Why wait!? Hades:
An Active, Content-Aware System for Precalculating Derived Quantities

SCALABLE COMPUTING SOFTWARE LABORATORY

Jaime Cernuda Luke Logan Anthony Kougkas Xian-He Sun
jcernudagarcia@hawk.iit.edu, llogan@hawk.iit.edu, akougkas@iit.edu, sun@iit.edu

ILLINOIS INSTITUTE OF TECHNOLOGY

Introduction

1. Many HPC workflows are divided into separate producer and analysis phases
2. Raw data dumped into a file during the producer phase
3. Analysis derives quantities by scanning the entire file – **Significant I/O cost!**
4. We propose Hades, a content-aware I/O system that actively calculates derived quantities while data is produced to reduce the I/O penalty in the analysis phase

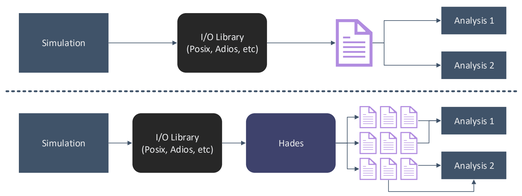


Figure 1: The benefit of content awareness

Challenges

Actively deriving quantities has a number of challenges:

1. Hades needs to manage user data in the complex space of devices that is an HPC cluster.
2. Hades needs a mechanism to accept user define operations and execute them on top of the data.
3. Hades needs to be performant in managing applications metadata and derived metadata as fast as possible.

Hades architecture

1. We intercept the I/O produced by ADIOS (Put/Get)
2. Users upload a custom operation schema to inform Hades on the derived quantities to produce
3. During Put, the raw data will be sent to the Hades runtime to asynchronously calculate the derived quantities
4. Hades will store derived quantities across memory and storage using the Hierarchical Manager
5. Derived data will be promoted to faster storage when they are expected to be used and demoted otherwise

Calculating derived quantities

1. User submits derived quantity operation schema to Hades
2. Raw data shipped to the Calculator Runtime during writes
3. Asynchronous away from the data path
4. Various data transformations are provided by the Hades schema.
5. Currently, MinMax and Inqurievariable

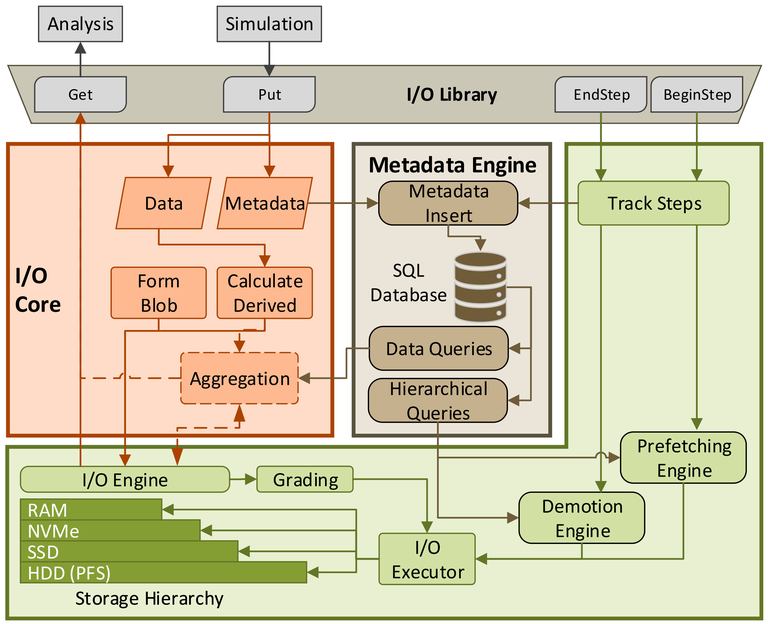


Figure 2: Hades Architecture

Managing the hierarchy

To manage the hierarchy, Hades leverages two core ideas:

1. Data weight. Taking into account, the blob size, usage frequency, etc.
2. The step-wise design of scientific applications.

This is combined through the Hierarchical Manager, which manages two operations:

1. Demoting: Initially, Hades places every data blob in memory. Hades leverages the call to endStep in the simulation to demote blobs. Blobs with high data weigh are demoted earlier.
2. Prefetching: Hades has a parameter, *look ahead steps* that defines how far ahead the prefetcher looks. On a beginStep call, Hades will start prefetching the Blobs for the next n steps.

Metadata

- Operations: value of the current step across the processes.
- Data: global per-variable
  entry of the name, shape, size, and status and local per-process entry representing (start, size).
- Hierarchy: current utilization (data-wise) of devices and the current placement of blobs.

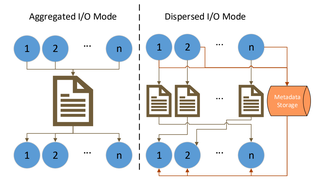
I/O Model

Raw Data: Dispersed I/O (per-process data **blobs** independently)

1. (Pro) Does not require synchronization during writes
2. (Con) Requires more metadata to track each of the blobs

Derived Quantities: Aggregated I/O (data in a single data **blob**)

1. (Pro) Lower metadata cost Unified I/O batch for operations
2. (Con) Requires synchronization during writes



Evaluations

Compute rack, local cluster:

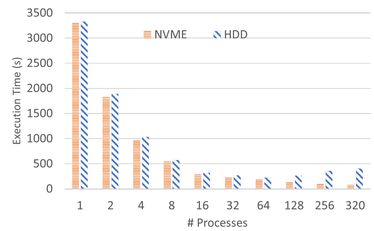
- 40Gb/s isolated network with RoCE enabled
- Dual Intel(R) Xeon Scalable Silver 4114
- 48 GB RAM, NVMe PCIe x8 drive

OrangeFS as PFS

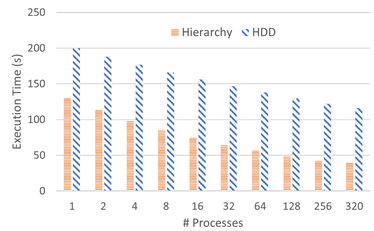
Correctness

- Gray-Scott: models the chemical reaction between two chemicals. Uses queries and Put/Get
- Outputs match except on doubles. Differences in serialization performance and numerical stability
- Can switch between Cereal (fast) and ADIOS (accurate) for serialization

I/O Performance



(a) Simulation Time



(b) Analysis Time

- Compare Gray-Scott with and without hierarchical buffering
- PFS uses HDD, while hierarchy includes node-local NVMe burst buffers