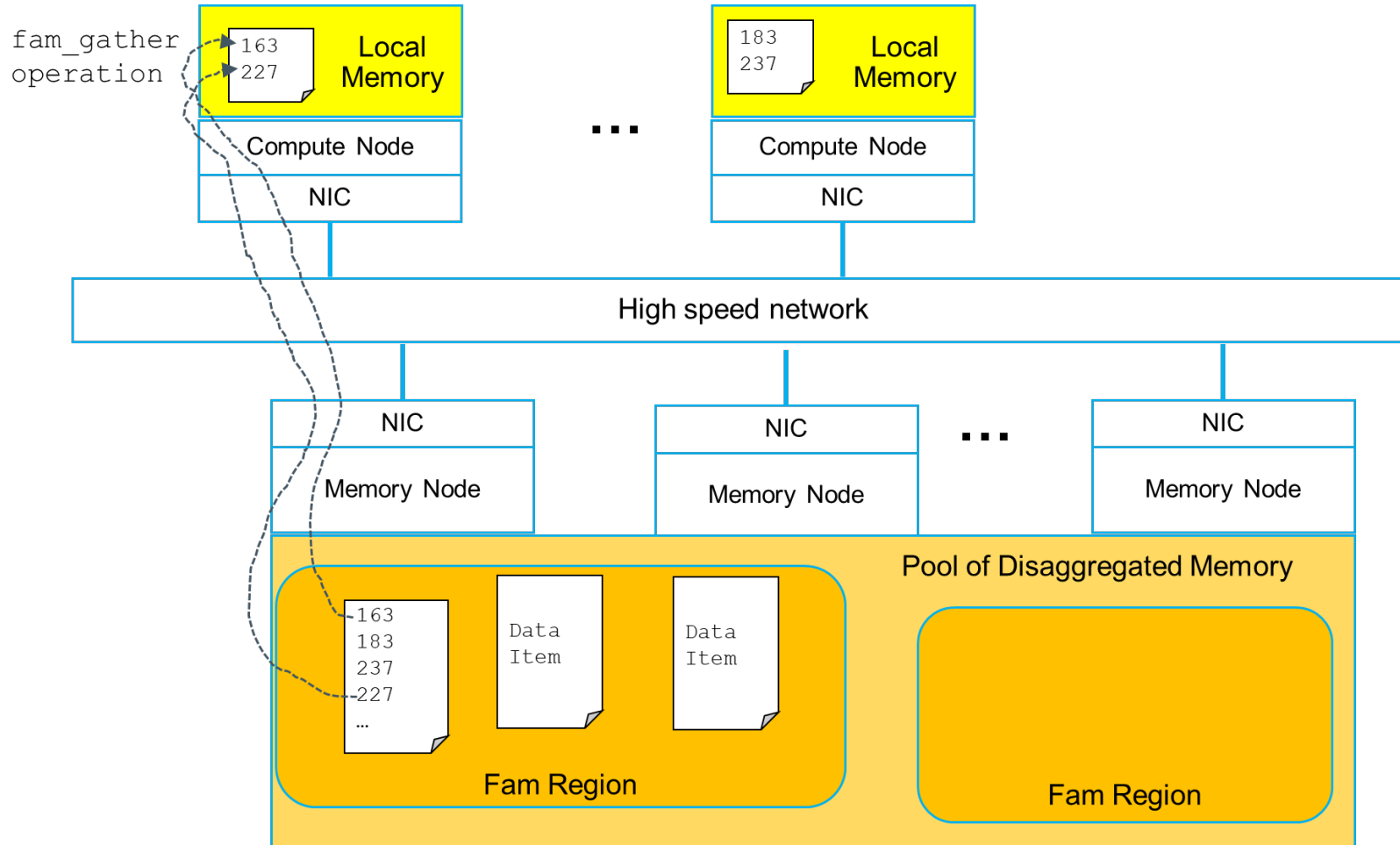


Coupling Chapel-Powered HPC Workflows for Python

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)



- 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

Python code (Jupyter Notebook)

```
import arkouda as ak

// Create pdarray of random ints
pdarray1= ak.randint(0,100000000,10000)

// Create pdarray with int sequence
pdarray2 = ak.arange(0,10000,1)
```

Arkouda Server (Chapel program on compute nodes)

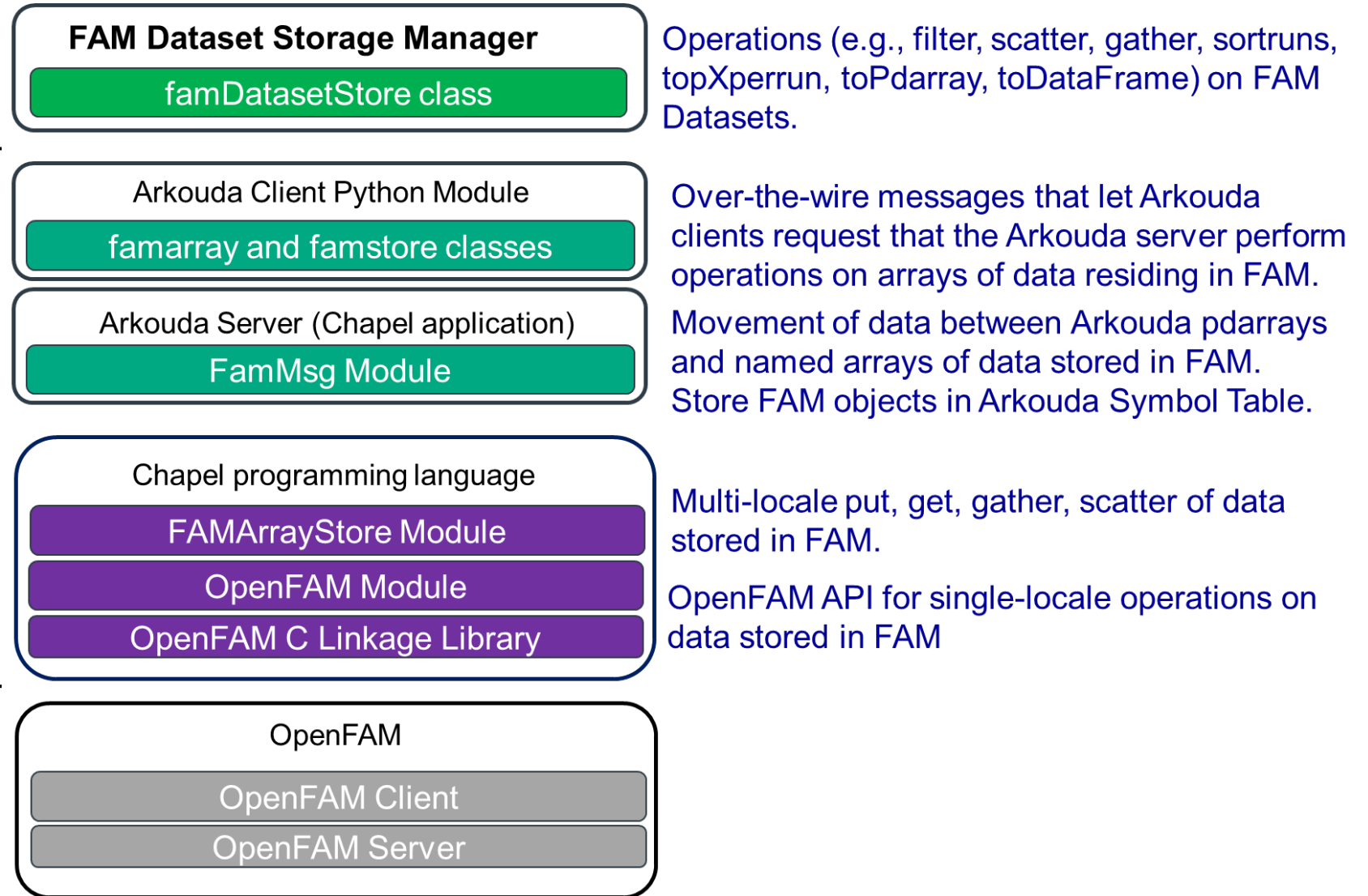


`pdarray1` and `pdarray2` are distributed in memory across compute nodes.

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



FAMArray Storage Manager for Chapel and Arkouda

Python code (Jupyter Notebook)

```
// Create large FamArray
famArray1 = fam_region.create ('aName', ak.int, 100000000)

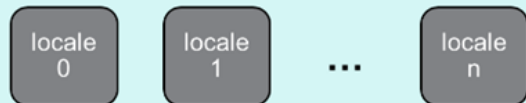
// Create pdarray of random ints
pdarray1 = ak.randint(0, 100000000, 10000)

// Create pdarray with int sequence
pdarray2 = ak.arange(0, 10000, 1)

// Scatter contents of pdarray2 across famArray1
famArray1[pdarray1] = pdarray2
```

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.

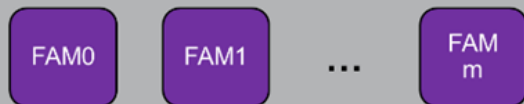
Arkouda Server (Chapel program on compute nodes)



- `pdarray1` and `pdarray2` are distributed in memory across compute nodes.
- `famArray1` is a proxy for a FAM distributed array.

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.

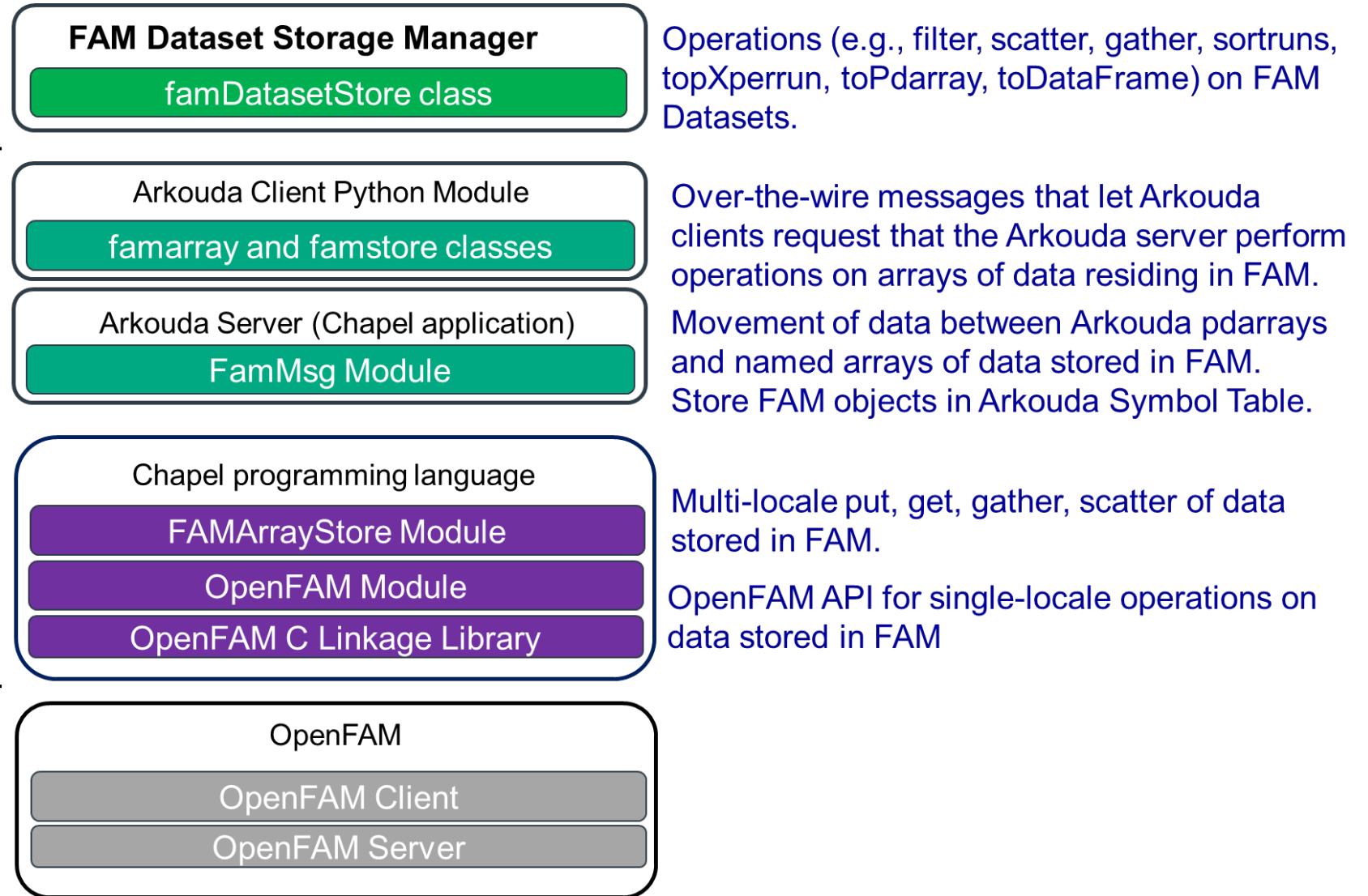
FAM Array Store / OpenFAM modules (on FAM nodes)



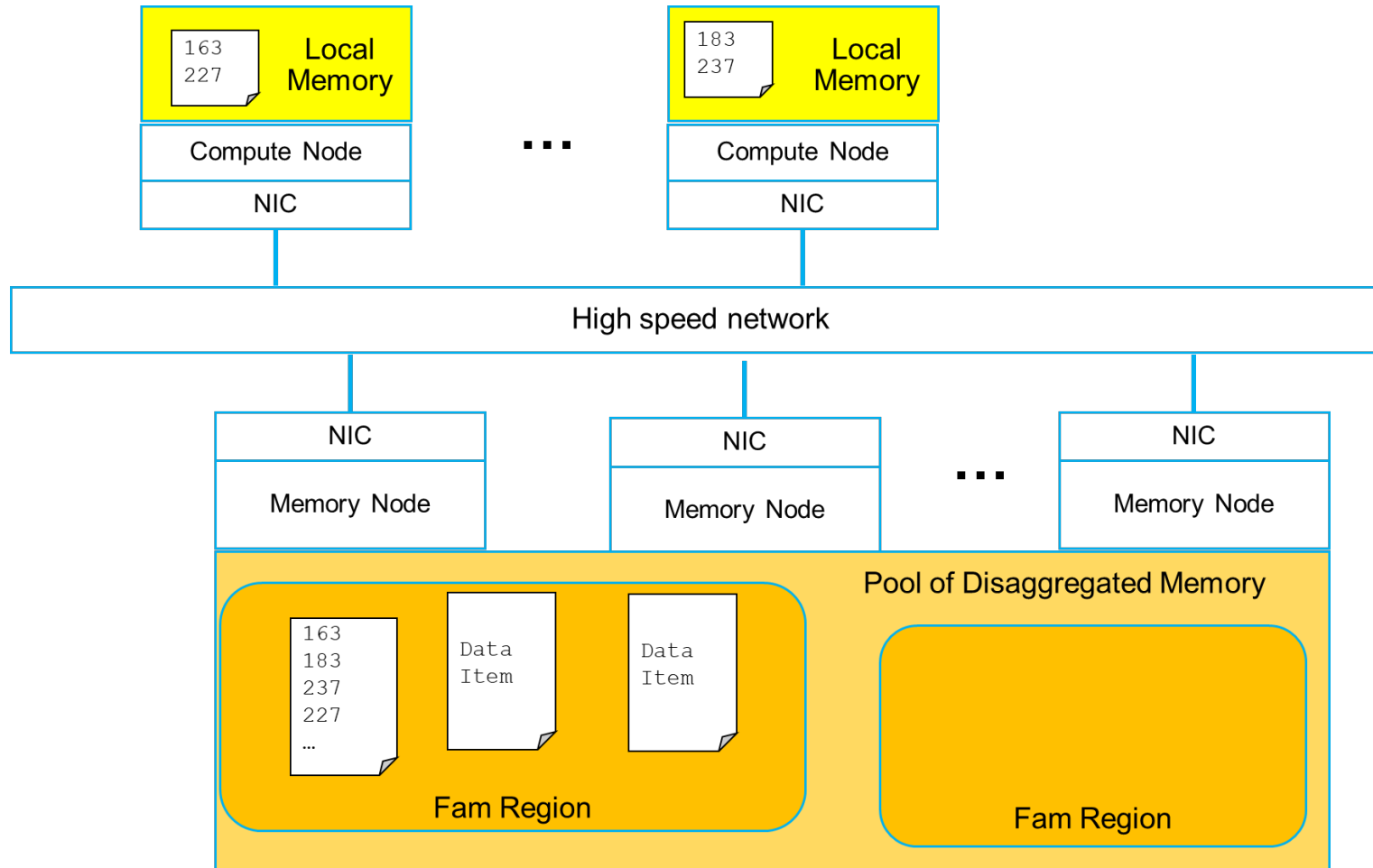
- `famArray1` distributed across FAM.
- FAMArrayStore invokes OpenFAM module to execute the scatter operation.

FAM Dataset Storage Manager

FAMArray Storage Manager



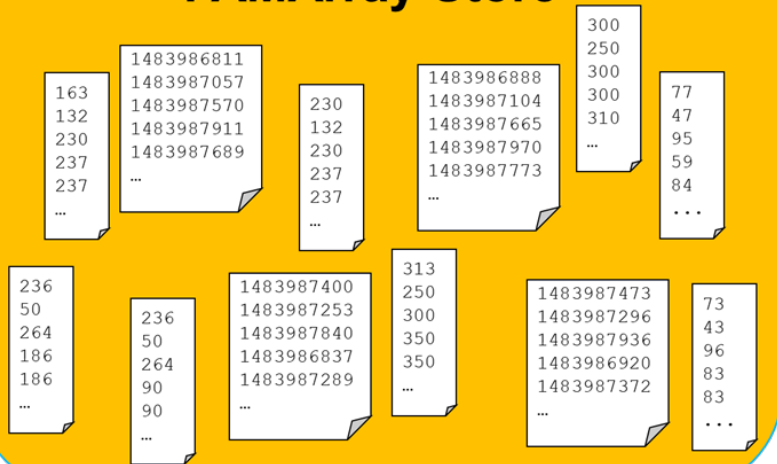
Getting the most from FAM + Arkouda



- 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.

Taxi Data FAM region FAMArray Store



Presented to
Arkouda
programmer as
columns in a
logical dataset

Taxi Data FAM Dataset Store

pu_zone	do_zone	pu_time	do_time	fare	duration
163	230	1483986811	1483986888	300	77
132	132	1483987057	1483987104	250	47
230	230	1483987570	1483987665	300	95
237	237	1483987911	1483987970	300	59
237	237	1483987689	1483987773	310	84
...
236	236	1483987400	1483987473	313	73
50	50	1483987253	1483987296	250	43
264	264	1483987840	1483987936	300	96
186	90	1483986837	1483986920	350	83
186	90	1483987289	1483987372	350	83

pickup_zone	dropoff_zone	duration	fare
265	265	130	13500
265	265	130	13500
265	265	130	13500
265	265	130	13500
265	265	130	13500
...
138	265	4072	17100
138	265	4072	17100
138	265	4072	17100
138	265	4072	17100
138	265	4072	17100

pickup_zone	dropoff_zone	duration	fare
265	265	130	13500
265	265	130	13500
265	265	130	13500
265	265	130	13500
265	265	130	13500
...
138	265	4072	17100
138	265	4072	17100
138	265	4072	17100
138	265	4072	17100
138	265	4072	17100

Derived FAM Datasets represent indices into base data

Trip

pickupzone	dropoffzone	pickuptime	dropofftime	fare
230	100	1486146910	1486147282	600
246	233	1486147980	1486149965	2050
186	48	1486146833	1486147538	900
132	265	1486147893	1486150863	12600
163	141	1486146916	1486147462	725
161	170	1486147691	1486148005	569
107	107	1486147745	1486148415	10850
141	262	1486148260	1486148477	500
162	186	1486146899	1486148169	1306
162	237	1486147680	1486148313	862

Number_of_Batches

`filter(Trip,
'fare',
'gt', 10000)`

Trip.fare_gt_10000

3
6

Number_of_Batches

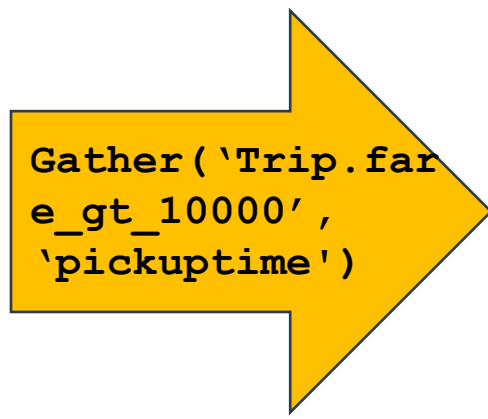
Can use the index to gather columns of interest

Trip.fare_gt_10000: 3
6

Trip

pickupzone	dropoffzone	pickuptime	dropofftime	fare
230	100	1486146910	1486147282	600
246	233	1486147980	1486149965	2050
186	48	1486146833	1486147538	900
132	265	1486147893	1486150863	12600
163	141	1486146916	1486147462	725
161	170	1486147691	1486148005	569
107	107	1486147745	1486148415	10850
141	262	1486148260	1486148477	500
162	186	1486146899	1486148169	1306
162	237	1486147680	1486148313	862

Number_of_Batches



Trip.fare_gt_10000

pickuptime
1486147893
1486147745

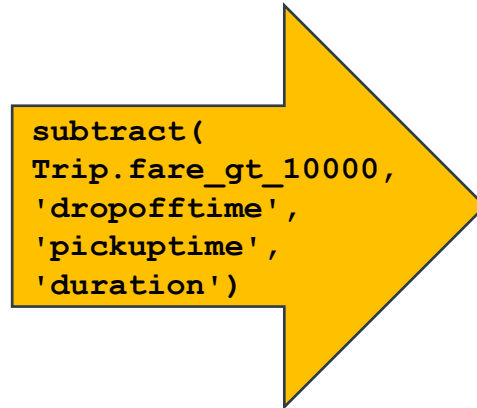
Number_of_Batches

Some operations produce derived columns

Trip.fare_gt_10000
pickuptime **dropofftime**

1486147893	1486150863
1486147745	1486148415

Number_of_Batches

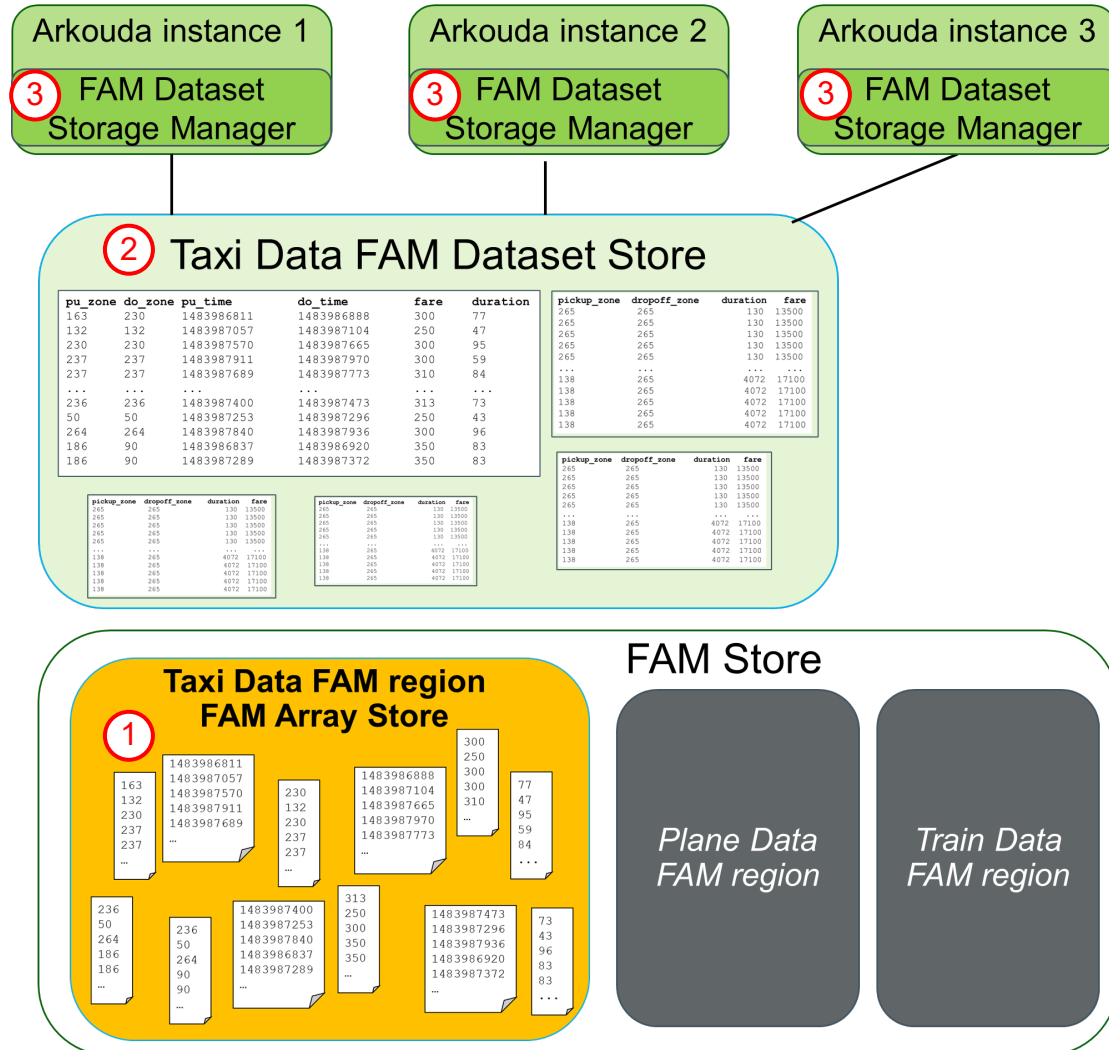


Trip.fare_gt_10000
pickuptime **dropofftime** **duration**

1486147893	1486150863	2970
1486147745	1486148415	670

Number_of_Batches

FAM Dataset Storage Manager

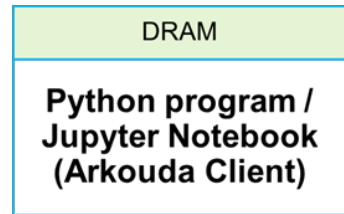


1. Ingested data is stored in famarrays in a FAM Array Store.
2. The FAM Dataset Storage Manager presents integrated views of related base and derived data items as a collection of FAM Datasets.
3. Multiple Arkouda processes can attach to the same FAM Dataset Store, sharing FAM data while maintaining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

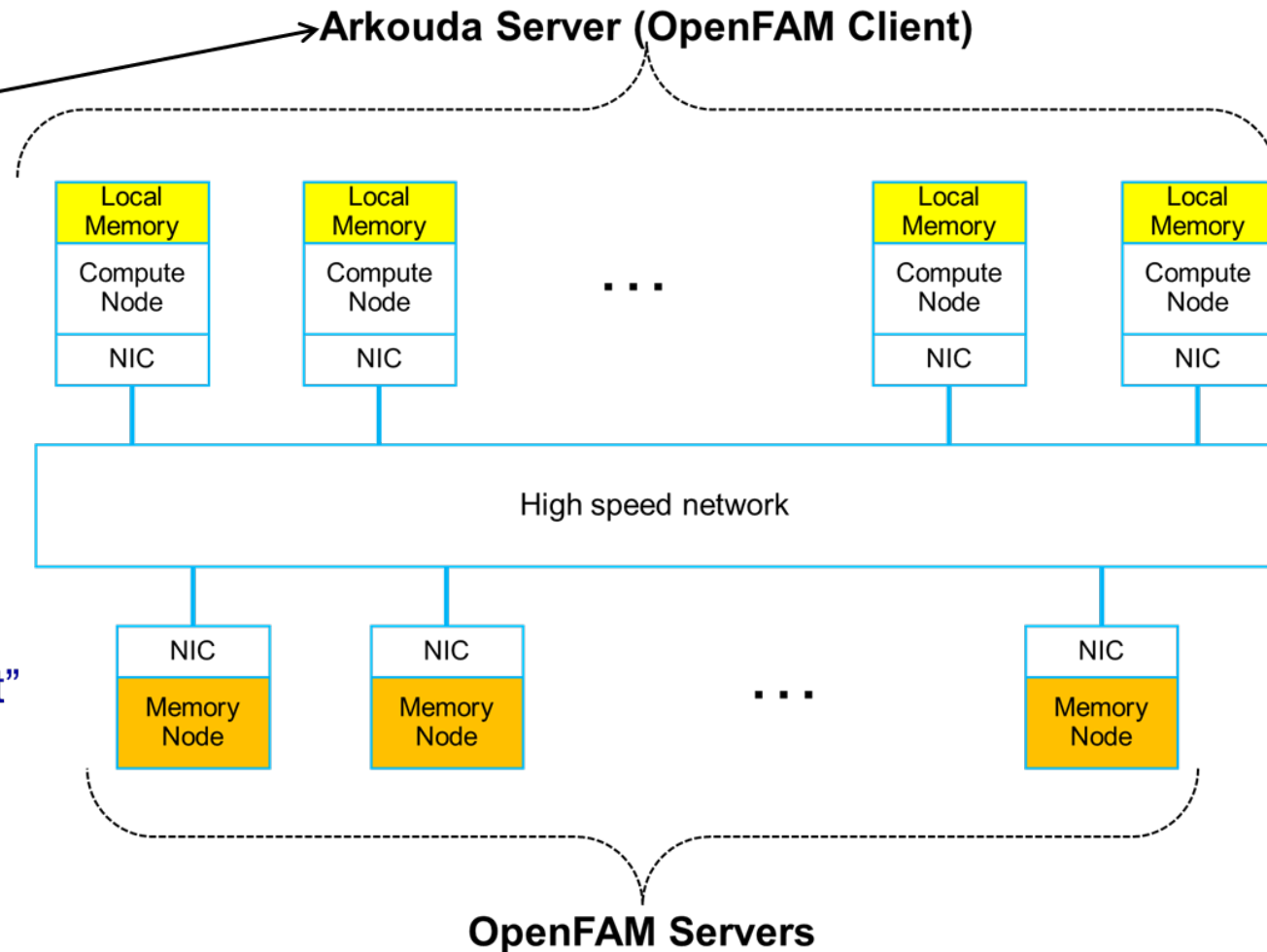
Derived datasets could be updated using an automatic update mechanism as new batches of data are ingested.

Working vs. Shared Data and Metadata



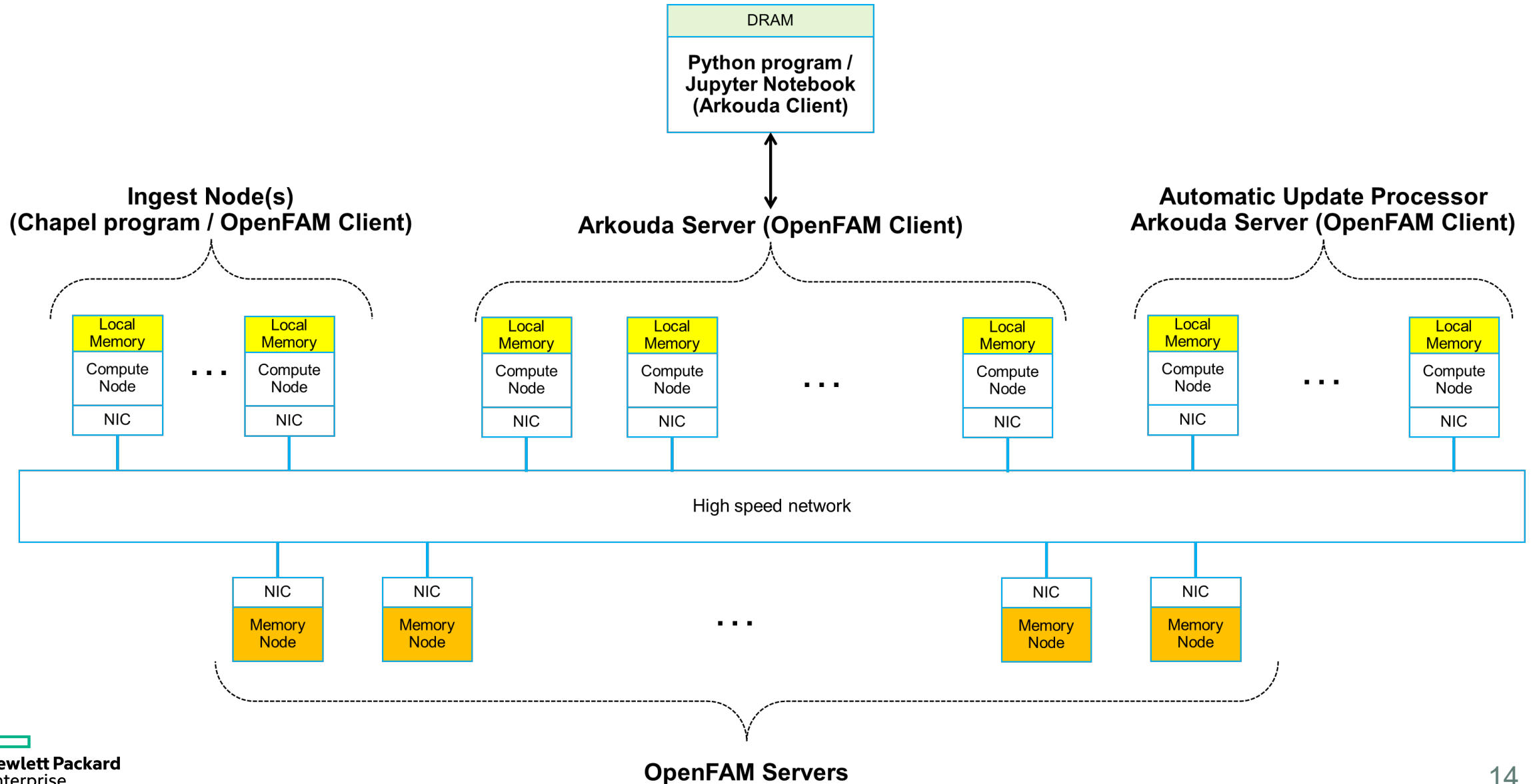
A transient FAM Dataset Storage Manager instance runs within the Jupyter Notebook process. Working metadata resides in memory here.

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

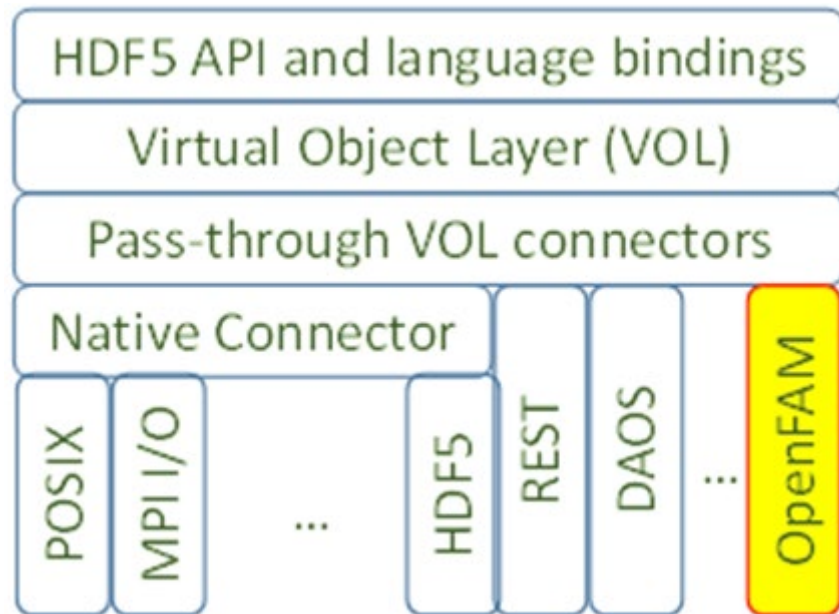


Within the Arkouda server processes, the FAM Dataset Storage Manager “pages” batches of data between famarrays and Arkouda parallel distributed arrays (pdarrays). Working data resides in the local memory of compute nodes.

Arkouda and Chapel programs can use FAM to share results



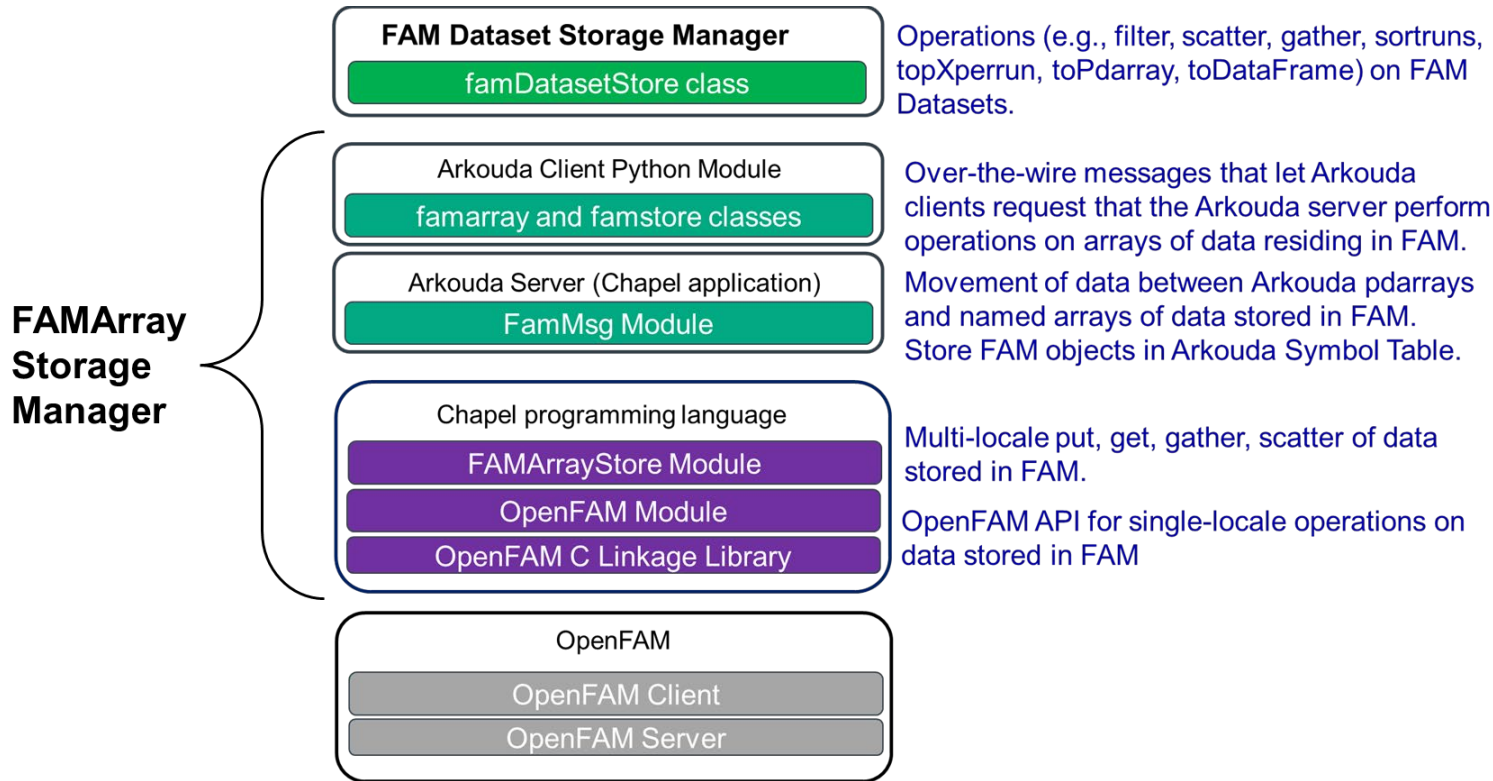
In addition, we also provide an HDF5 interface for FAM



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 2020.
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, heterogeneous 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