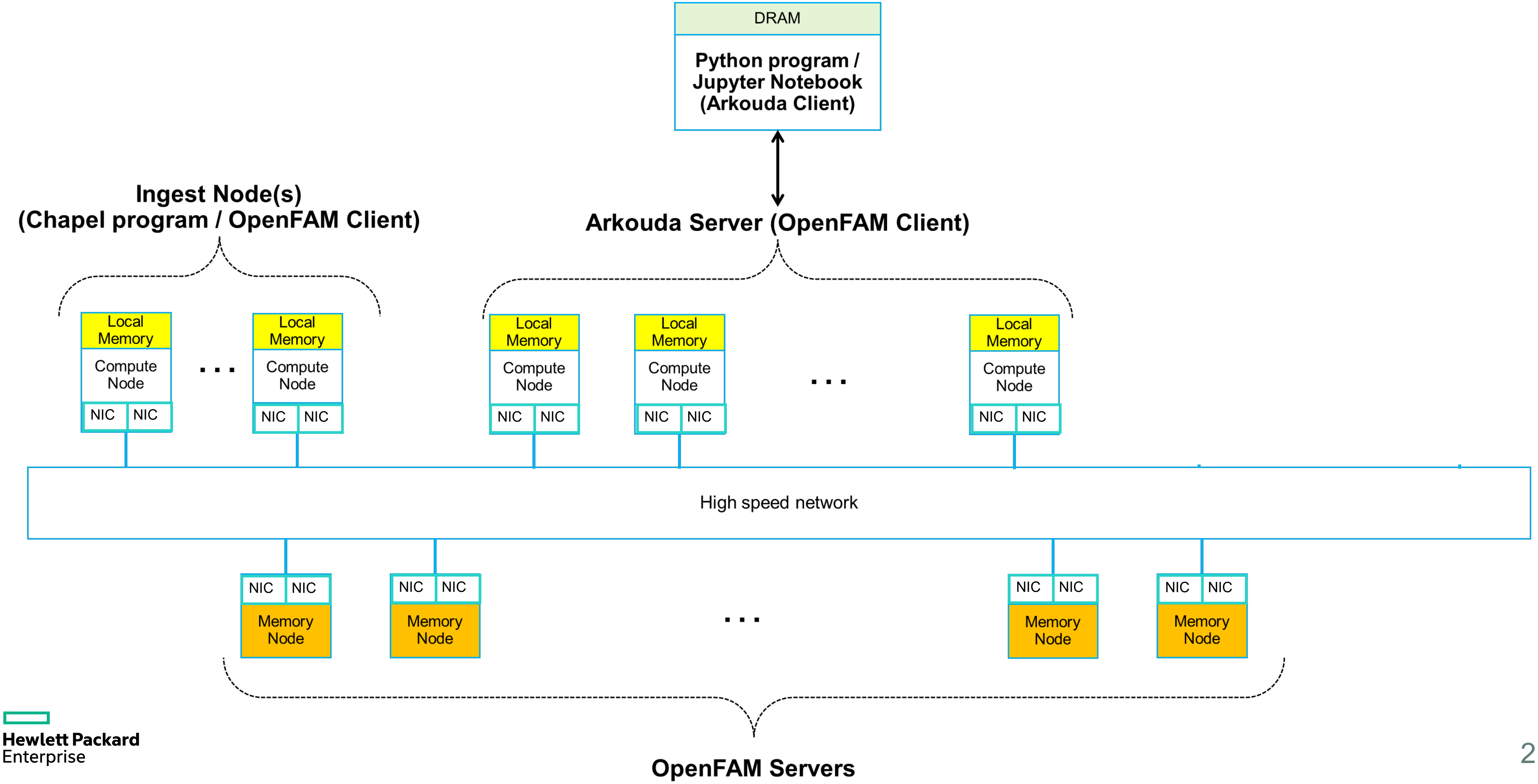
# Coupling Chapel-Powered HPC Workflows for Python, part 2

John Byrne, **Harumi Kuno**, Chinmay Ghosh, Porno Shome, Amitha C, Sharad Singhal, Clarete Riana Crasta, David Emberson, Abhishek Dwaraki

June 7, 2023

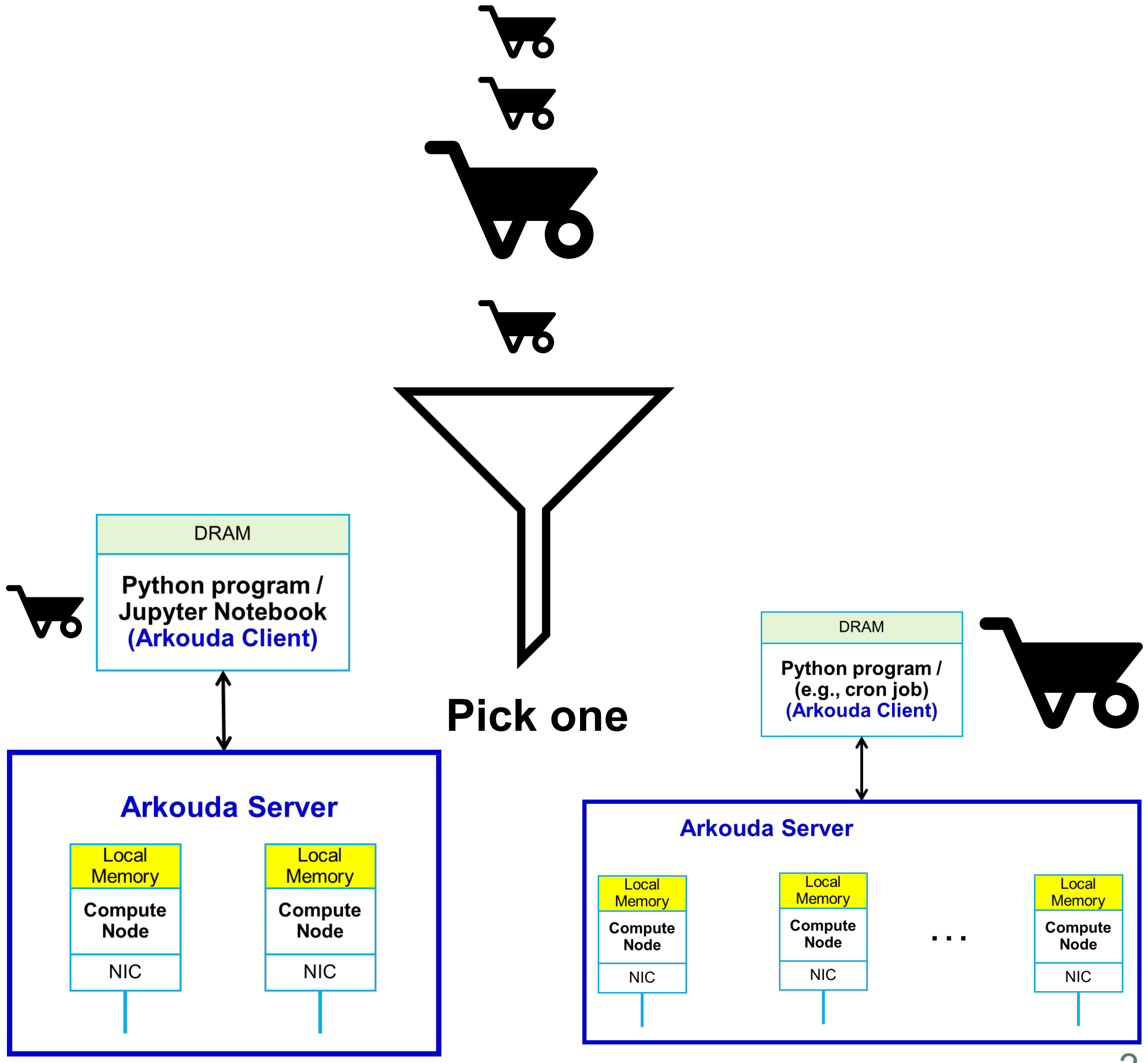
# How quickly can we simultaneously ingest and interactively process data … using Chapel and Fabric-Attached Memory?



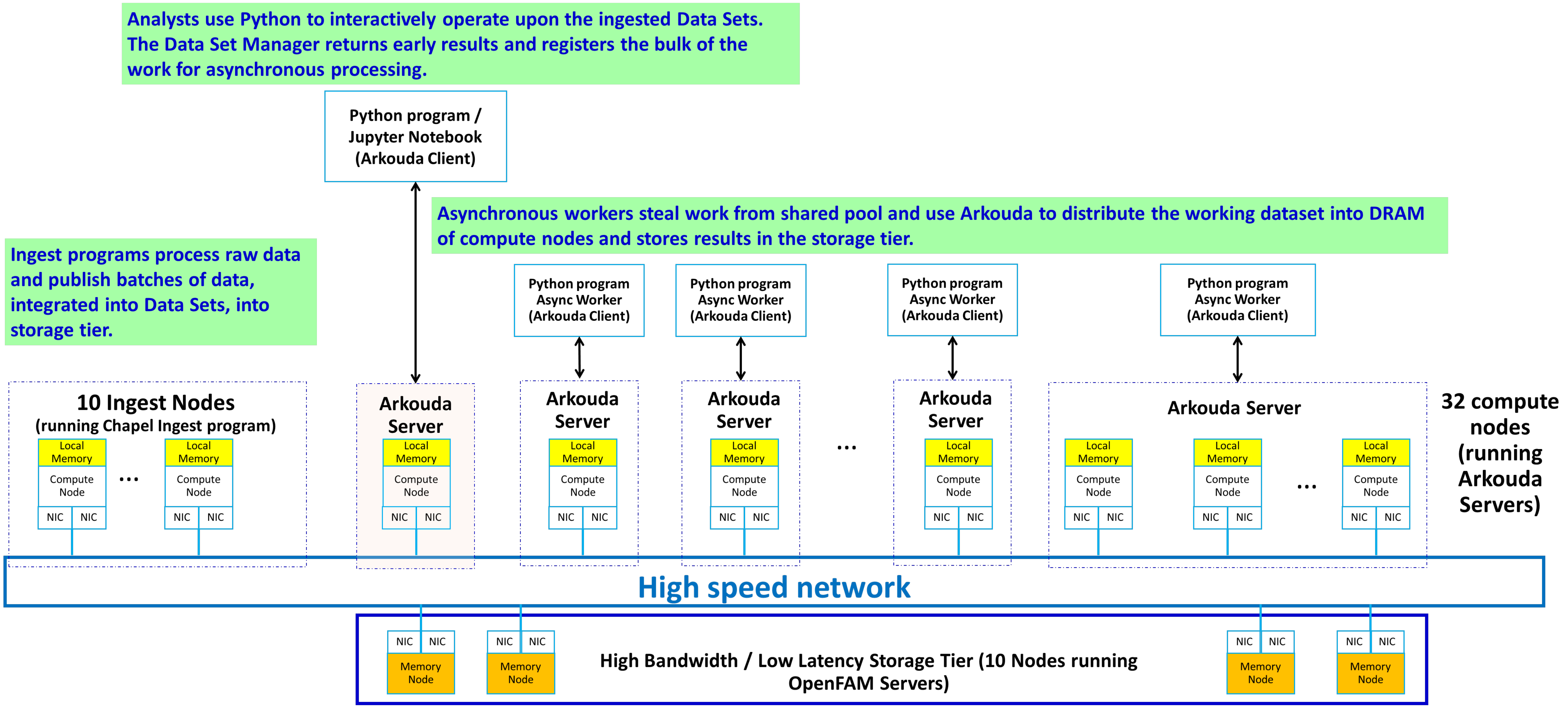
# Challenge: maximize concurrency for interactive workloads

## Sizing:

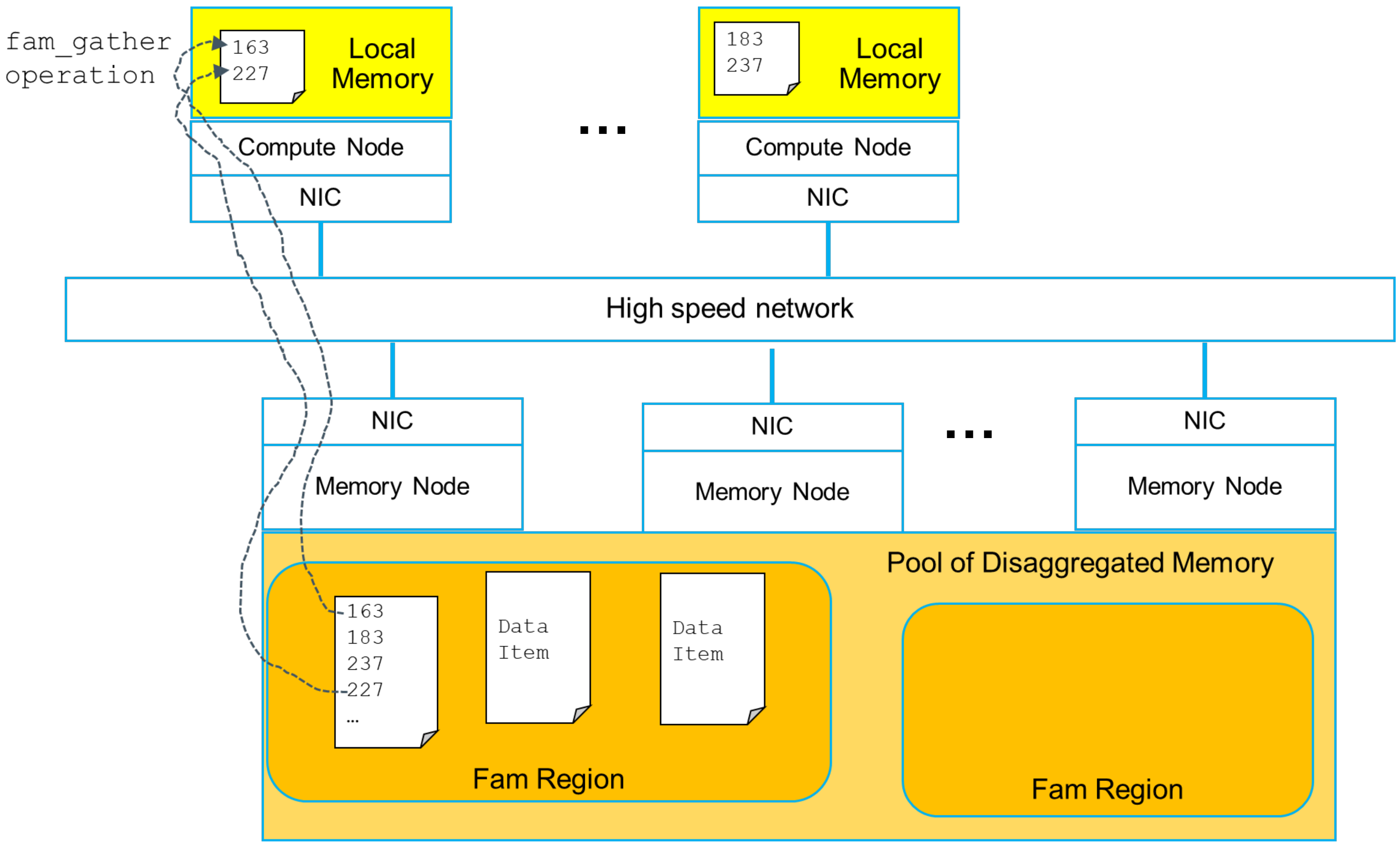
- Allocate too few nodes: could run out of resources (e.g., out of memory)
- Allocate too many nodes: could underutilize resources (e.g., low concurrency)
- Share nodes between servers: could run out of resources.



# Increase Concurrency via Batch-oriented dataset manager + Work-Stealing



# Arkouda extension for Fabric-Attached Memory (FAM)



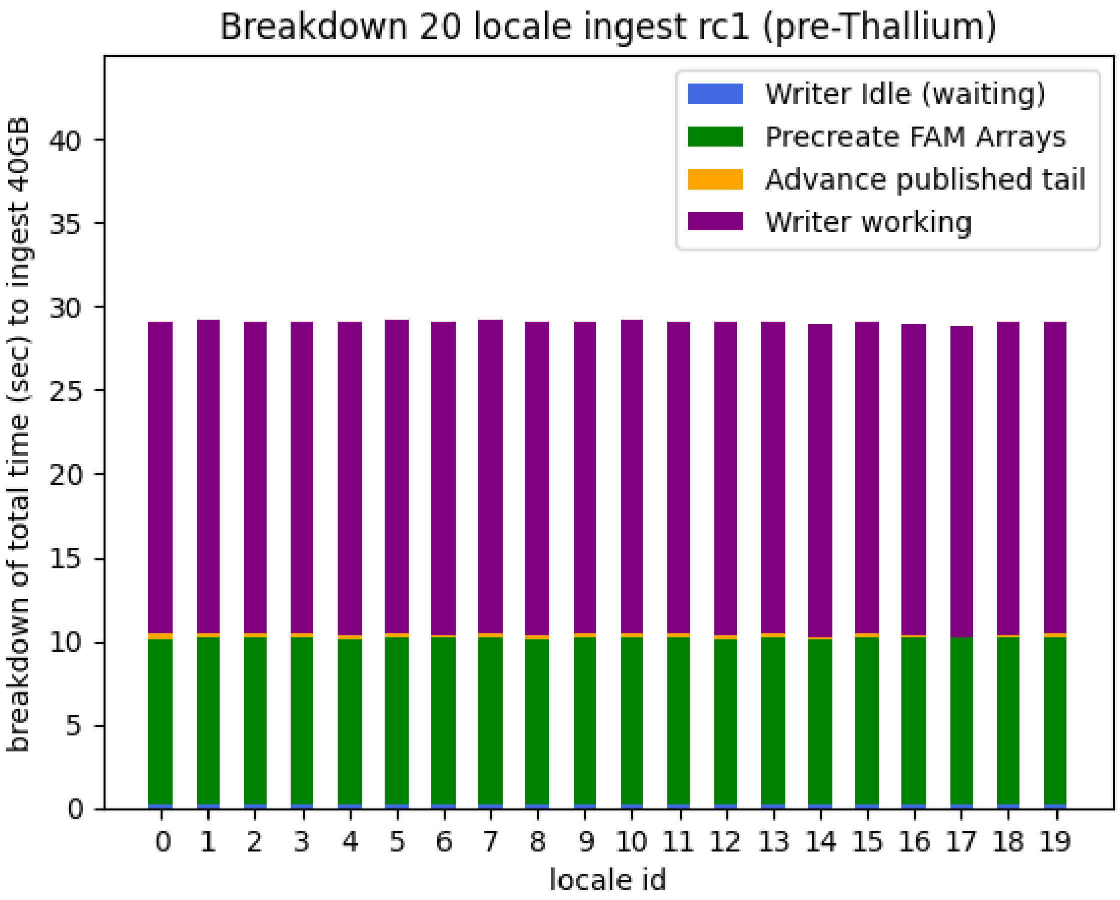

FAM bandwidth and latency are currently superior to Flash but inferior to local DRAM

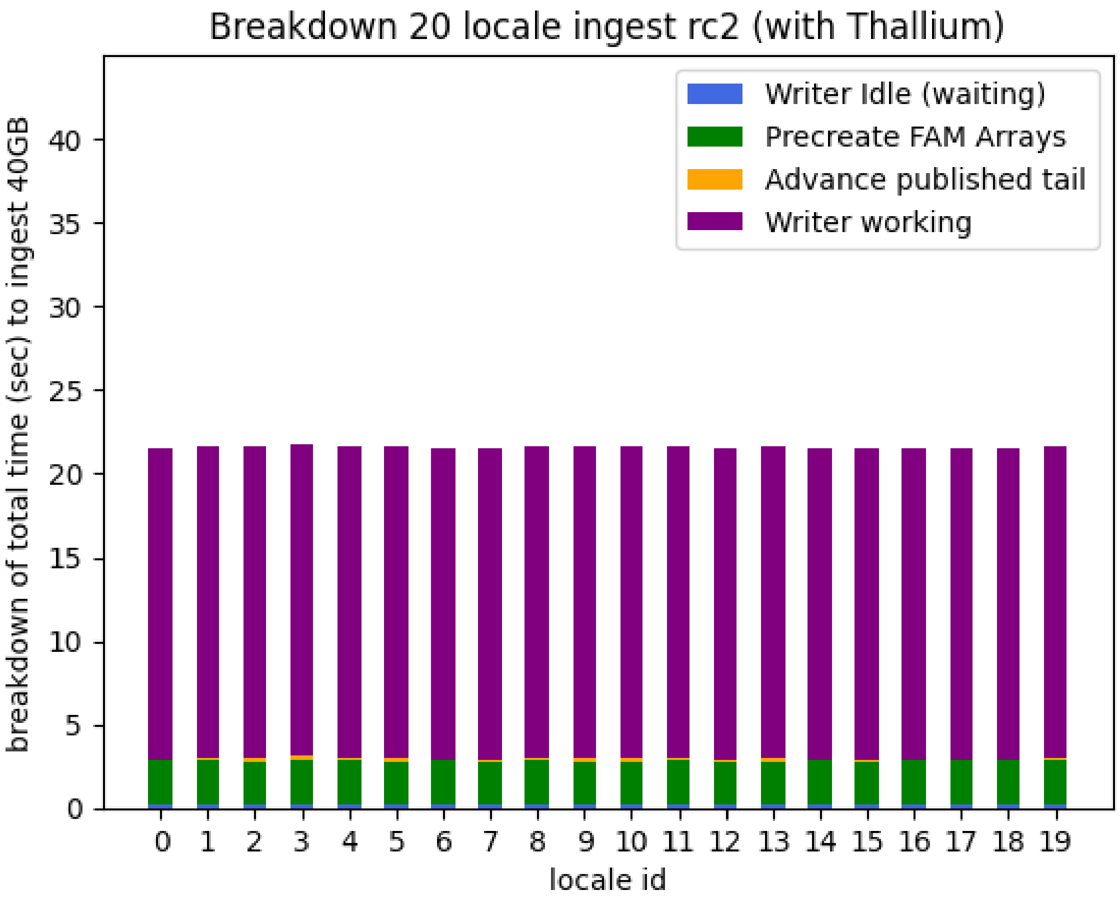
- The OpenFAM library for programming Fabric-Attached Memory lets programmers create and share in-memory data using fabric-attached memory (FAM) hosted on conventional nodes.
- OpenFAM uses RDMA for operations like put, get, scatter, gather, copy, backup, and restore, as well as standard atomic operations like fetch-and-add, compare-and-swap.
- Arkouda extension lets programmers move data between Arkouda pdarrays and OpenFAM.

# Performance Tuning

1. Dual NIC, dual socket servers: bound a locale to each socket + NIC.
2. Lock-free concurrency control while processing batches using work-stealing
3. Memory registration – registered Chapel memory with OpenFAM's Libfabric endpoints so all memory within every Chapel locale was pre-registered for RDMA.
4. Update OpenFAM to use Mochi Thallium instead of GRPC for communication between OpenFAM servers and clients.
5. Extended FAM Dataset Storage Manager to support option for registering operations as ongoing, meaning results with available data will be returned immediately and as the source data evolves, the result dataset will be updated automatically by workers.

# Ingest  8 TB (20 locales, 10 ingest nodes)

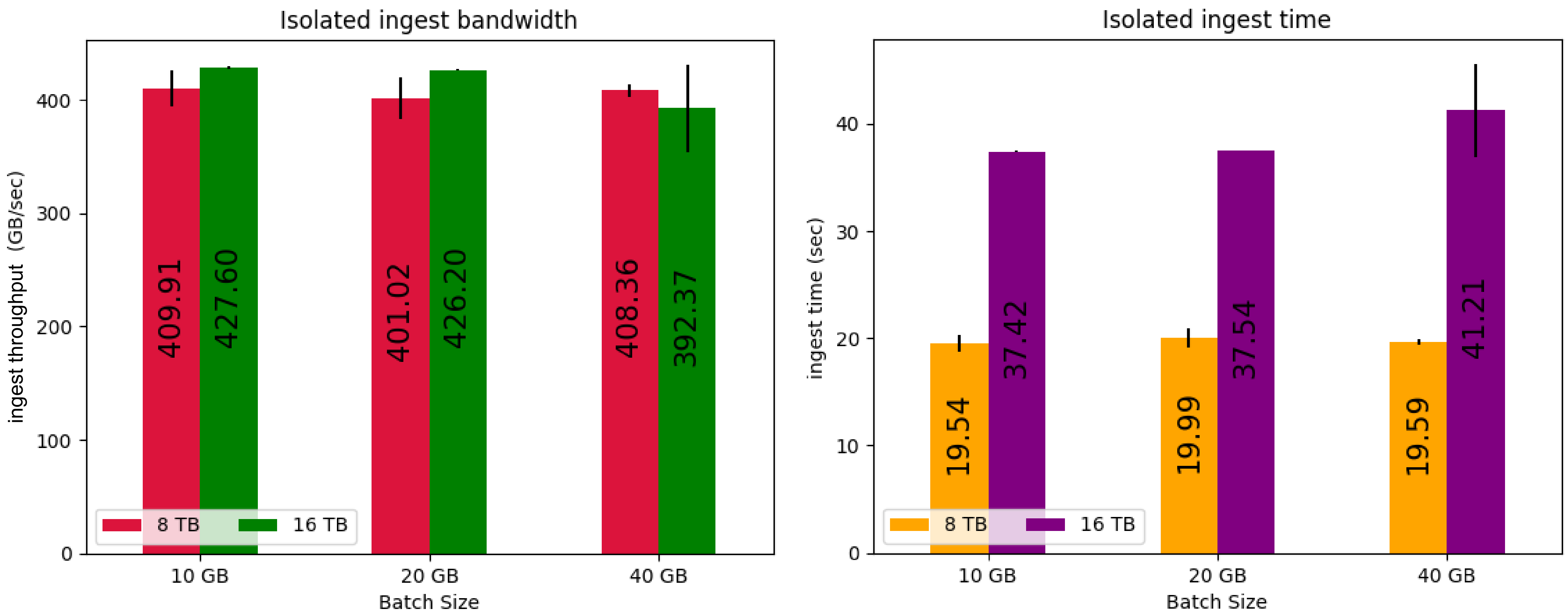



Update OpenFAM to use Mochi Thallium instead of GRPC
for communication between OpenFAM servers and clients.

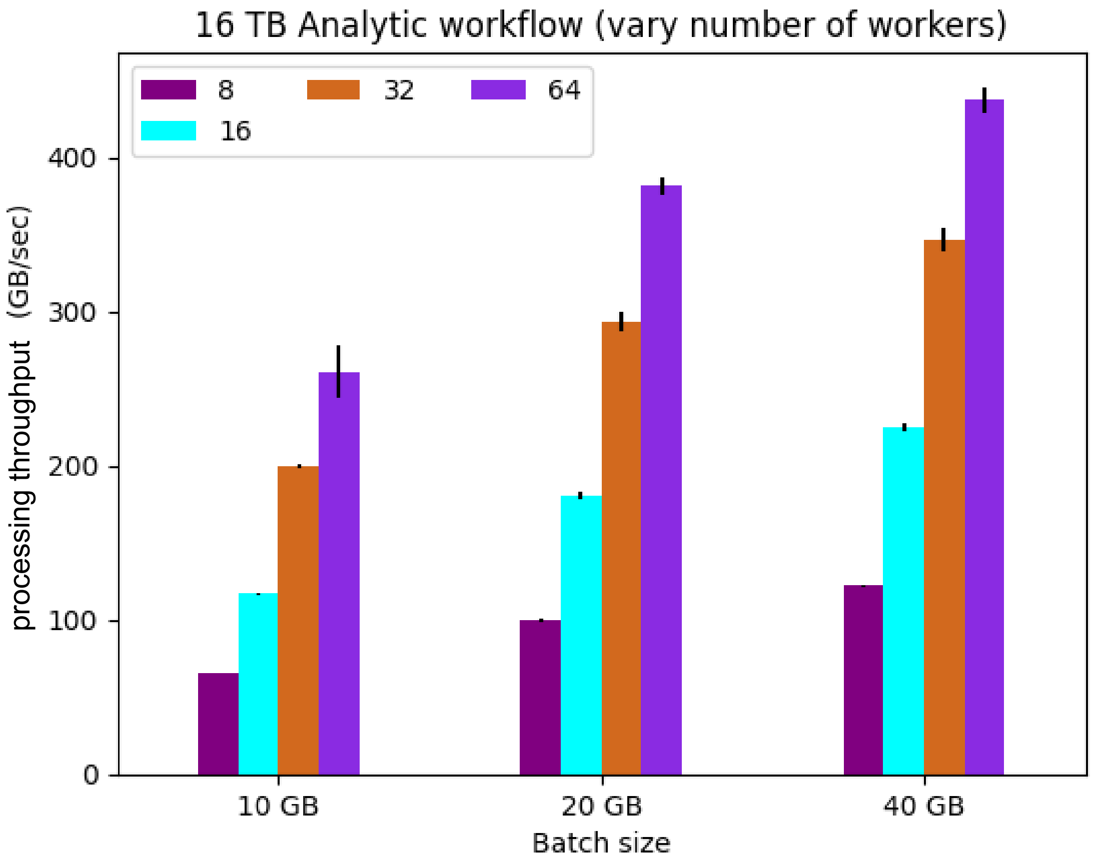
Hewlett Packard Enterprise

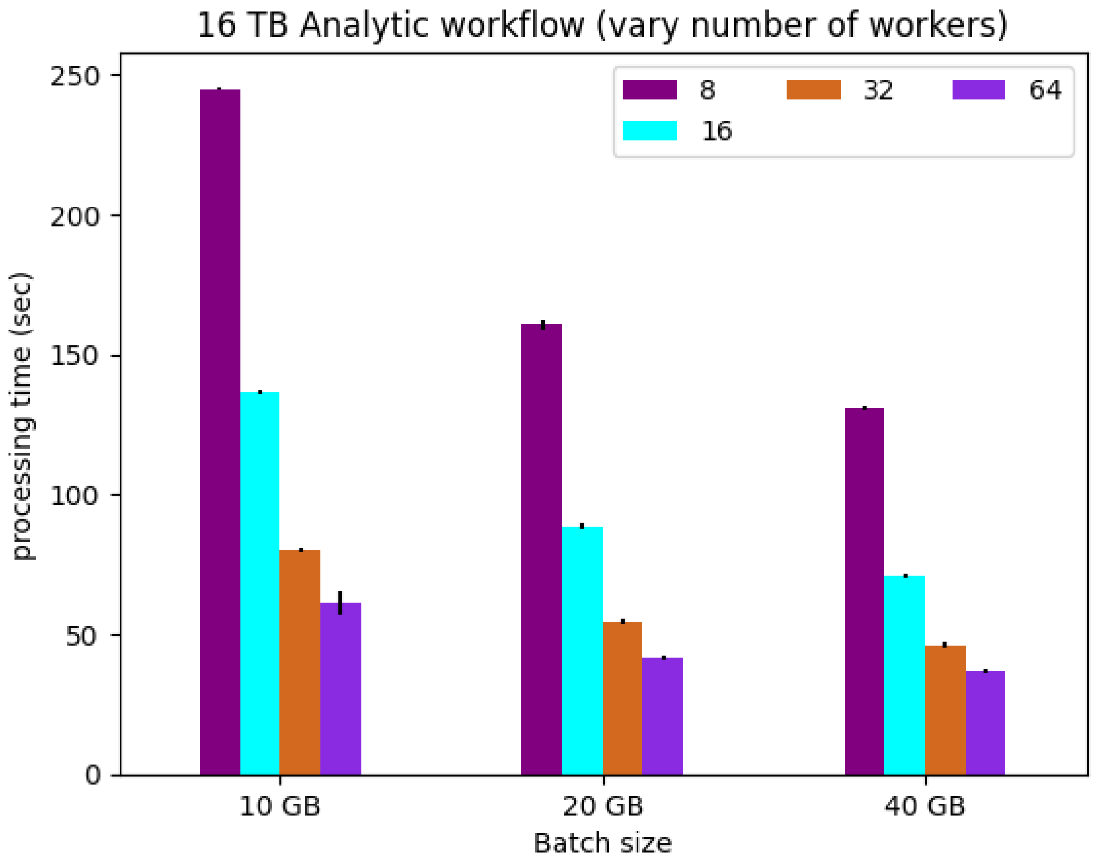
# Isolated Ingest Performance for 8 and 16 TB, varying batch size




Measured uni-directional bandwidth is about 24 GB/s per NIC: total 480 GB/s.

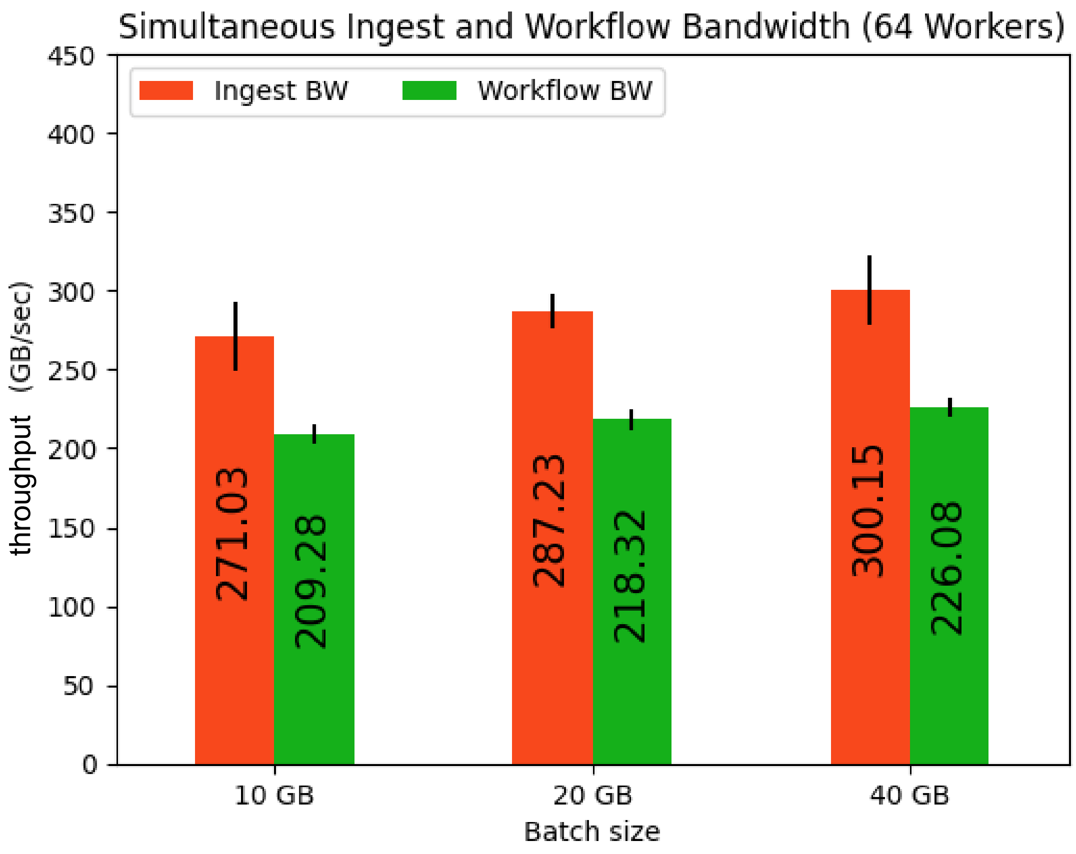
# Isolated Workflow Performance 16 TB, varying number of workers

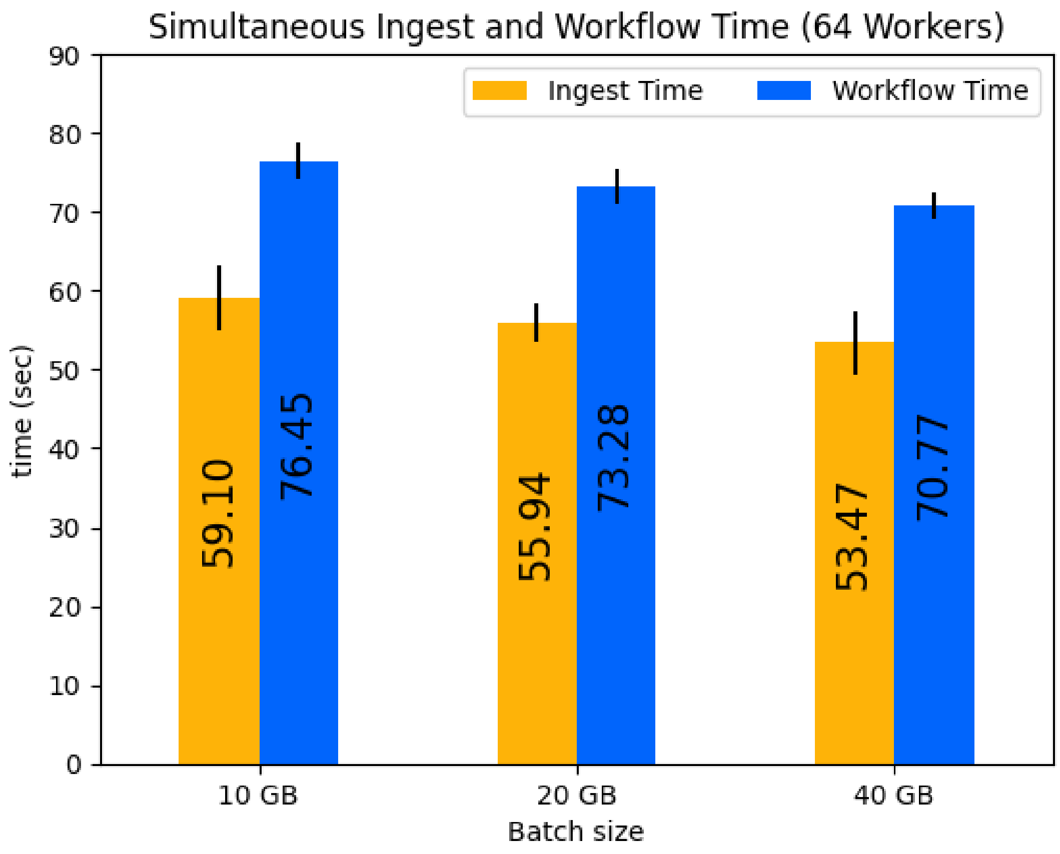



Measured uni-directional bandwidth is about 24 GB/s per NIC: total 480 GB/s.

# Simultaneous Workflow and Ingest, 16 TB, 64 workers




Measured bi-directional bandwidth is about 17 GB/s per NIC: total 340 GB/s each way.

# Trace-Driven Animation: ingest & process 16 TB

- **Experiment Setup**
  - 64 asynchronous worker Arkouda servers/clients running on 32 nodes
  - 20 ingest locales running on 10 nodes
  - 20 memory servers running on 10 nodes
  - Workflow creates 21 derived data items (derived datasets and derived columns)

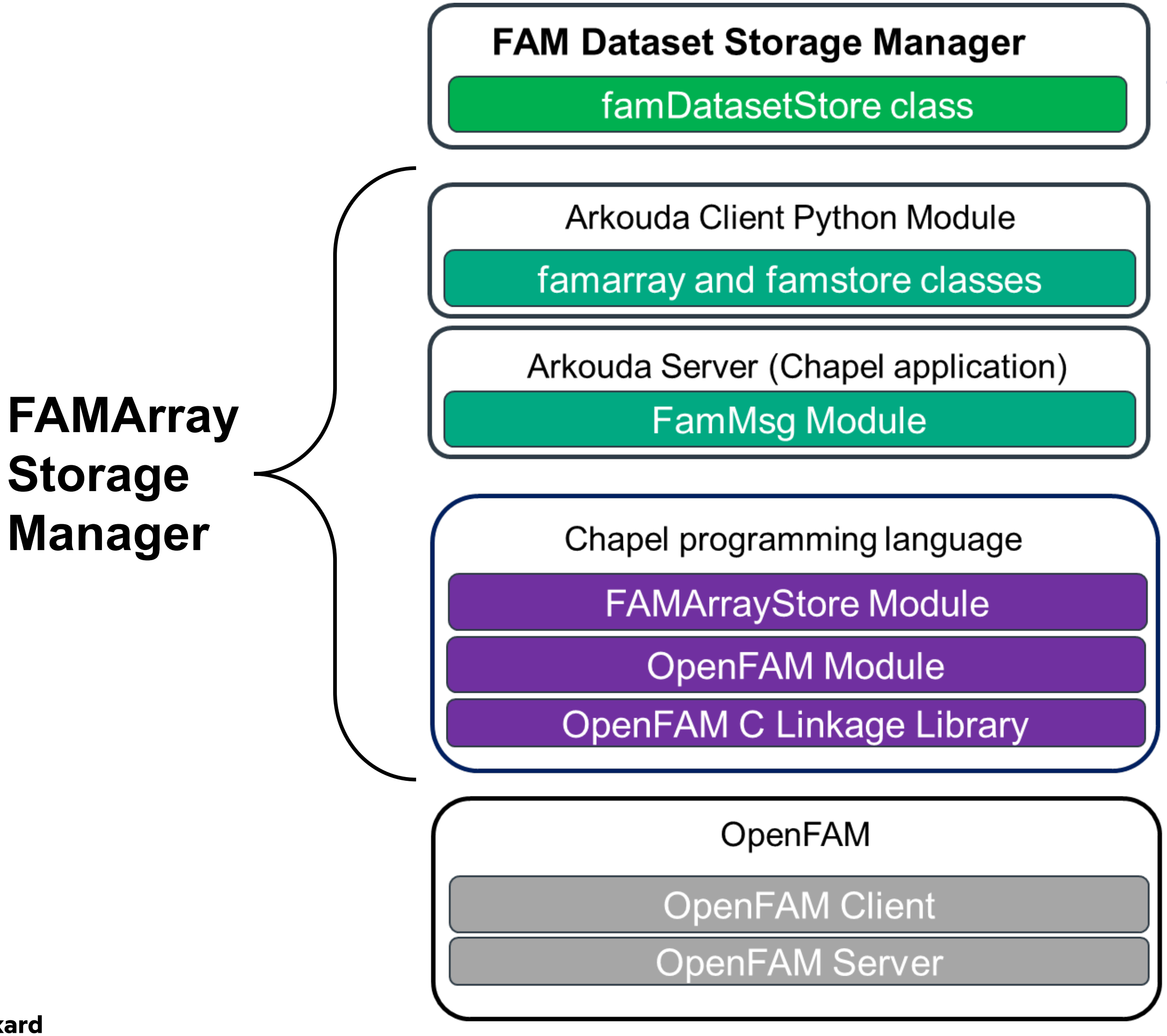
# BACKUP

# Summary of Dataset Management Approach

1. Partition incoming data into ordered discrete batches.
2. Single batch of data can be efficiently processed by an Arkouda Server running on the compute nodes.
3. Provide a Dataset Storage Manager that organizes the discrete batches of data into logical datasets (like an Arkouda/pandas dataframe).
4. The Dataset Storage Manager supports the creation of derived dataset (indexes) and derived columns.
5. The Dataset Storage Manager supports the incremental maintenance of derived datasets and derived columns. Multiple instances of the FAM Dataset Storage Manager can attach to a store and leverage each other's results.
6. To increase concurrency, we extended the FAM Dataset Storage Manager, leveraging its support for incremental maintenance to implement work-stealing functionality.

When a data analyst explores a large dataset in an interactive session by deriving new datasets and columns, they can request that the Dataset Storage Manager present them with early results and delegate completion of the new datasets to a multitude of Arkouda Servers that will complete the work.

# FAM Dataset Storage Manager for Arkouda




Operations (e.g., filter, scatter, gather, sortruns, topXperrun, toPdarray, toDataFrame) on FAM Datasets.

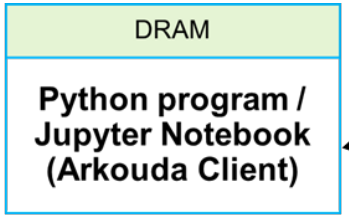
Over-the-wire messages that let Arkouda clients request that the Arkouda server perform operations on arrays of data residing in FAM.

Movement of data between Arkouda pdarrays and named arrays of data stored in FAM. Store FAM objects in Arkouda Symbol Table.

Multi-locale put, get, gather, scatter of data stored in FAM.

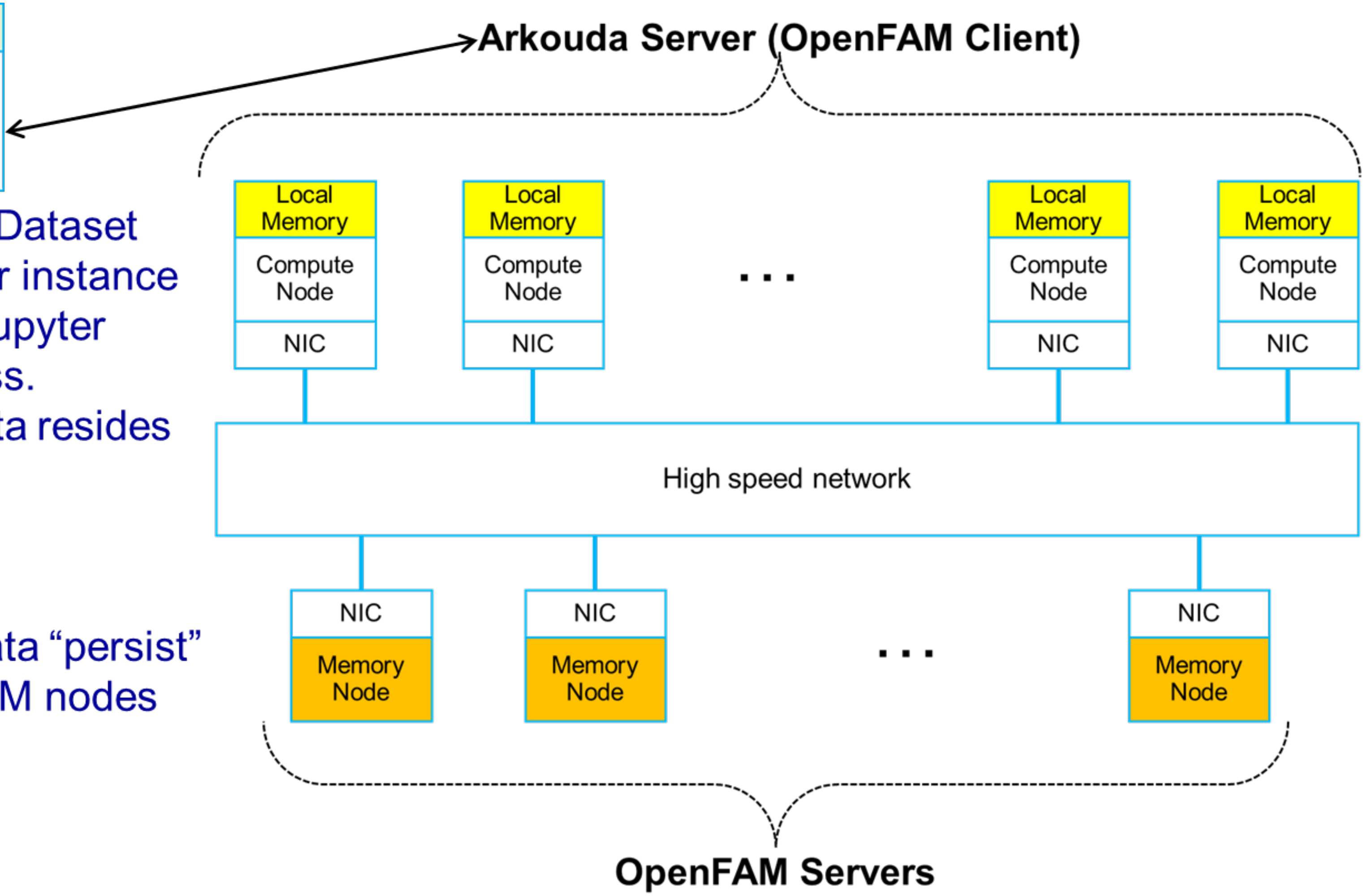
OpenFAM API for single-locale operations on data stored in FAM

# FAM Dataset Storage Manager



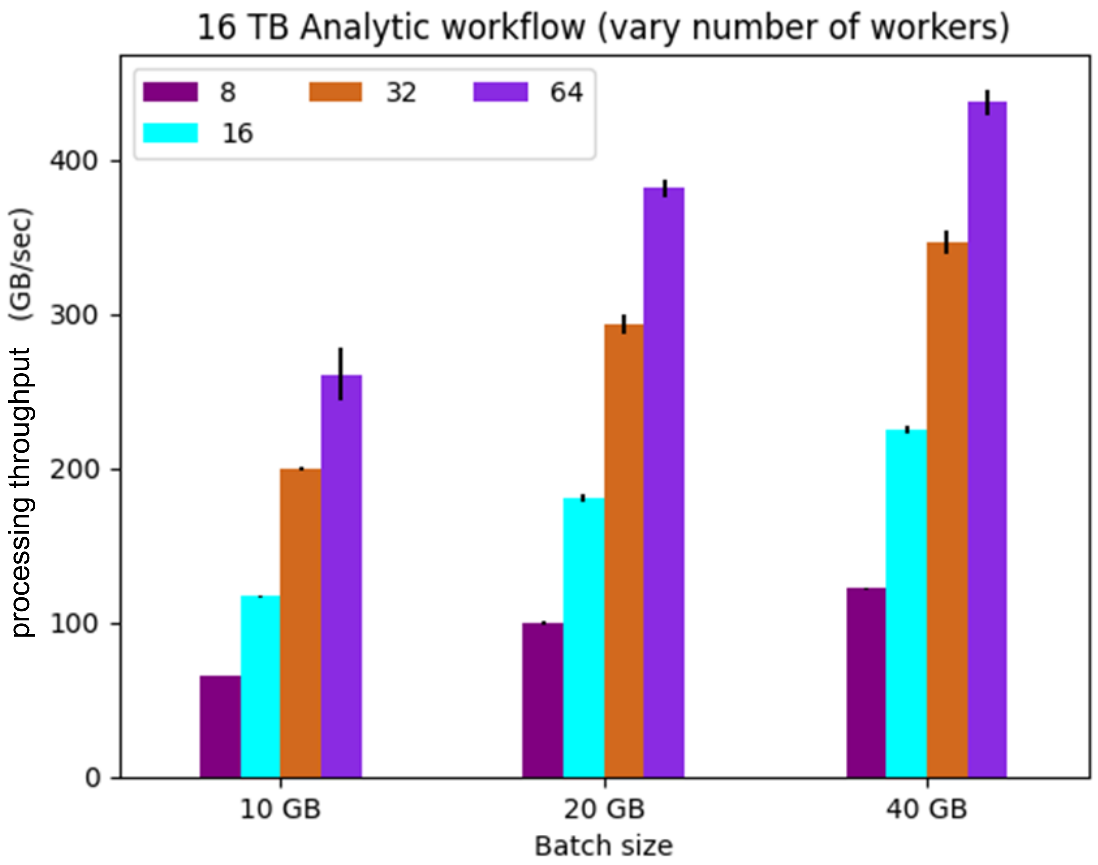

A transient FAM Dataset Storage Manager instance runs within the Jupyter Notebook process. Working metadata resides in memory here.

Data and metadata "persist" in memory on FAM nodes as FAM Arrays.




Within the Arkouda server processes, the FAM Dataset Storage Manager "pages" batches of data between famarrays and Arkouda parallel distributed arrays (pdarrays). Working data resides in the local memory of compute nodes.

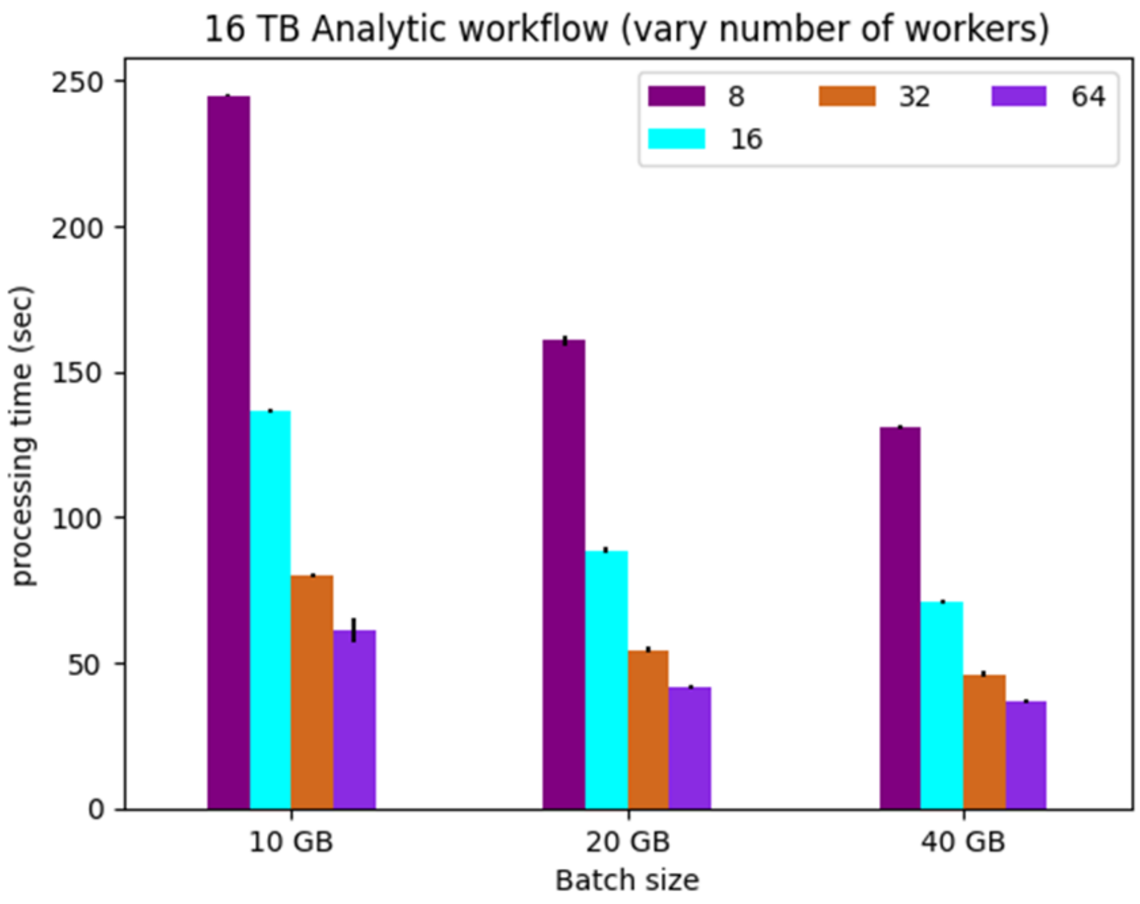
# Isolated workflow throughput, 16 TB




Batch Size

| number of workers | 10 GB | 20 GB | 40 GB |
|---|---|---|---|
| 8 | 65.2975 | 99.5303 | 122.0058 |
| 16 | 117.1965 | 180.805 | 225.0644 |
| 32 | 199.3732 | 293.4397 | 346.6944 |
| 64 | 260.8268 | 381.7748 | 437.3626 |

# Isolated workflow latency, 16 TB




**Batch Size**

| number of workers | 10 GB | 20 GB | 40 GB |
|---|---|---|---|
| **8** | 245 | 160.8 | 131.138 |
| **16** | 136.5 | 88.49 | 71.089 |
| **32** | 80.25 | 54.53 | 46.149 |
| **64** | 61.34 | 41.91 | 36.582 |