DTIO: Unifying I/O for HPC and AI

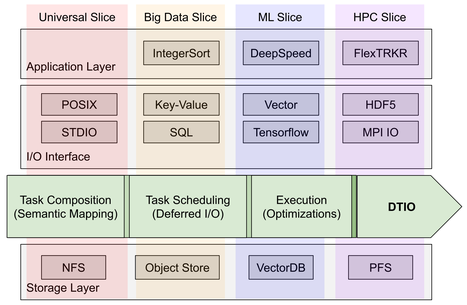
Keith Bateman (kbateman@hawk.iit.edu)
Advisors: Anthony Kougkas, Xian-He Sun
Collaborators: Bogdan Nicolae, Franck Cappello

Gnosis Research Center @ ILLINOIS TECH

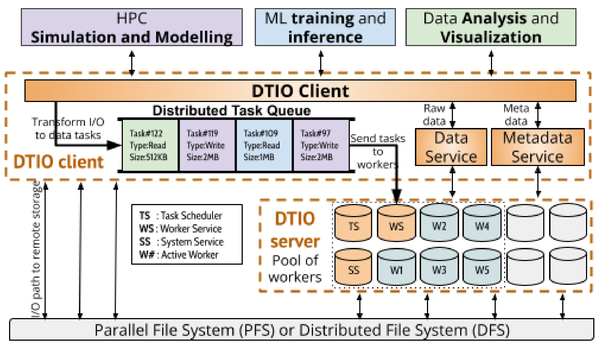
Introduction

- HPC, Big Data, and Machine Learning have become increasingly intertwined, as AI has driven discovery in science (e.g., GenSLMs, OpenFold).
- HPC I/O infrastructure involves a Parallel File System and MPI-IO or HDF5, while ML workloads may utilize tensors or a distributed vector database, and Big Data tends to utilize key-value and object stores.
- There is a need for a system which unifies the existing I/O stacks for the triple convergence of HPC, Big Data, and Machine Learning.

- **Unified I/O**: Unify via a scalable and flexible *task-based* data representation.
- **Deferred Consistency**: DataTasks should be executed as-needed.
- **Hierarchical Integration**: DataTasks should be replicated to improve locality.



Methodology



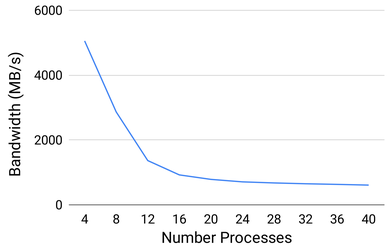
- **DTIO Client** creates and schedules a DataTask, and **DTIO servers** execute the tasks
- **Composition** is generally expected to be done alongside the application (client-side)
- For **scheduling**, centralized deployments can collate information from different apps, while multiprocess deployments scale better
- For **workers**, dedicated execution resources are the performant choice.

DataTask API

- I/O interfaces are intercepted and translated to DataTasks.
- In addition, DataTasks have their own general API:

Op Type	Arguments	Output	POSIX Equivalent
Put	(key, data)	Task_ID	write(open(key), data, count)
Get	(key, count)	Task_ID	read(open(key), &ret, count)
Wait	(Task_ID)	Task_Result	N/A
Query	(Task_ID)	Task_Status	N/A

Relaxing POSIX consistency to improve scalability



POSIX metadata and consistency guarantees cause performance drops for IOR at scale

- **Relaxation of POSIX consistency** in a task system can come in a few ways
- Delay when **creating tasks**, **scheduling tasks**, or **executing tasks**.
- These ideas are often represented naturally with **task queues**, as queued tasks need not be dequeued immediately.

Scheduling constraints to improve resource utilization

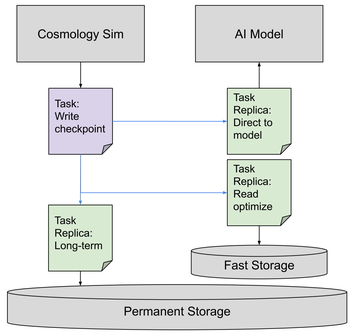
- Asynchronous scheduling of DataTasks allows them to execute during compute, or even delay execution until data is read.
- To achieve improved resource utilization, DataTasks can be scheduled to workers **depending on load**.
- Task Status can be unscheduled, scheduled, or completed

- A simple constraint: schedule a task to the *executor which is currently running the fewest tasks*
- More complex constraint: *track the I/O size of tasks to each executor* and schedule to the executor with the lowest I/O load

procedure Schedule(task, util[])
  Let min be initialized to 0
  for $i = 0; i < len(util); i++$ **do**
    if $util[i] < util[min]$ **then**
      $min \leftarrow i$
    end if
  end for
  Schedule to i
end procedure

Locality via DataTask replication

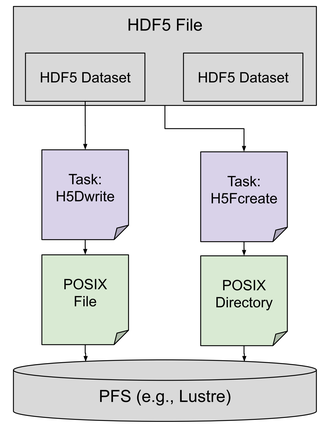
- When we have multiple underlying storage services, some of them may be better for certain usecases. For example, NVMe storage may be faster to read from than a PFS.
- DTIO can intelligently decide which storage to place data on, and even replicate across services as needed.
- For example, an AI-accelerated workload often has to read data to train its models while simulations are running, data could be copied so that the models could be trained at the same time that the simulation writes occur.
- In addition, later consumers may be able to read from a faster storage instead of going to PFS.



Producer-consumer replication optimizations

Mapping HDF5 Calls

- There are various ways to represent higher-level HDF5 data structures as lower-level tasks.
- Different aspects of an HDF5 file should have their own unique IDs, with a subset of the ID used to indicate the file itself and perhaps an indication of type (e.g., dataset, attribute).
- When mapping this to POSIX, either HDF5 datasets would be represented as POSIX files within a directory that represents the HDF5 file or HDF5 datasets would be represented as particular offsets of an HDF5 file represented as a POSIX file.
- Mapping to other interfaces would follow a similar pattern, the important thing is to appropriately represent the HDF5 namespace, which uses the file as a top-level container name.



Example of mapping HDF5 to POSIX through DataTasks

In Partnership With
Argonne NATIONAL LABORATORY

Funded By
U.S. DEPARTMENT OF ENERGY

More Information