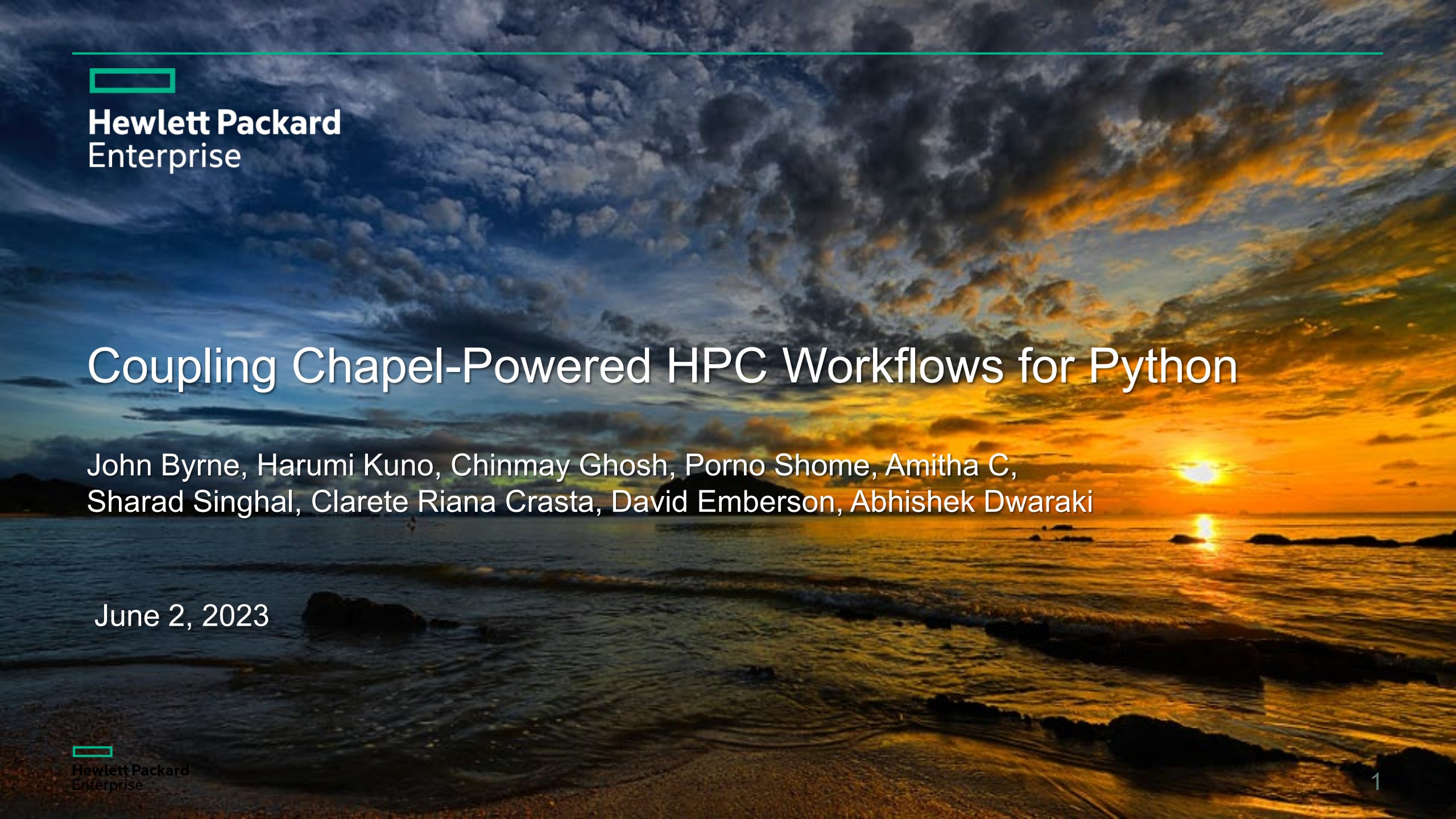
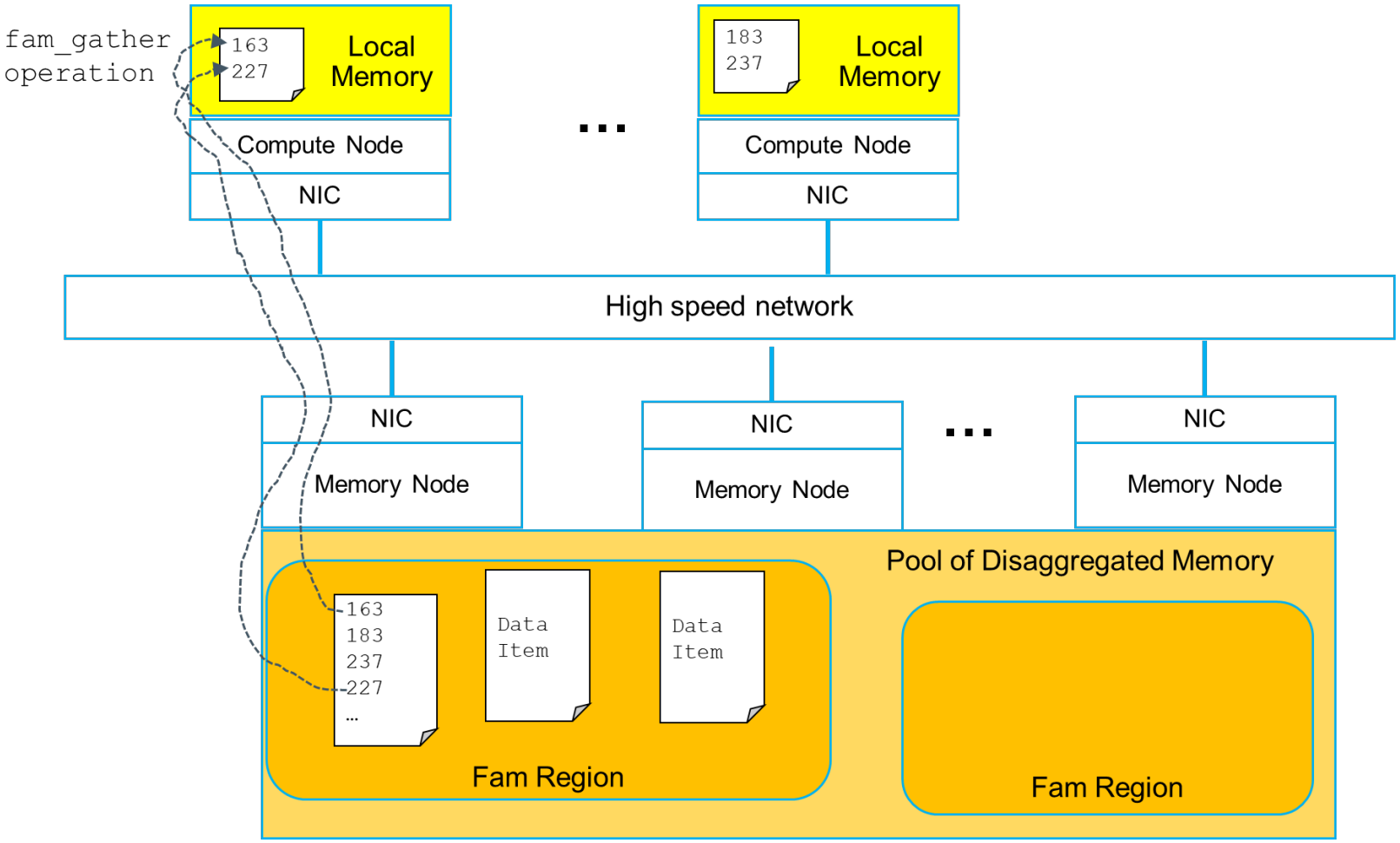


Coupling Chapel-Powered HPC Workflows for Python

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

June 2, 2023

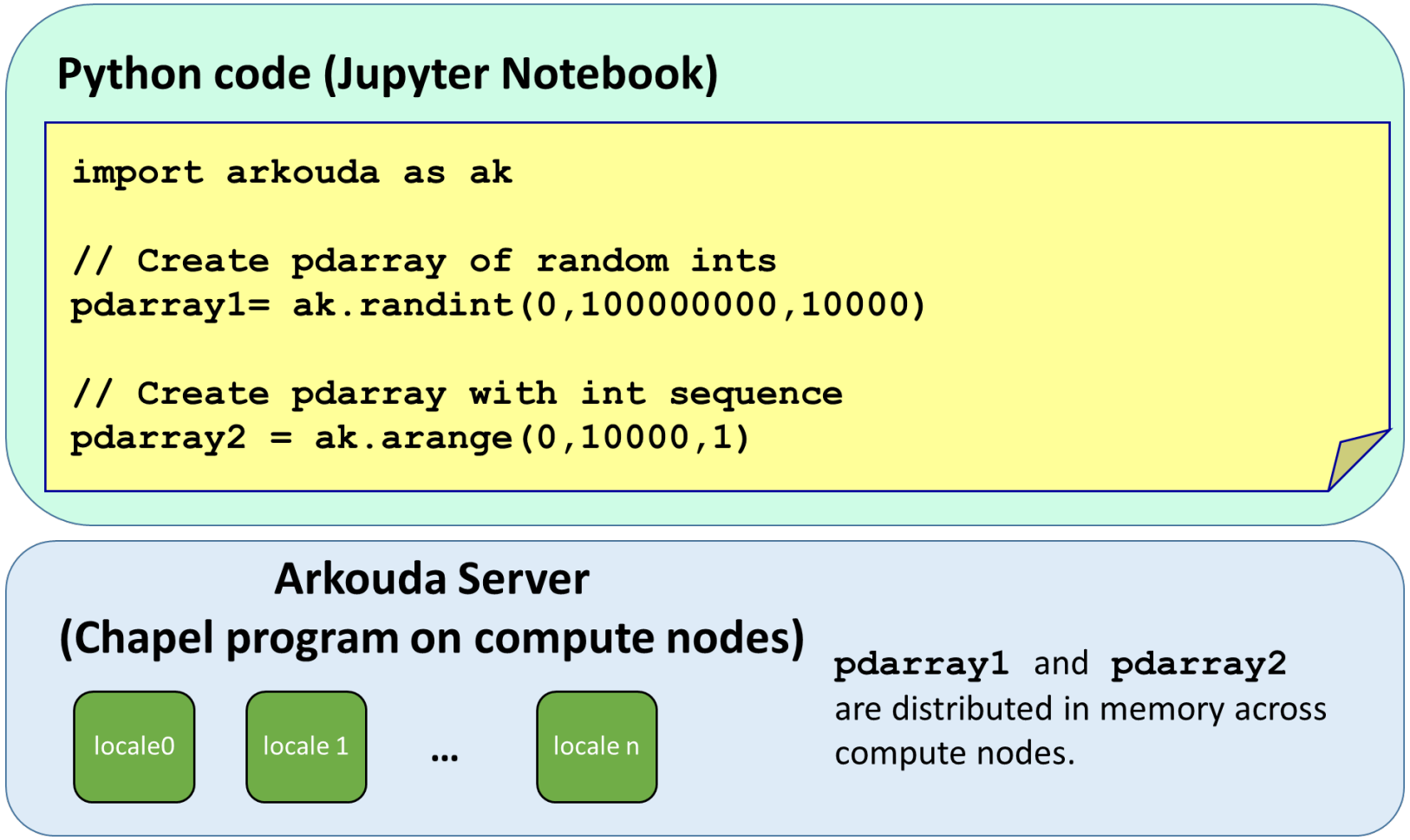
Fabric-Attached Memory (FAM)



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

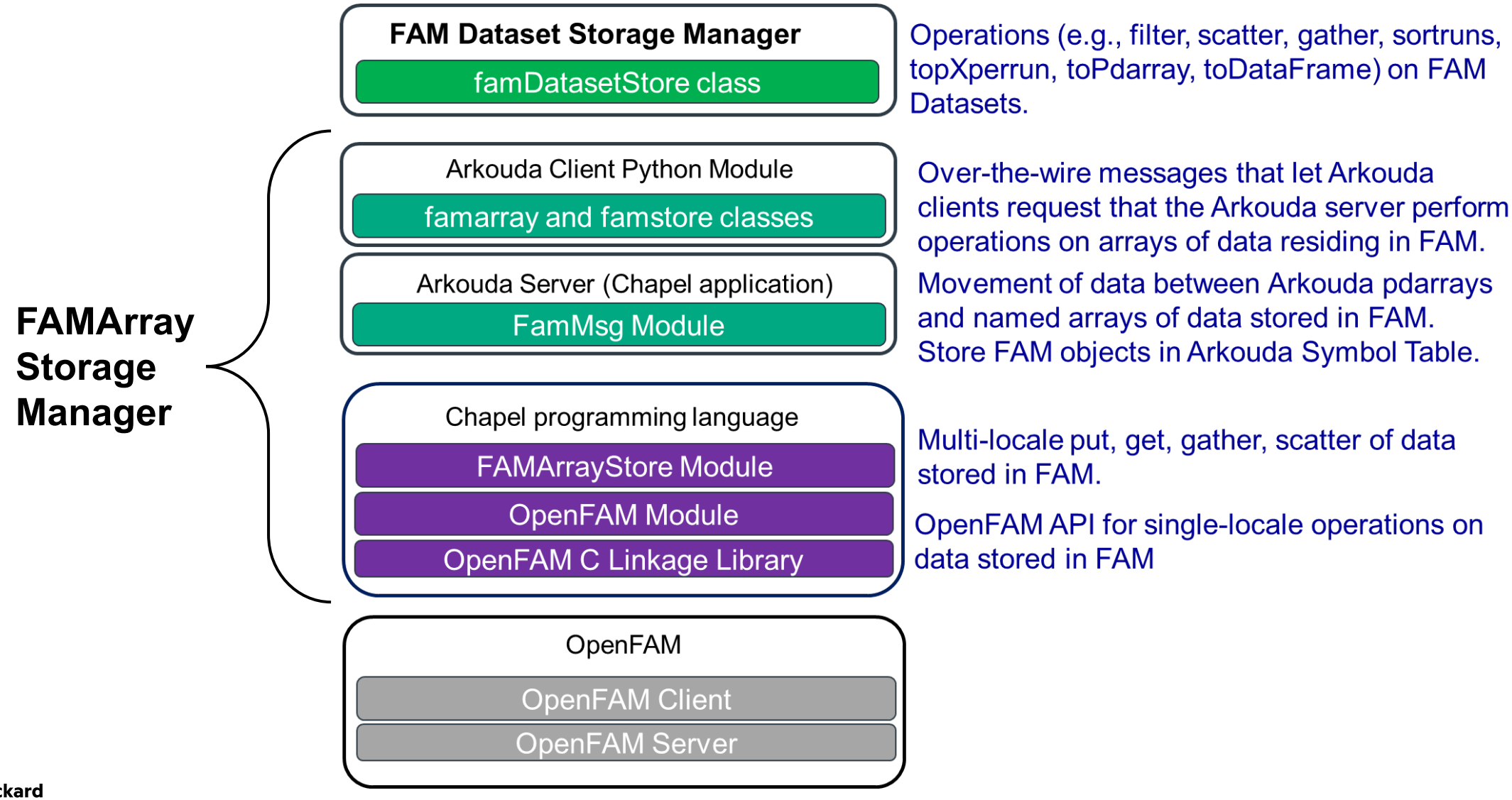
- The OpenFAM library for programming Fabric-Attached Memory supports shared pools of fabric-attached memory (FAM) hosted on conventional nodes.
- The OpenFAM API lets programmers create and share regions of FAM, and also data items within regions.
- OpenFAM uses RDMA to support operations like put, get, scatter, gather, copy, backup, and restore, as well as standard atomic operations such as fetch-and-add, compare-and-swap.

Arkouda lets Python programmers work interactively with data residing in the memory of compute nodes

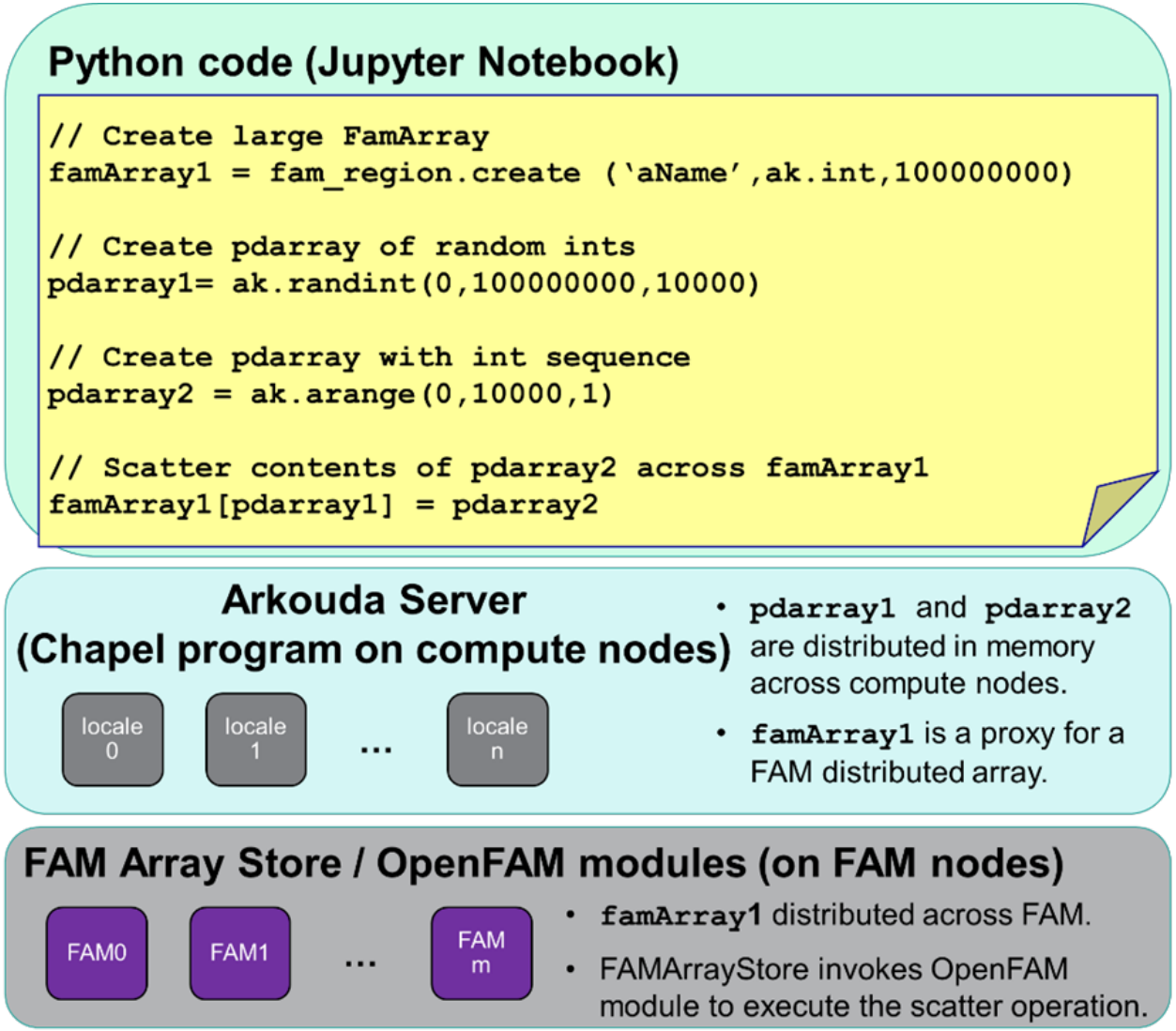


For each line of code, the Python program, acting as an Arkouda client, sends commands and metadata over the wire to the Arkouda server running on the compute nodes.

FAMArray Storage Manager



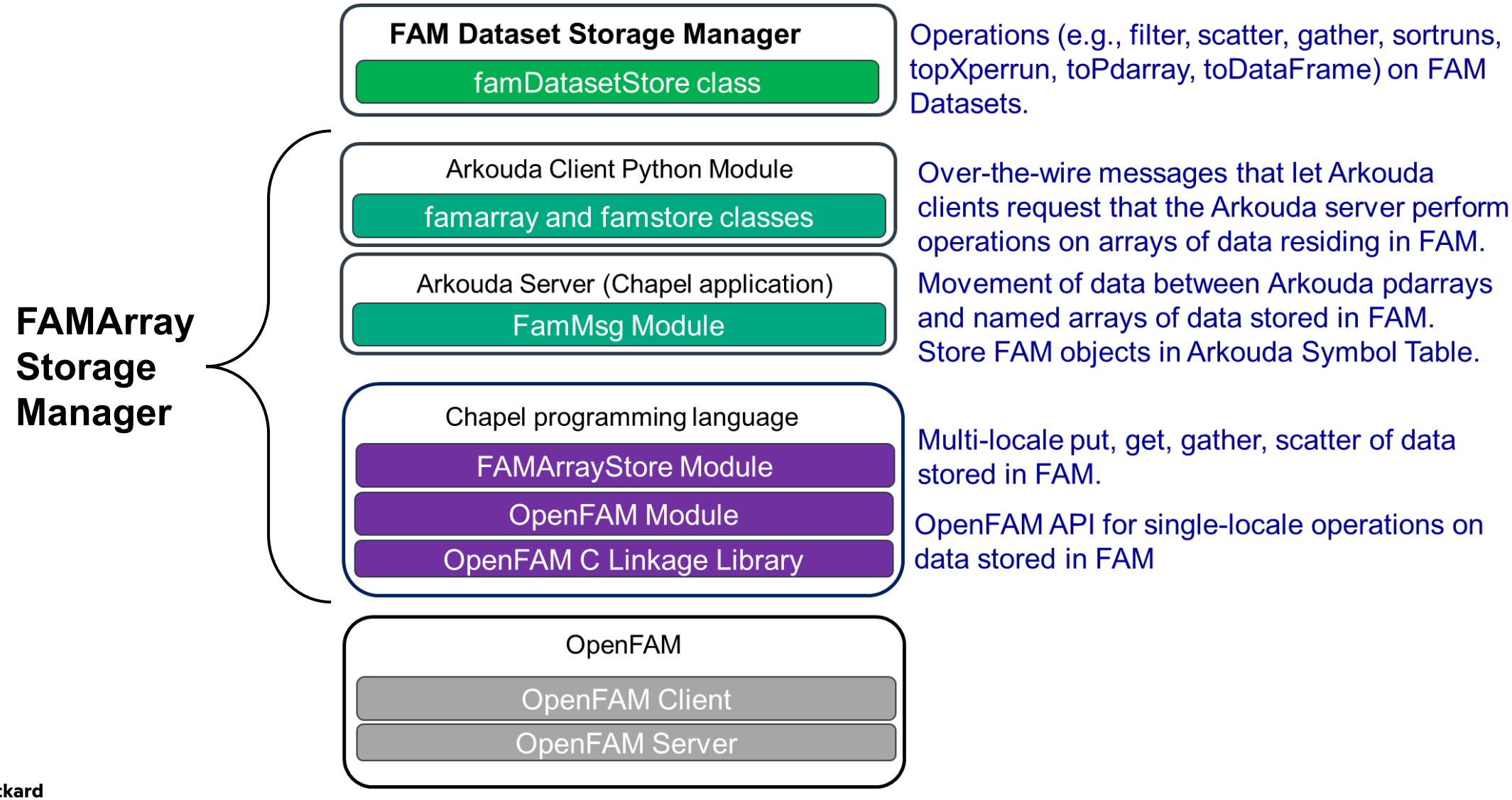
FAMArray Storage Manager for Chapel and Arkouda



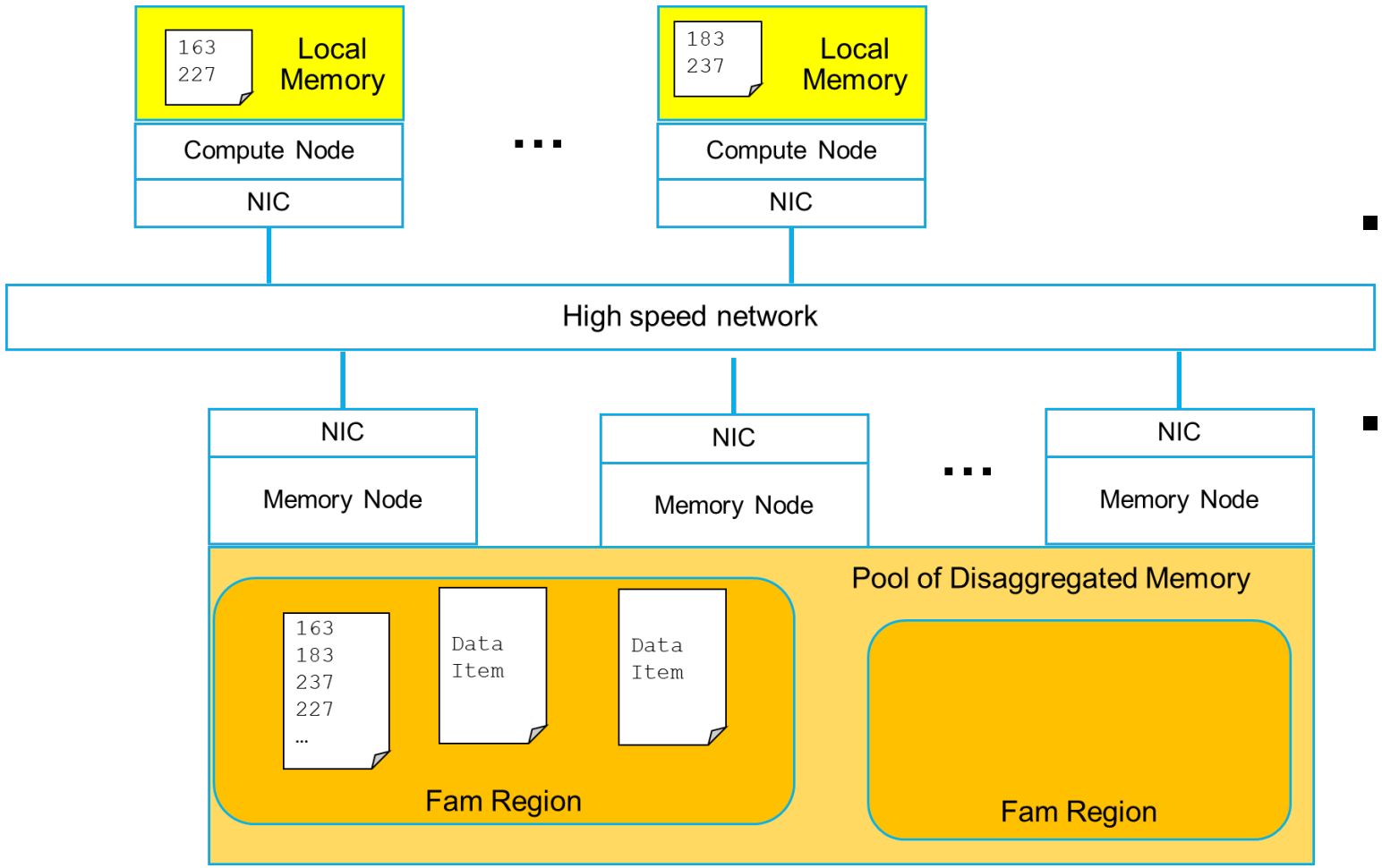
For each line of code, the Python program, acting as an Arkouda client, sends commands and metadata over the wire to the Arkouda server running on the compute nodes.

The Arkouda server, acting as a FAM client, uses the FAM Array Store to explode the **`famArray1[pdarray1] = pdarray2`** assignment into scatter operations executed on each locale for the subset of **`pdarray1`** and **`pdarray2`** located on that locale.

FAM Dataset Storage Manager

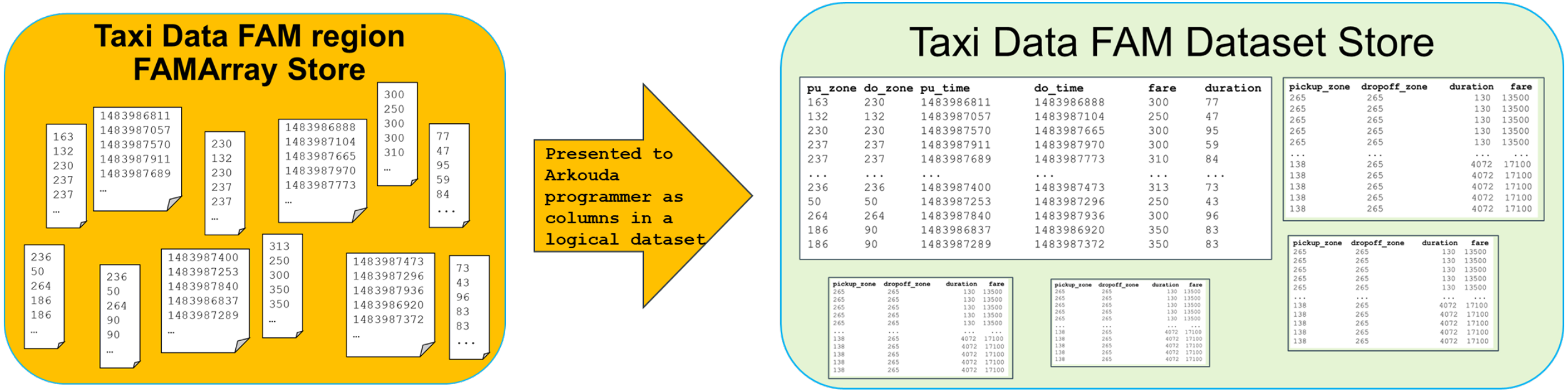


Getting the most from FAM + Arkouda

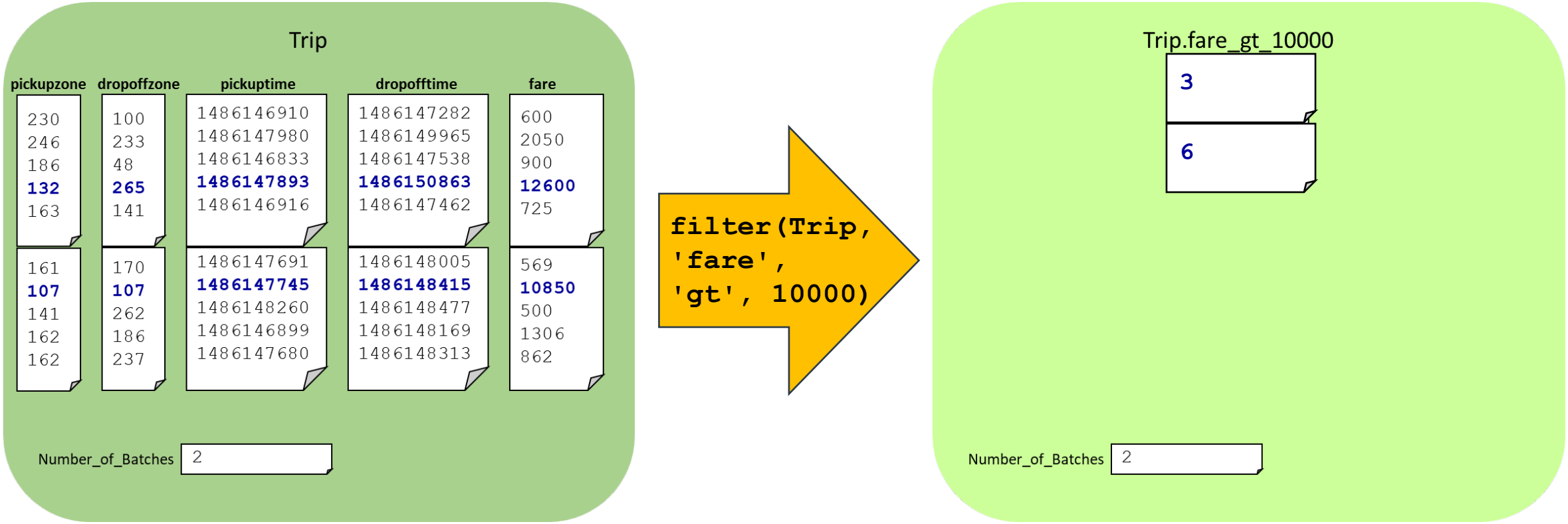


- Enable Python programmers to solve problems using datasets too large to fit in the memory of compute nodes.
- Remote memory has lower bandwidth and higher latency than local DRAM.
- Move data between the FAM pool and the local memory of the compute nodes such that:
  - Data is placed in memory that is “close” to the processors that will operate upon that data.
  - Computations operate on distinct subsets of program data.

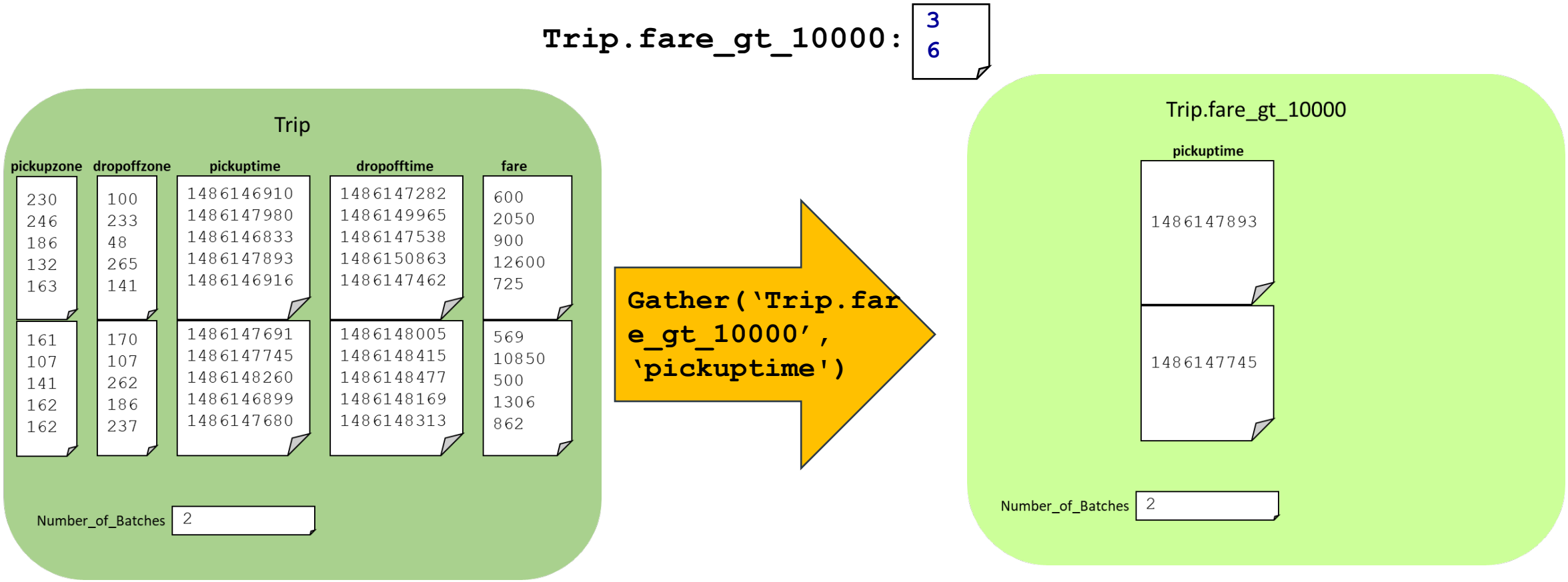
Store data in FAM in batches, as discrete arrays. Present data to the programmer as integrated Datasets.



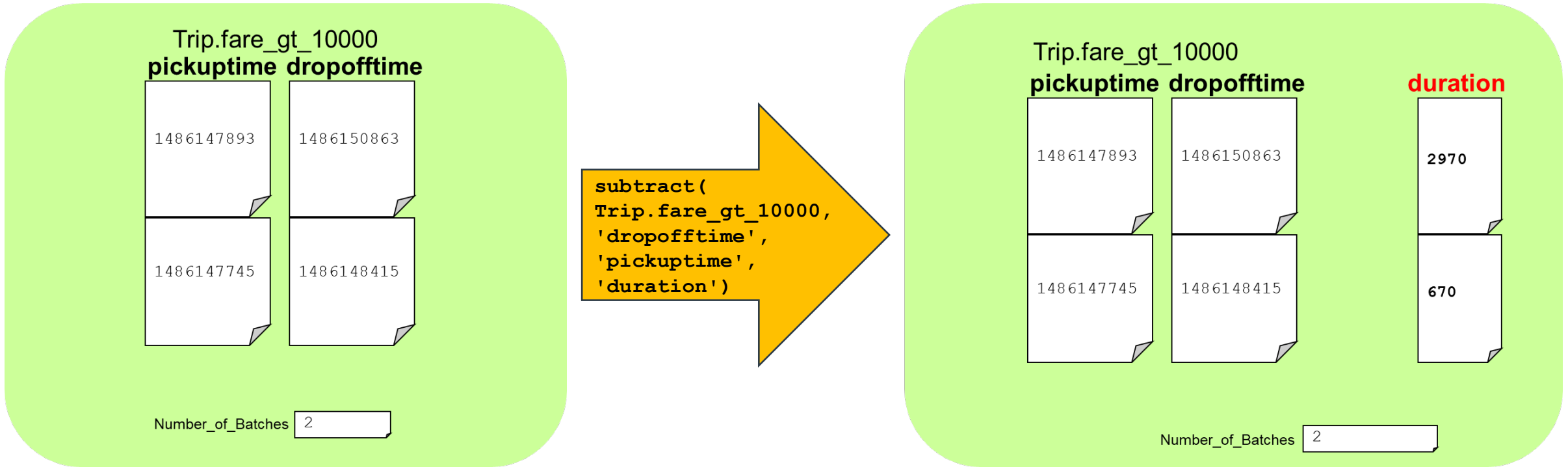
Derived FAM Datasets represent indices into base data



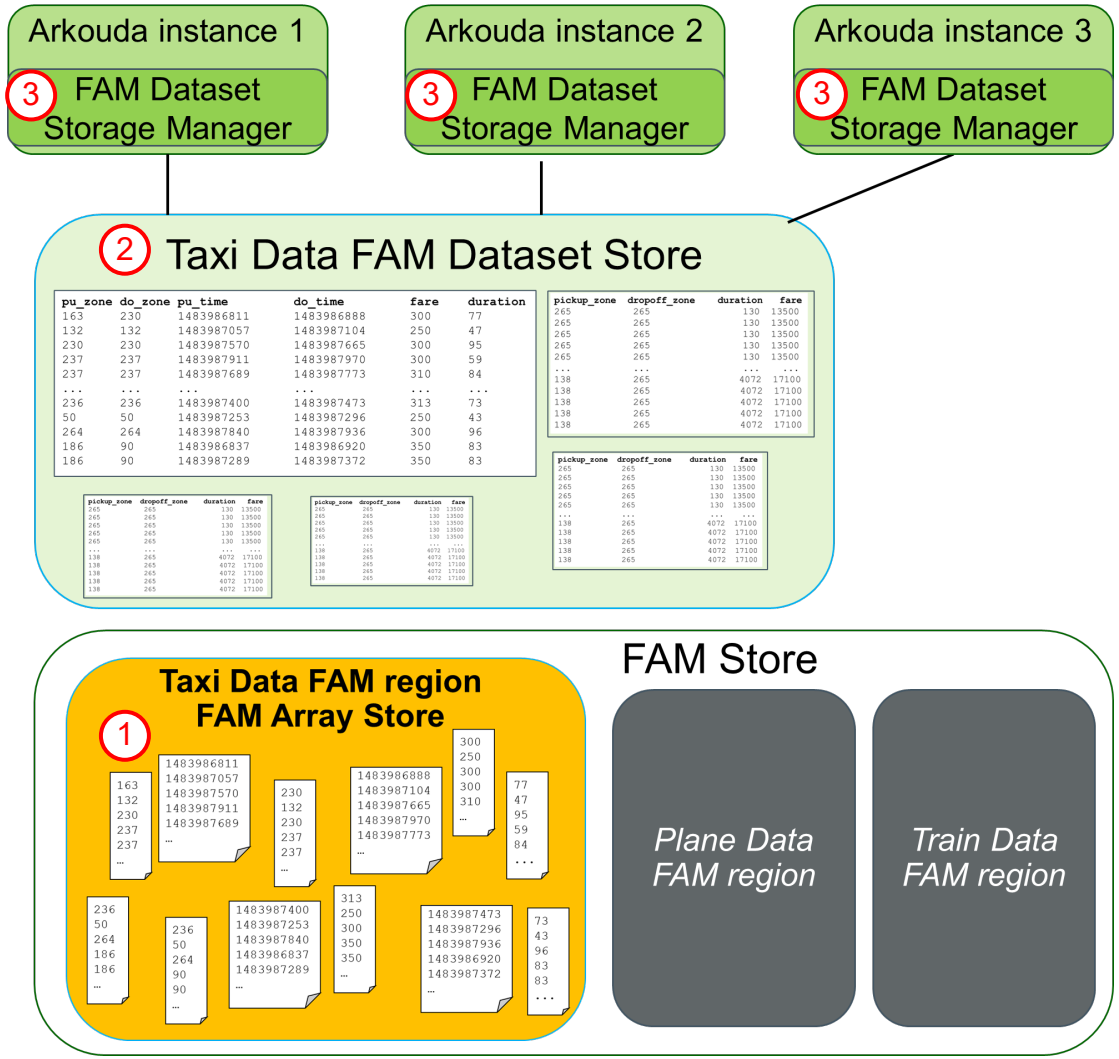
Can use the index to gather columns of interest



Some operations produce derived columns



FAM Dataset Storage Manager

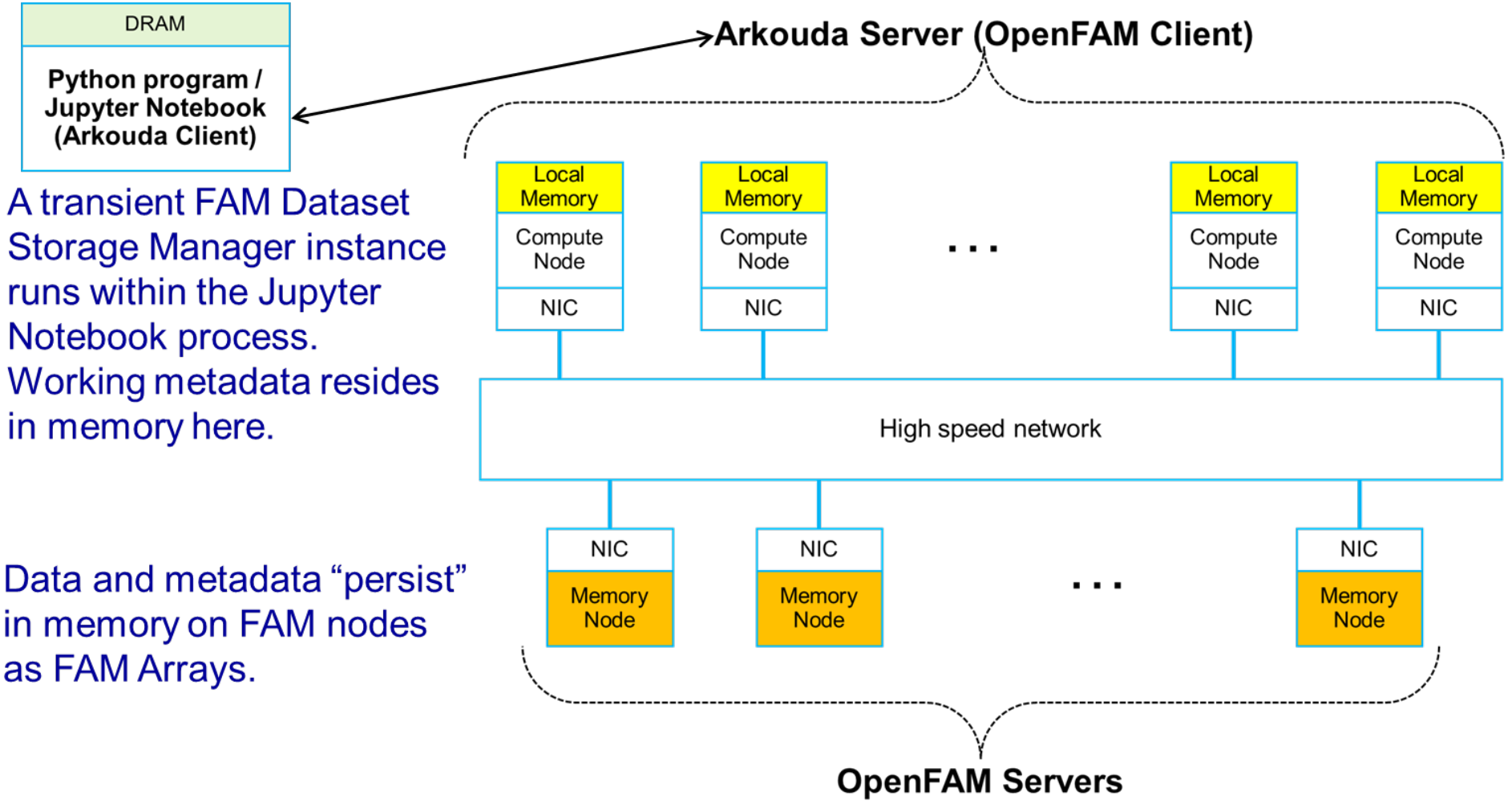


1. Ingested data is stored in famarrays in a FAM Array Store.
2. The FAM Dataset Storage Manager presents integrated views of related base and derived data items as a collection of FAM Datasets.
3. Multiple Arkouda processes can attach to the same FAM Dataset Store, sharing FAM data while maintaining their own symbol tables and internal working data sets.

Data is managed in terms of ordered batches, so it can be incrementally processed, and so that previous results can be leveraged to speed time-to-results for future results.

Derived datasets could be updated using an automatic update mechanism as new batches of data are ingested.

Working vs. Shared Data and Metadata

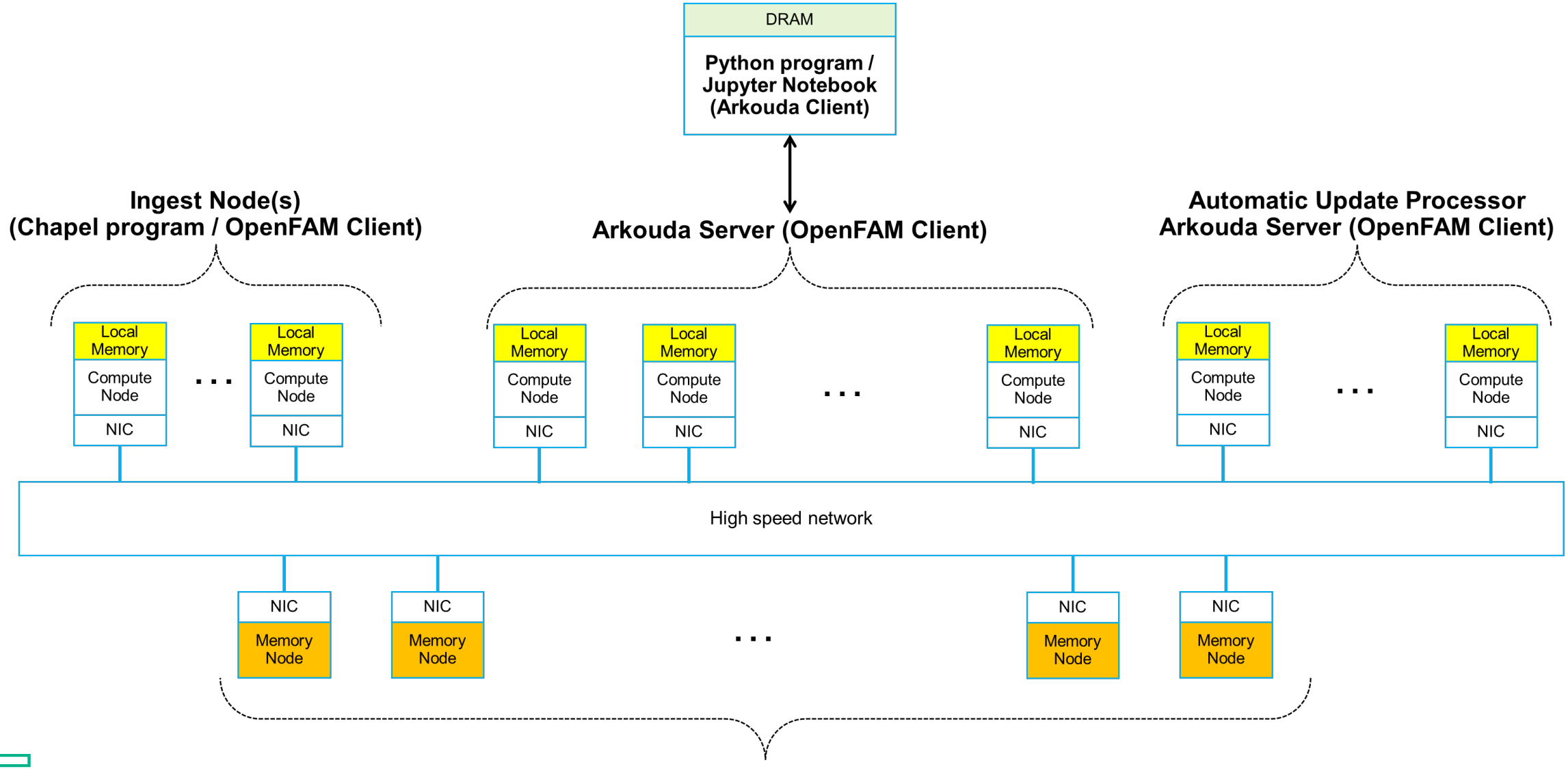


A transient FAM Dataset Storage Manager instance runs within the Jupyter Notebook process. Working metadata resides in memory here.

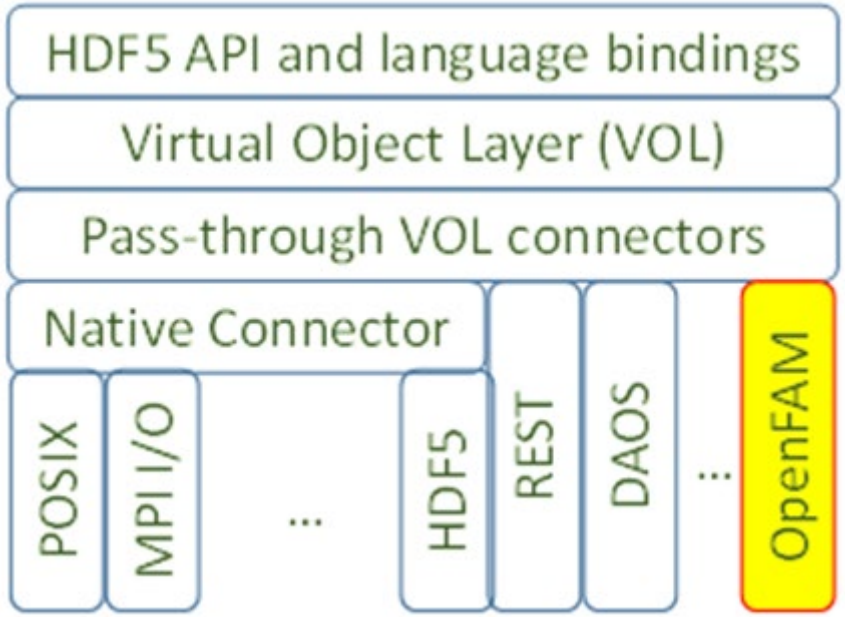
Data and metadata "persist" in memory on FAM nodes as FAM Arrays.

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.

Arkouda and Chapel programs can use FAM to share results



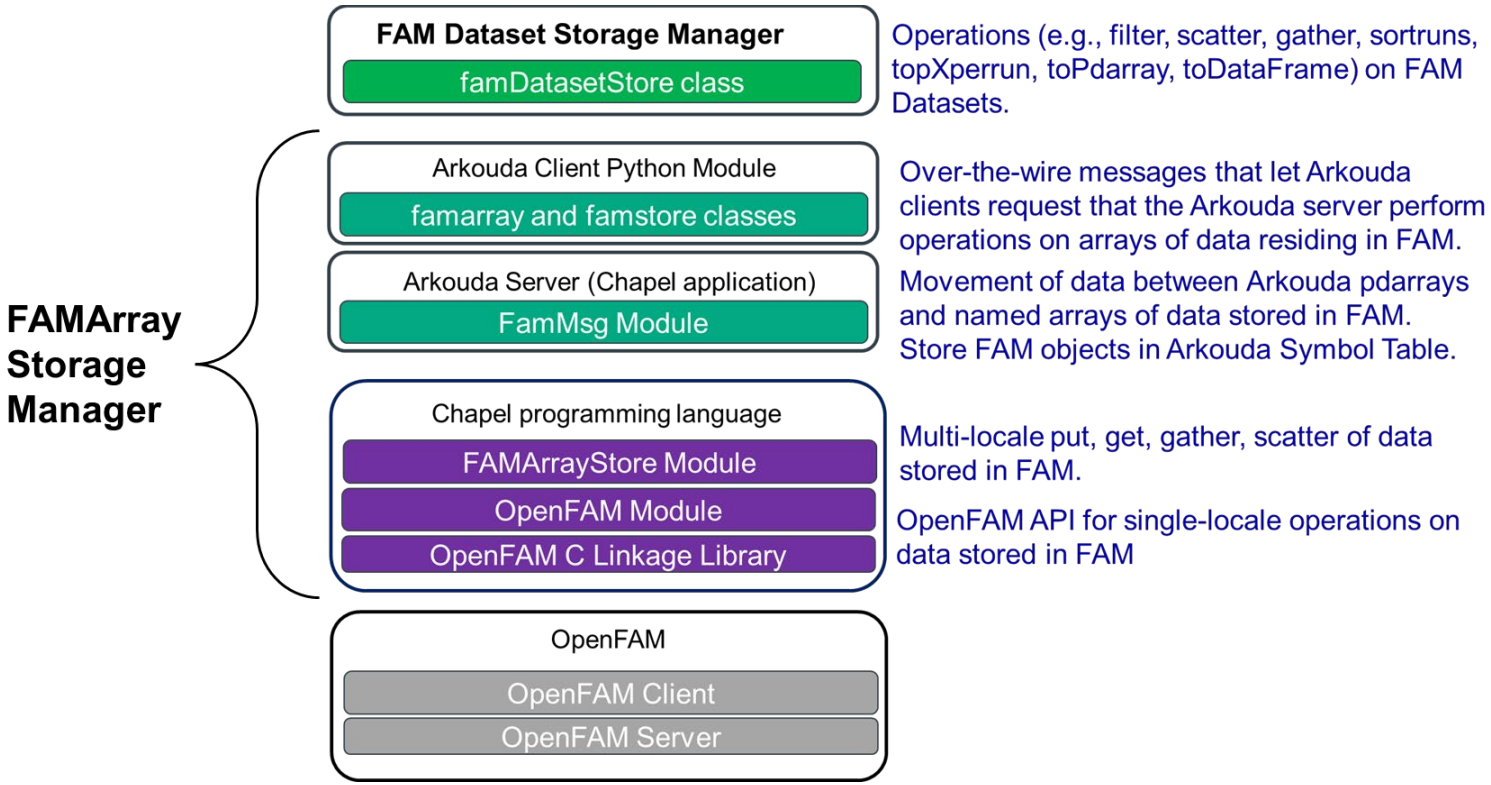
In addition, we also provide an HDF5 interface for FAM



This figure is based on a drawing from slide 31 of:
M. Scot Breitenfeld, Elena Pourman, Suren Byna, and Quincey Koziol. 2020
Achieving High Performance I/O with HDF5.
HDF5 Tutorial ECP Annual Meeting 20202.
https://www.hdfgroup.org/wp-content/uploads/2020/02/20200206_ECPTutorial-final.pdf

- Hierarchical Data Format version 5 (HDF5) is an extensible data model/specification (backed by a library of open source software) that makes it easier to organize and share large, complex, heterogenous data.
- It works with a variety of backing stores, such as POSIX, DAOS, and AWS S3.
- The OpenFAM connector for HDF5 maps FAM storage into the HDF5 data model through the VOL layer.
- This connector enables applications, regardless of programming language, to read/write HDF5 datasets on FAM.

Take-Aways and Next Steps



Take-Aways

- Chapel does a very nice job translating array operations into parallel computation, and Arkouda brings this to Python programmers.
- FAMArray extensions to Chapel and Arkouda help Chapel and Python programmers work with disaggregated memory when using a compute cluster.
- FAM Dataset Storage Manager leverages these extensions to help Python programmers derive datasets through workflows, to maintain derived index and column data automatically and incrementally, and to save and share results.

Potential Next Steps

- FAM-side computation
- Native paging across FAM