fclose(file_id);
```
Outline

- Challenges and Motivation
- Introduction to Proactive Data Containers (PDC)
- HDF5 Virtual Object Layer (VOL) and PDC
- Performance Evaluation
- Conclusion and Future work
Experimental Setup

- Cori supercomputer at LBNL
  - 1600 “Haswell” compute nodes
  - 128 GB DRAM/node
  - 32 cores/node
  - SSD-based “Burst Buffer”
  - HHD-based shared file system Lustre

- Shared mode
  - 1 PDC server on each node, remaining 31 cores for application

- Dedicated mode
  - PDC servers and user’s application are on separate nodes

- Relies on Mercury (http://mercury-hpc.github.io) + Cray DRC + OFI libfabric

- Benchmarks
  - VPIC-I/O: I/O of a large-scale plasma physics simulation code
  - BD-CATS-I/O: I/O of a big data clustering analysis code for particle data
Performance evaluation

- H5Dwrite() performance
  - Varying number of PDC servers
  - VPIC-IO

- Results
  - Shared mode is 3.3X and 3.5X faster compared to independent and collective native HDF5 on average
  - Dedicated mode is 4.7X and 4.4X faster
  - Dedicated mode with Cray GNI is 8.3X and 6.6X faster
Performance evaluation (cont’d)

- **H5Dwrite() performance**
  - Varying number of PDC servers with fixed 32 node clients
  - Dedicated mode w/ up to 32 nodes
  - VPIC-IO

- **Results**
  - Having less than a half of the nodes still provides good performance
Performance evaluation (cont’d)

- **H5Dwrite() performance**
  - 5 time steps
  - Shared-mode
  - VPIC-IO

- **Results**
  - An average speedup of 7.3X and 6.6X compared to native HDF5 collective and independent I/O separately
Performance evaluation (cont’d)

- H5Dwrite() performance
  - Single time step
  - Total Execution Time for VPIC-IO

- Results
  - Achieves 2.9X and 2.5X faster performance compared to HDF5 collective and independent I/O methods respectively
Performance evaluation (cont’d)

- **H5Dread() performance**
  - BD-CATS-IO

- **Results**
  - Shared mode is 1.4X and 2.3X faster compared to collective and independent native HDF5 on average
  - Dedicated mode is 2.5X and 3.8X faster
  - Dedicated mode with Cray GNI is 3.3X and 5.0X faster
Outline

- Challenges and Motivation
- Proactive Data Containers (PDC)
- HDF5 Virtual Object Layer (VOL) and PDC
- Performance Evaluation
- Conclusion and Future work
Conclusion and Future Work

- **HDF5 PDC VOL connector**
  - Async I/O and transparent data movement to multi storage tiers
  - Object-oriented storage

- **Minimal to zero code modification (none w/ environment variable)**

- **Use native network fabric transports such as Cray GNI**

- **Current limitation of HDF5 API**
  - Explicit read and write of data
  - Will look at bringing map semantics into HDF5
  - However would require application code changes

- **External VOL connector development**
Questions

This work is supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract No. DE-AC02- 05CH11231 and DE-SC0016454. This research used resources of the National Energy Research Scientific Computing Center (NERSC).