Use of a New I/O Stack for Extreme-scale Systems in Scientific Applications

Michael Breitenfeld, Neil Fortner, **Jerome Soumagne**

The HDF Group

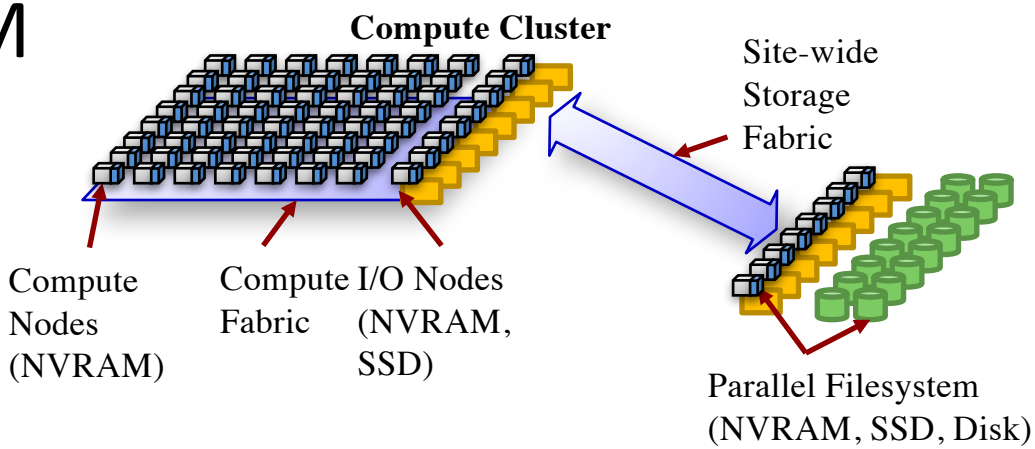
Mohamad Chaarawi, Intel

Quincey Koziol, Lawrence Berkeley National Laboratory

Collaborators: Intel and LBNL

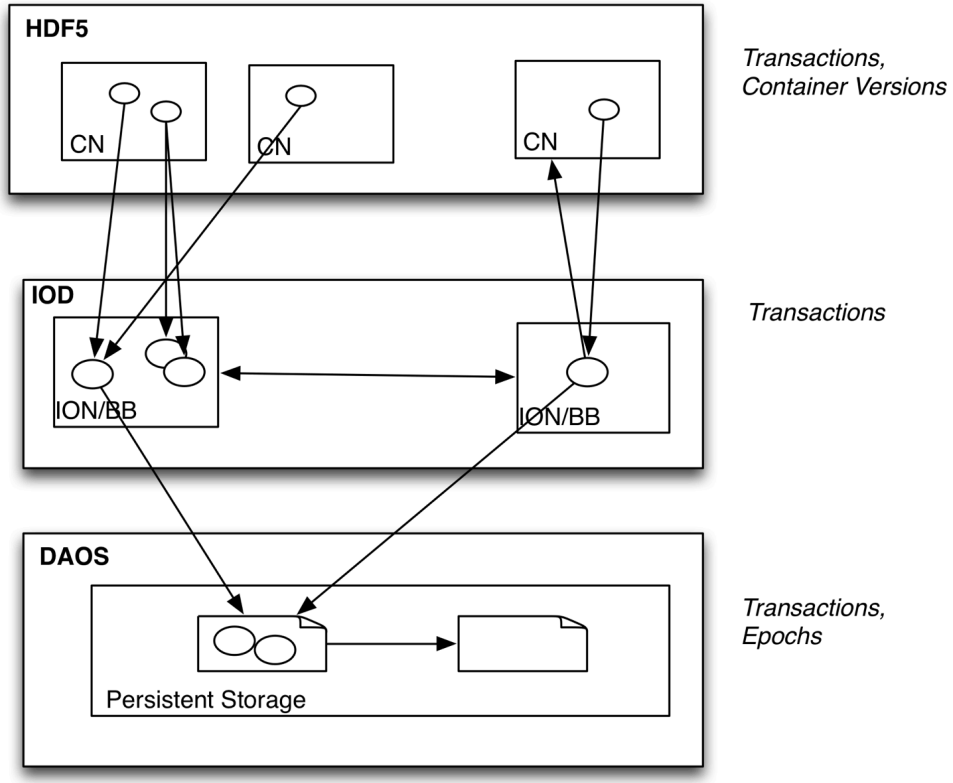
ESSIO Storage Architecture

- Compute Node NVRAM
  - Hot data
    - High valence & velocity
    - Brute-force, ad-hoc analysis
    - Extreme scale-out
  - Full fabric bandwidth
    - O(1PB/s)→O(10PB/s)
  - Extremely low fabric & NVRAM latency
    - Extreme fine grain
    - New programming models



- I/O Node NVRAM/SSD
  - Semi-hot data/staging buffer
  - Fractional fabric bandwidth
    - O(10TB/s)→O(100TB/s)
- Parallel Filesystem NVRAM/SSD/Disk
  - Site-wide shared warm storage
    - SAN limited – O(1TB/s)→O(10TB/s)

FastForward/ESSIO Transactional Stack



- A ***transaction*** consists of a set of updates to a container
  - container ≈ file
  - Updates are added to a transaction, not made directly to a container
  - Updates include additions, deletions, and modifications

HACC — Overview

HACC - Hardware/Hybrid Accelerated Cosmology Code

N-body cosmology code framework where a typical simulation of the universe demands extreme scale simulation capabilities

Primary data model

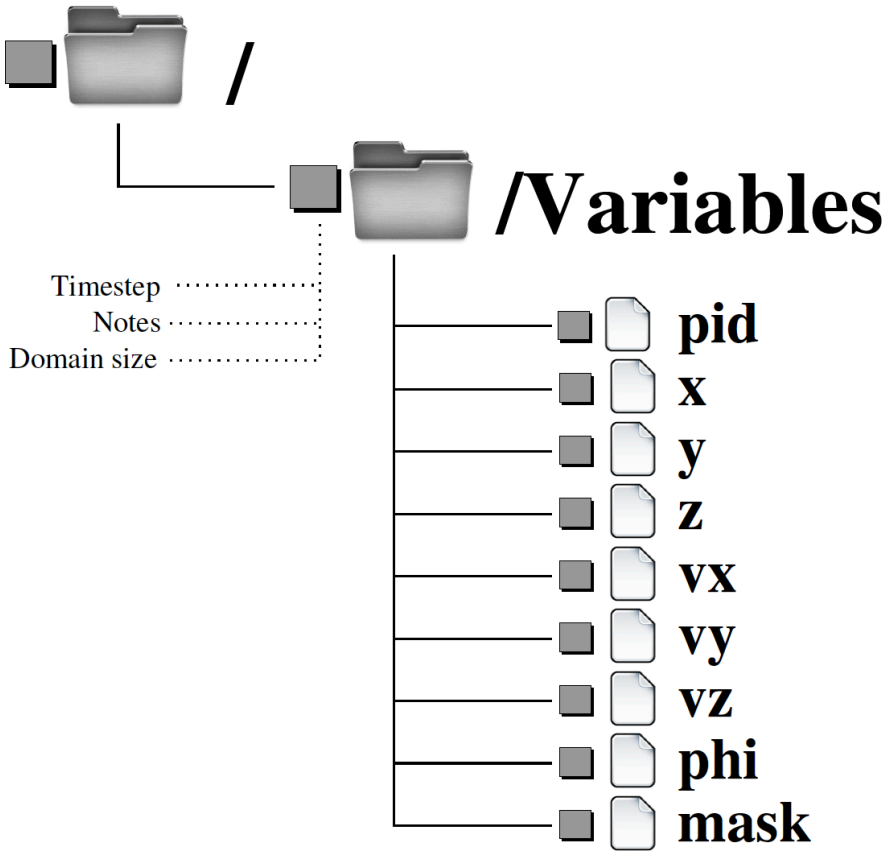
- 9 arrays at full scale of application
  - Position in 3-D, Velocity in 3-D, Simulation Info, Science Data
- Additional metadata augmenting provenance, etc

HACC — Data Model: Currently

Application creates custom binary files

HACC — Data Model: **ESSIO**

~~Application creates custom binary files~~



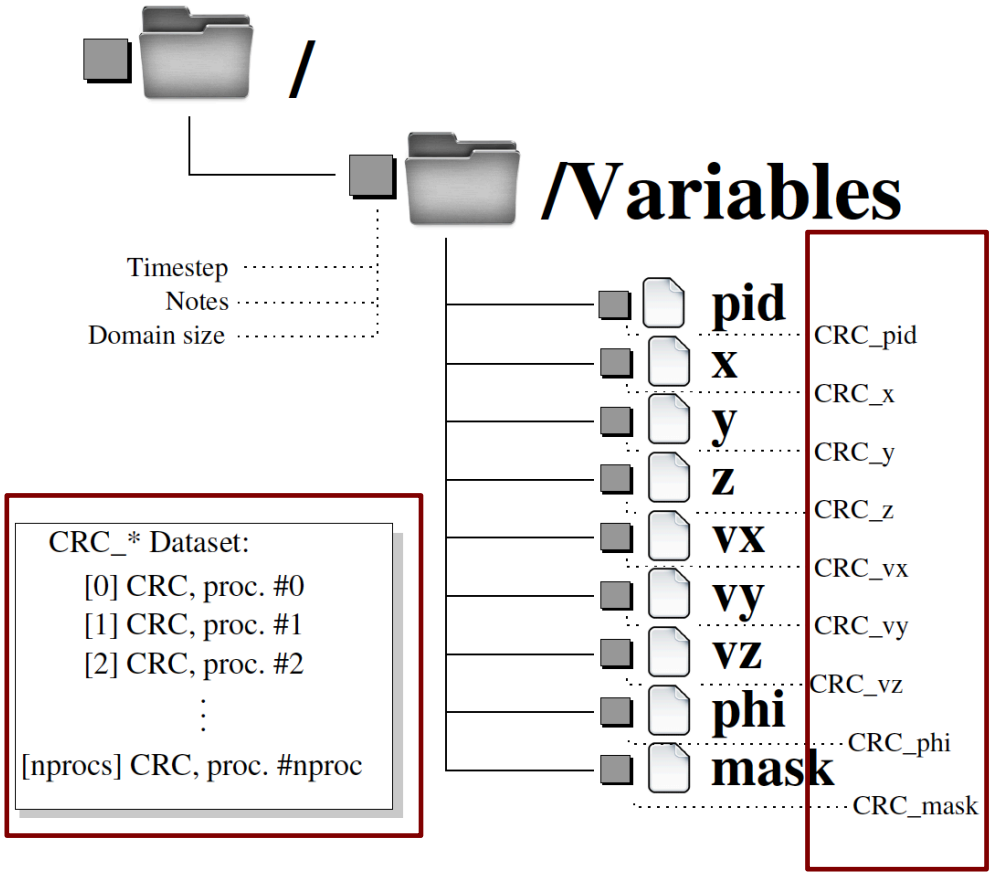
- All application metadata stored in HDF5 container
- HDF5 format is self-describing, using groups, datasets and attributes
- Any visualization or analysis process can be used to investigate science results

HACC — Data Resiliency: Currently

☑ Application stores and verifies checksum from memory to the file and back

HACC — Data Resiliency: **ESSIO**

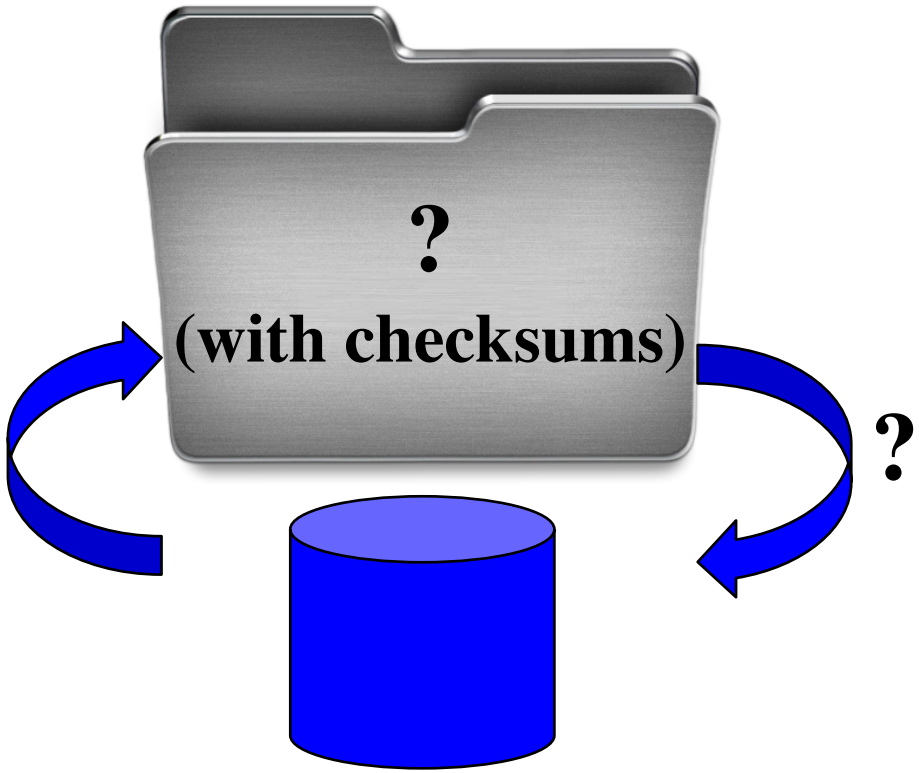
☑ ~~Application stores and verifies checksum from memory to the file and back~~



- Each process calculates and passes checksum of the local array section to HDF5
- HDF5 optionally verifies buffer, and passes checksum with data down the stack
- Checksum verified for every data buffer operation from HDF5 to storage and back

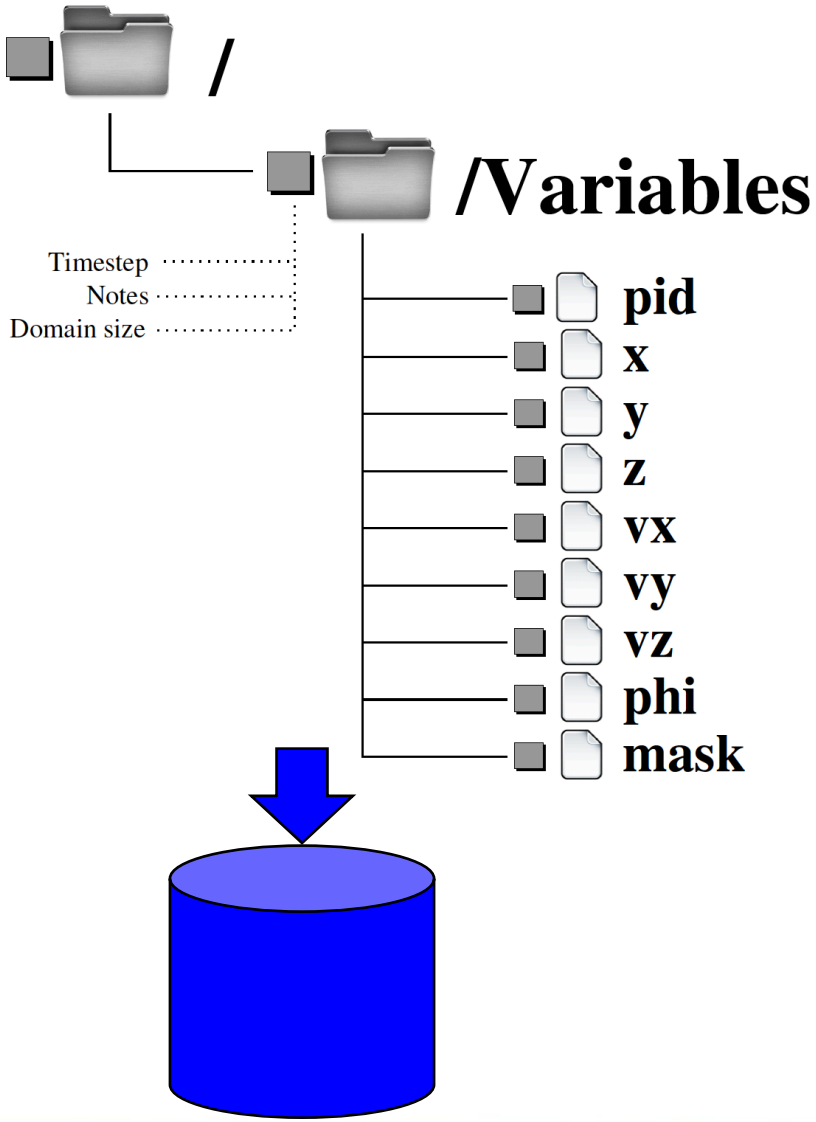
HACC — Fault Tolerance: Currently

Application retries I/O until completed



HACC — Fault Tolerance: **ESSIO**

~~Application retries I/O until completed~~



- Each process writes all checkpoint data to transaction
- Transaction is committed to storage, possibly asynchronously
- If asynchronous, application can test/wait to guarantee data is persistent
- *Future work*: replay event stack on error

High-level HDF5 ESSIO stack libraries

Objective

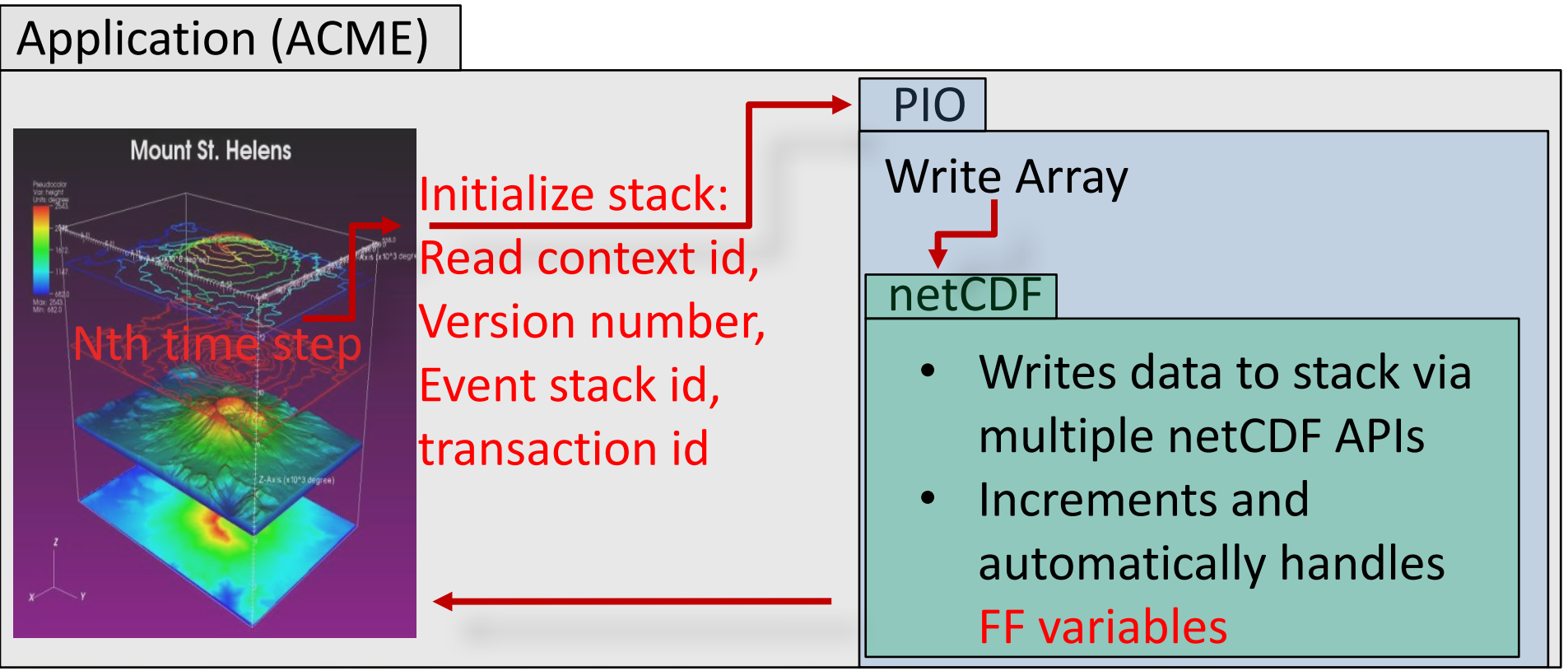
Have the high-level I/O code manage the transaction requests and isolate the application code from the ESSIO stack

Ported Two High-level HDF5 based I/O libraries

(1) NetCDF – A set of software libraries used to facilitate the creation, access, and sharing of array-oriented scientific data in self-describing, machine-independent data formats

(2) Parallel I/O (PIO) – A high-level I/O library which uses as its backend NetCDF

- Global stack variables are passed as arguments to NetCDF and from the application
  - Stack parameters are controlled from within PIO



Future Work

New superset of DAOS – **DAOS-M**

Distributed Persistent Memory Class Storage Model

- DAOS-M server will access memory class storage using a Persistent Memory programming model that directly utilizes load-store access to NVRAM DIMMs
- Extends the current DAOS API to support key-value objects natively

Port and benchmark to DAOS-M:

(1) Legion Programing System (not presented here)

(2) NetCDF to DAOS-M