Los Alamos NATIONAL LABORATORY

Secure In-Storage Execution of VTK Workloads on Modern Parallel NFS (*pNFS*) Data Servers

Qing Zheng, Brian Atkinson, Jason Lee, Gary Grider, Los Alamos National Laboratory
qzheng@lanl.gov, batkinson@lanl.gov, jasonlee@lanl.gov, ggrider@lanl.gov

Daoce Wang, University of Nebraska Omaha
daocewang@unomaha.edu

LA-UR-25-31157

Overview

Goal

Fast scientific insight

Problem

Large data transfers increasingly limit VTK/ParaView viz performance

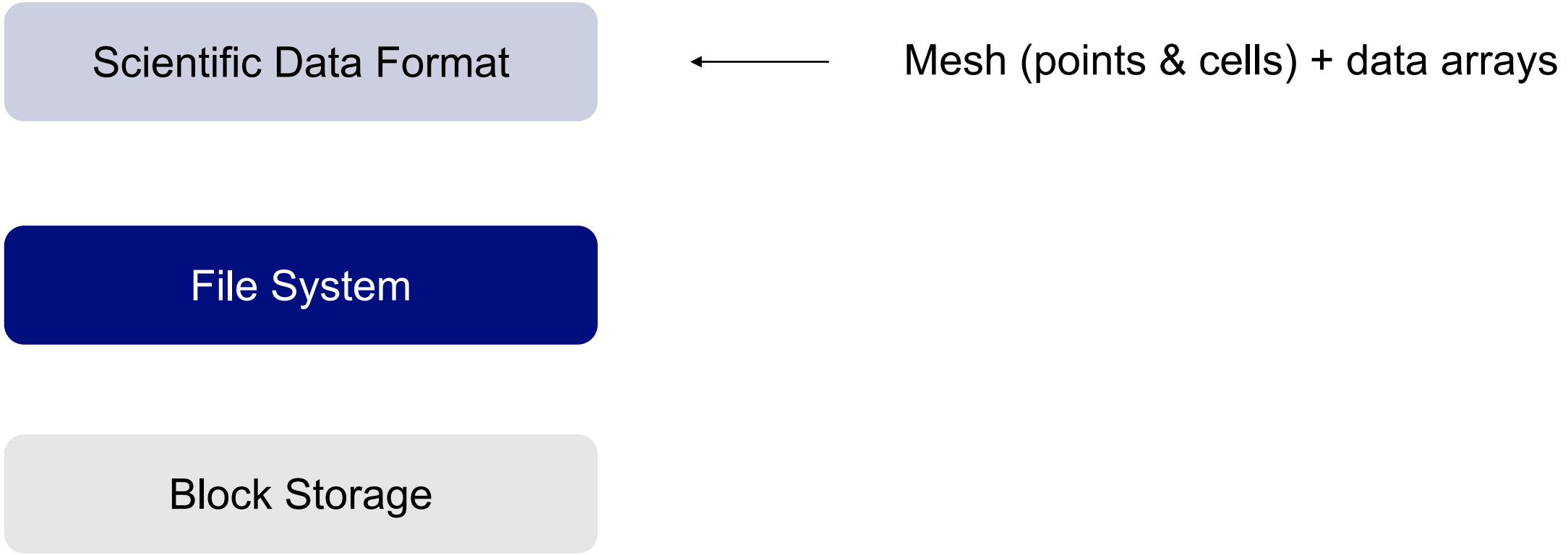
Technique

A pNFS pushdown architecture for in-storage execution of VTK code

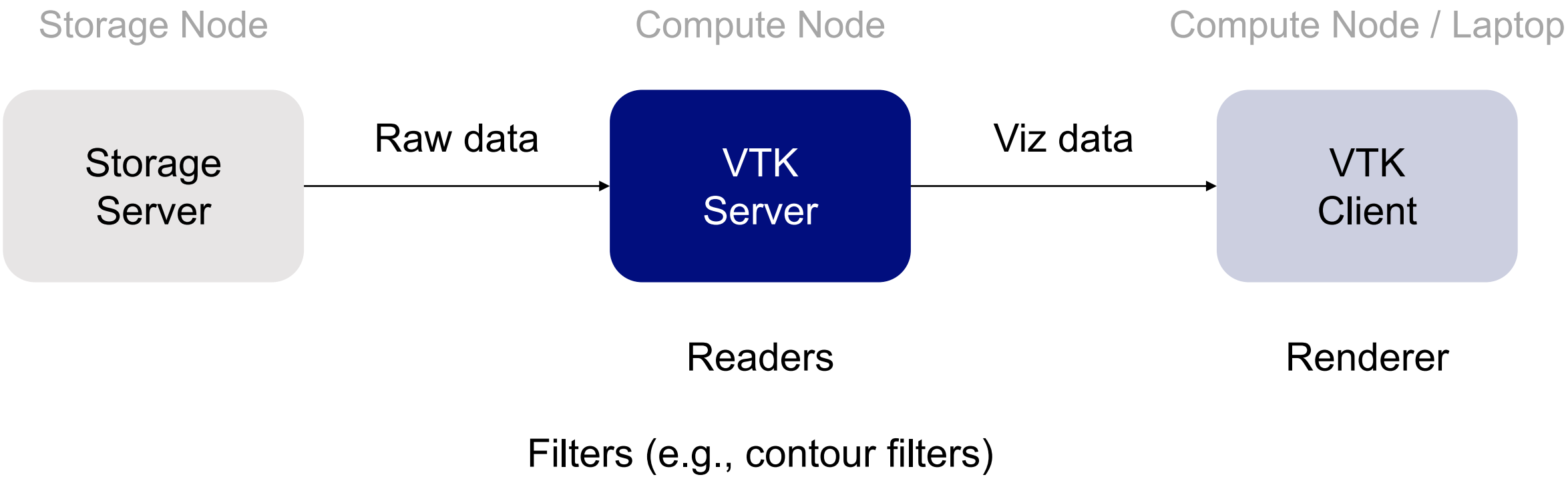
Results

6x speedup in end-to-end runtime with 2 real-world apps

Scientific Storage I/O Stack

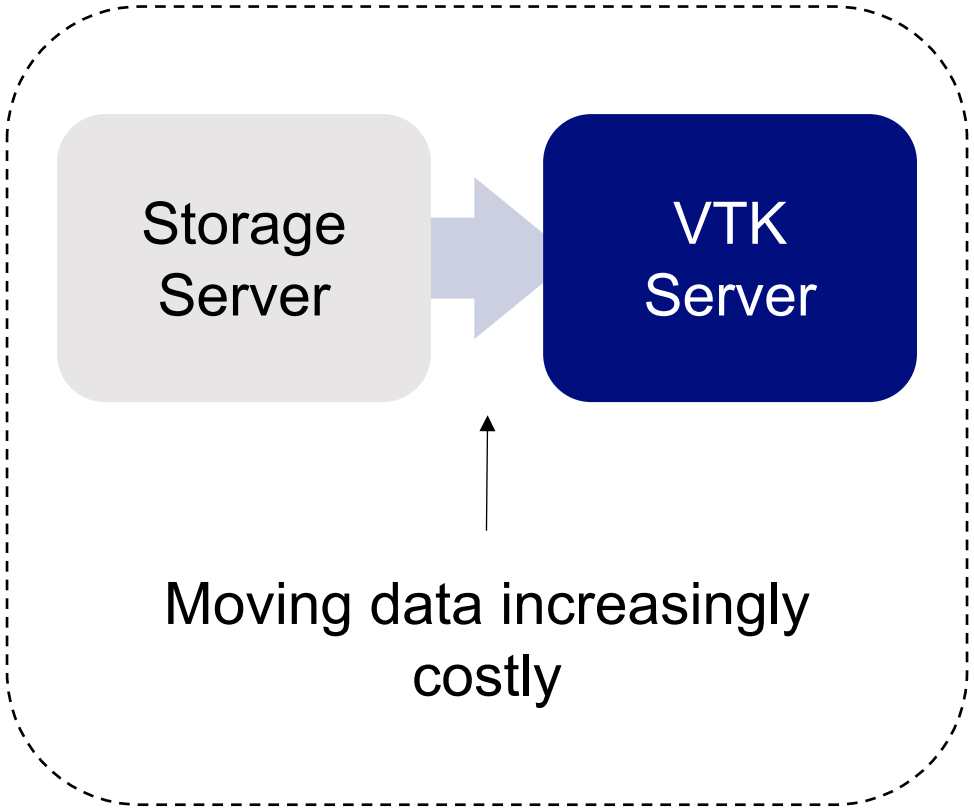


VTK Pipeline



Existing Data Reduction Techniques

Problem:



1. Data array selection

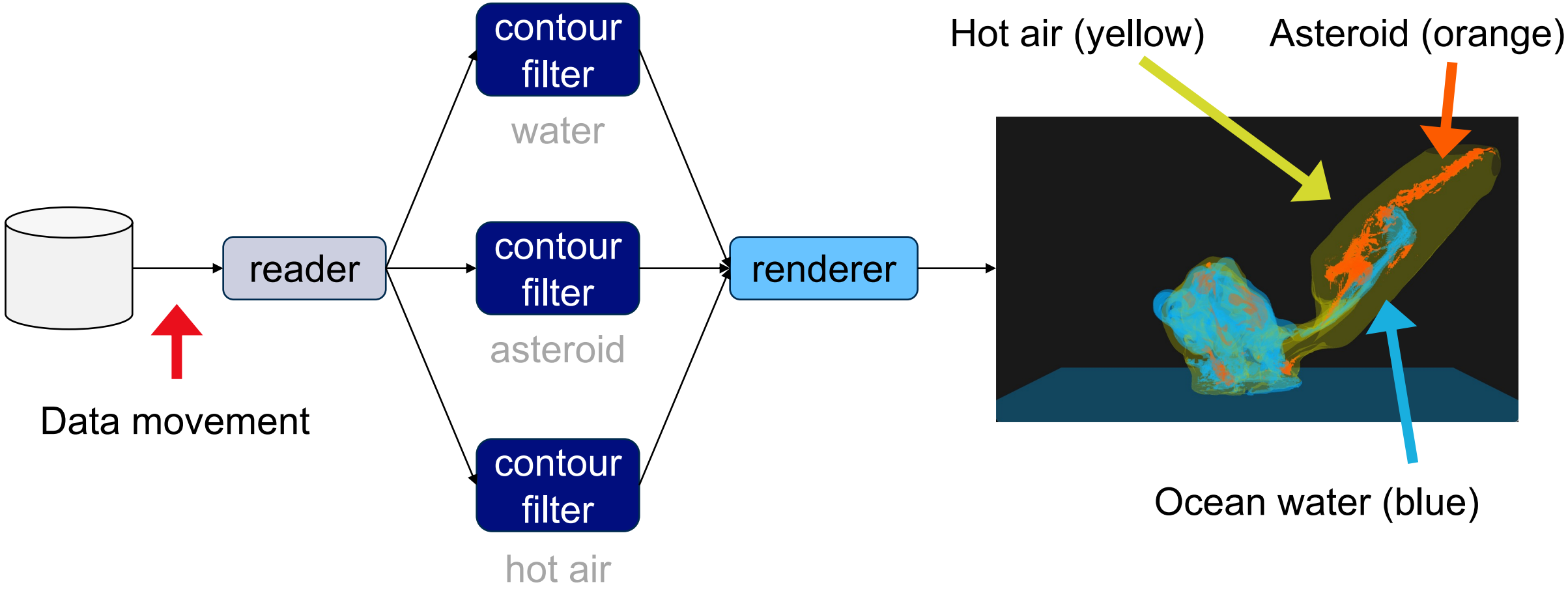
Only read relevant arrays

2. Compression

Lower the size of each array and the mesh

Example: LZ4, gzip, …

Example: Asteroid Impact (sc16 sci-viz showcase)



Applying Today's Reduction Techniques

Data array selection

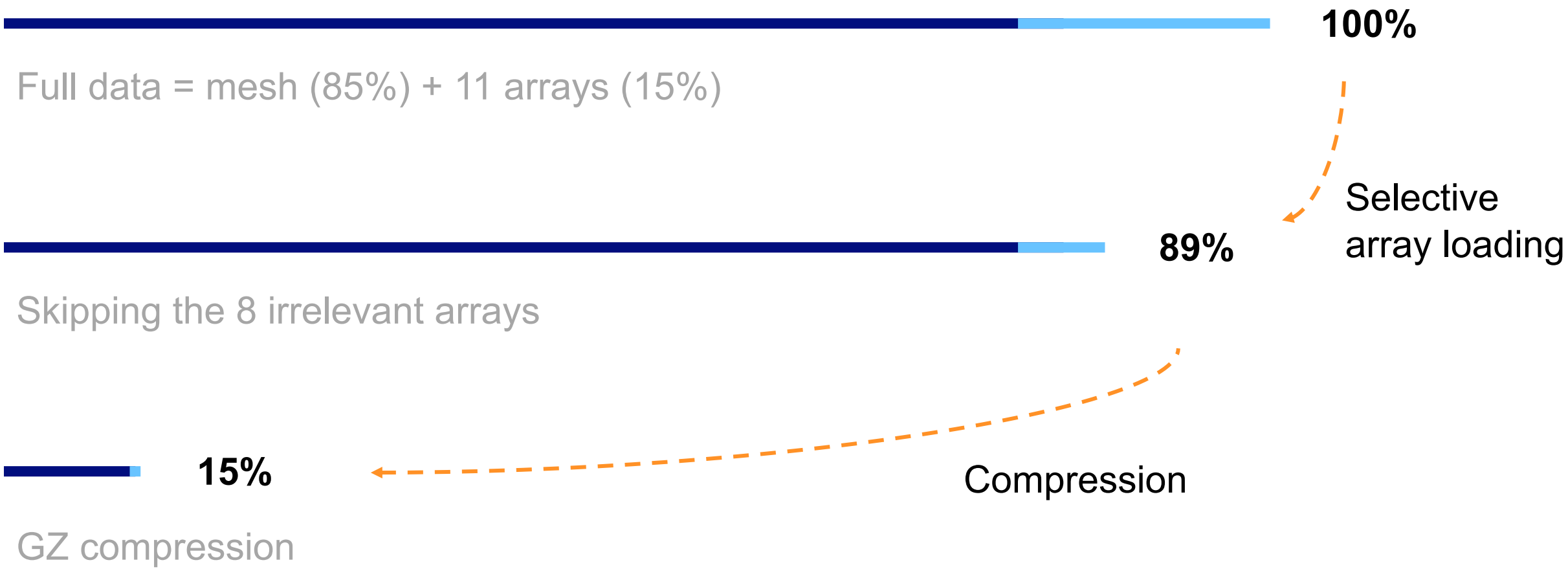
	Array	Type	Description
1	rho	float	density in grams per cubic centimeter
2	prs	float	pressure in microbars
3	tev	float	temperature in electronvolt
4	xdt	float	x component vectors in centimeters per second
5	ydt	float	y component vectors in centimeters per second
6	zdt	float	z component vectors in centimeters per second
7	snd	float	sound speed in centimeters per second
8	grd	float	AMR grid refinement level
9	mat	float	material number id
10	v02	float	volume fraction of water
11	v03	float	volume fraction of asteroid

Compression

VTK supports gzip, LZ4, and LZMA

most effective

Data Reduction Performance



Overall reduction ratio: **6.67x**

A Brief Discussion

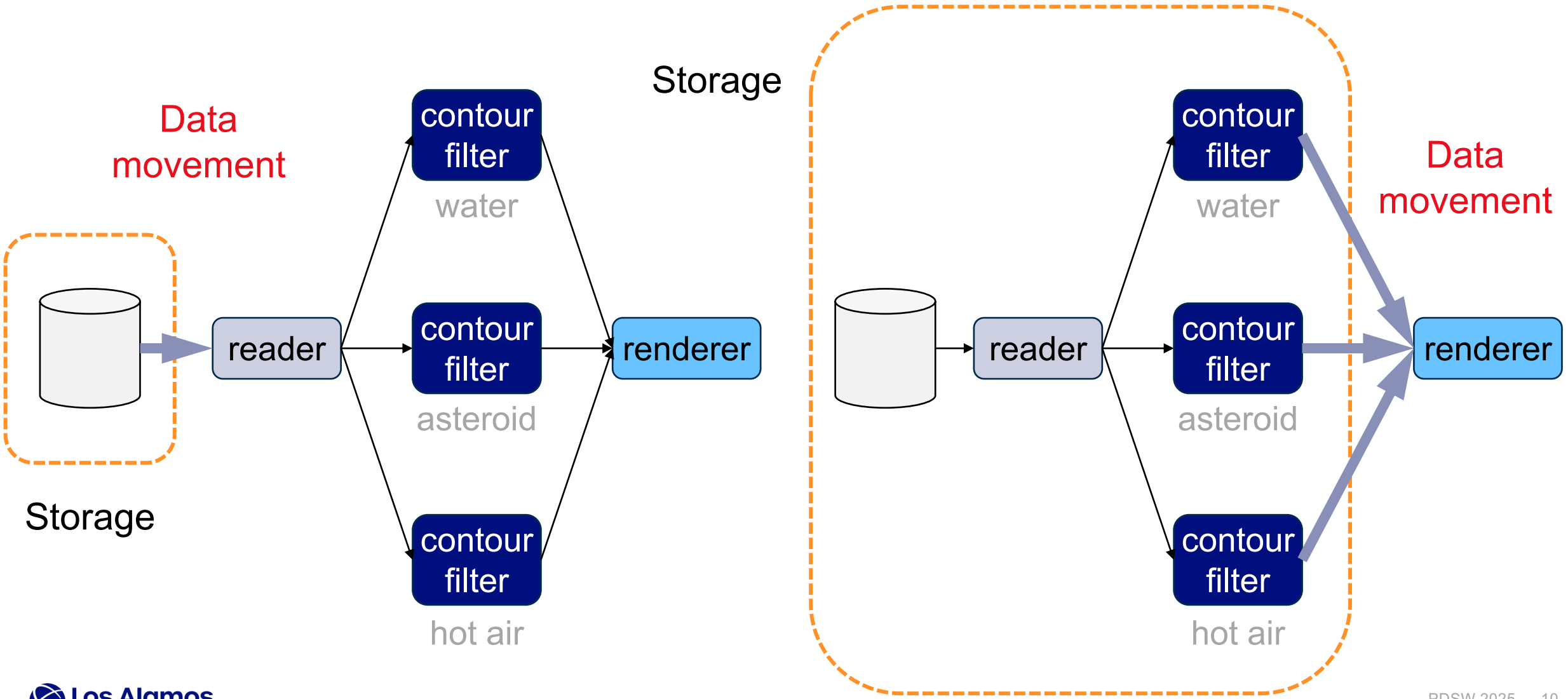
Compression isn’t effective for every dataset

Array-level selection is especially effective on structured meshes

Current techniques typically cut data volume by 1–2 orders of magnitude (~100×)

Can we do better?

Offloading Viz Processing to Storage

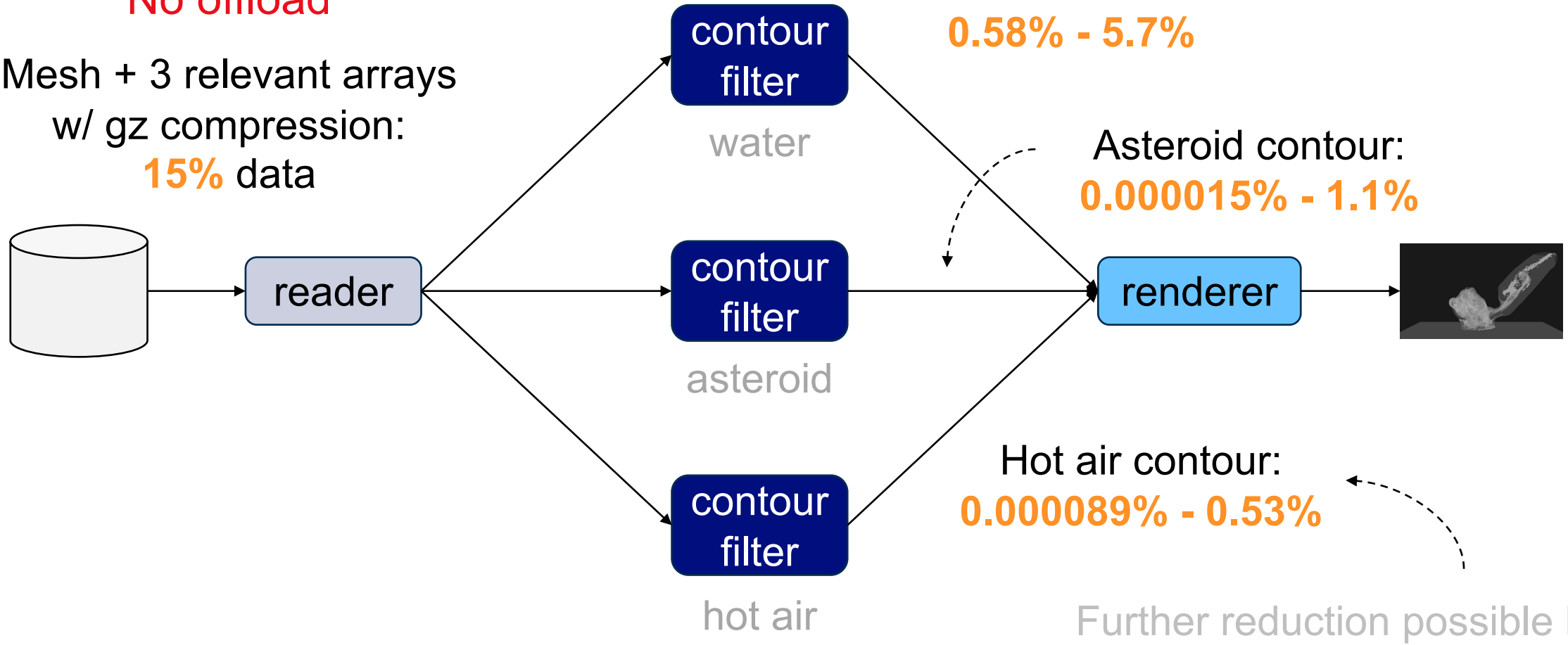


Near-Zero Data Movement

With offload

No offload

Mesh + 3 relevant arrays
w/ gz compression:
15% data



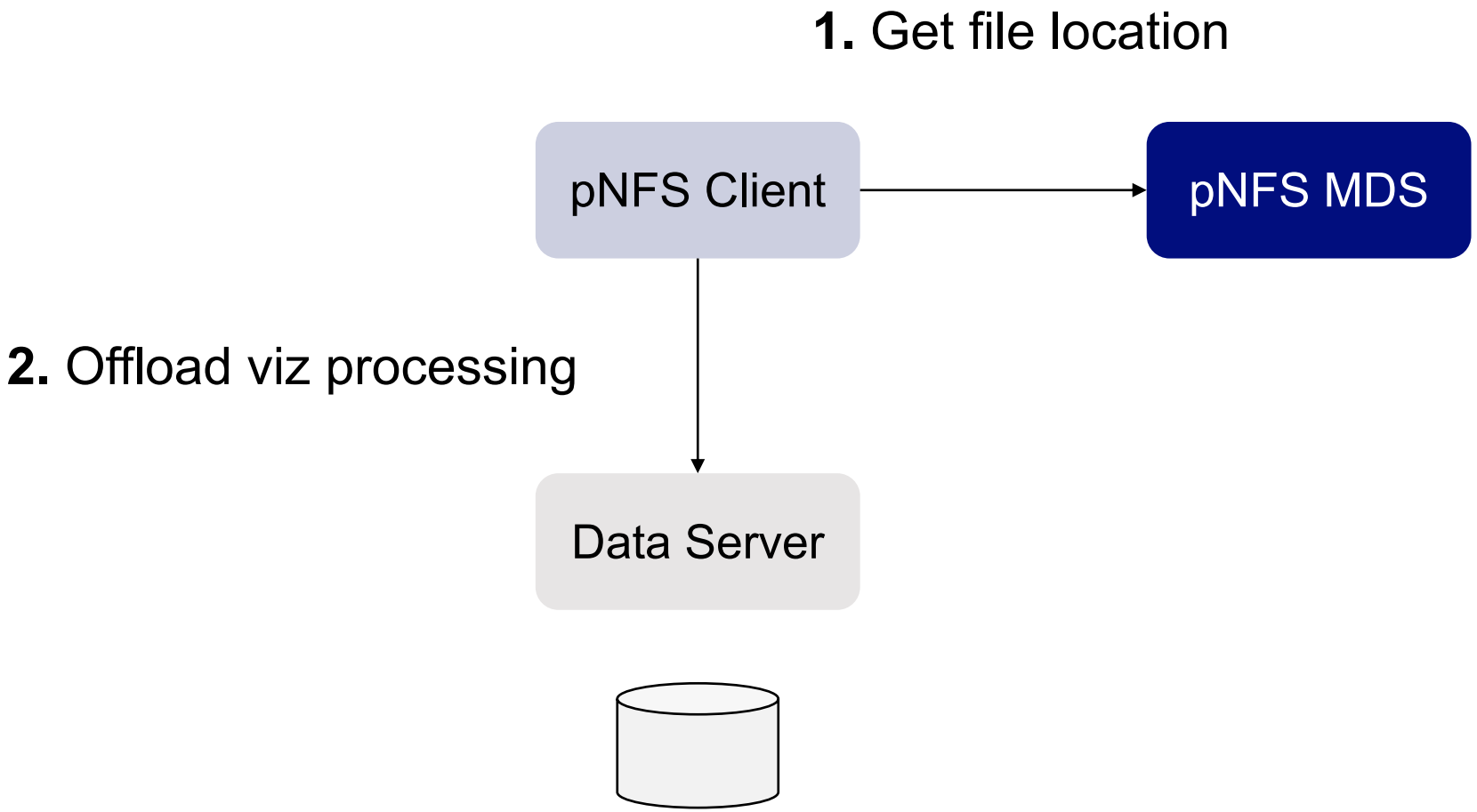
Water contour:
0.58% - 5.7%

Asteroid contour:
0.000015% - 1.1%

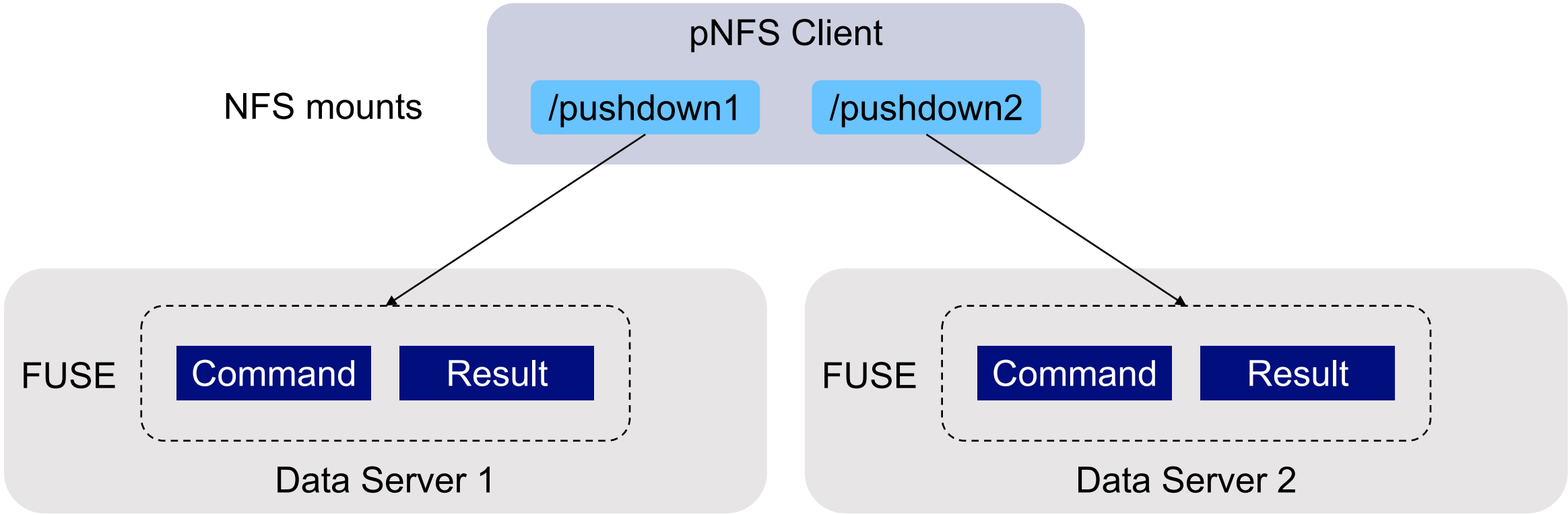
Hot air contour:
0.000089% - 0.53%

Further reduction possible by applying compression

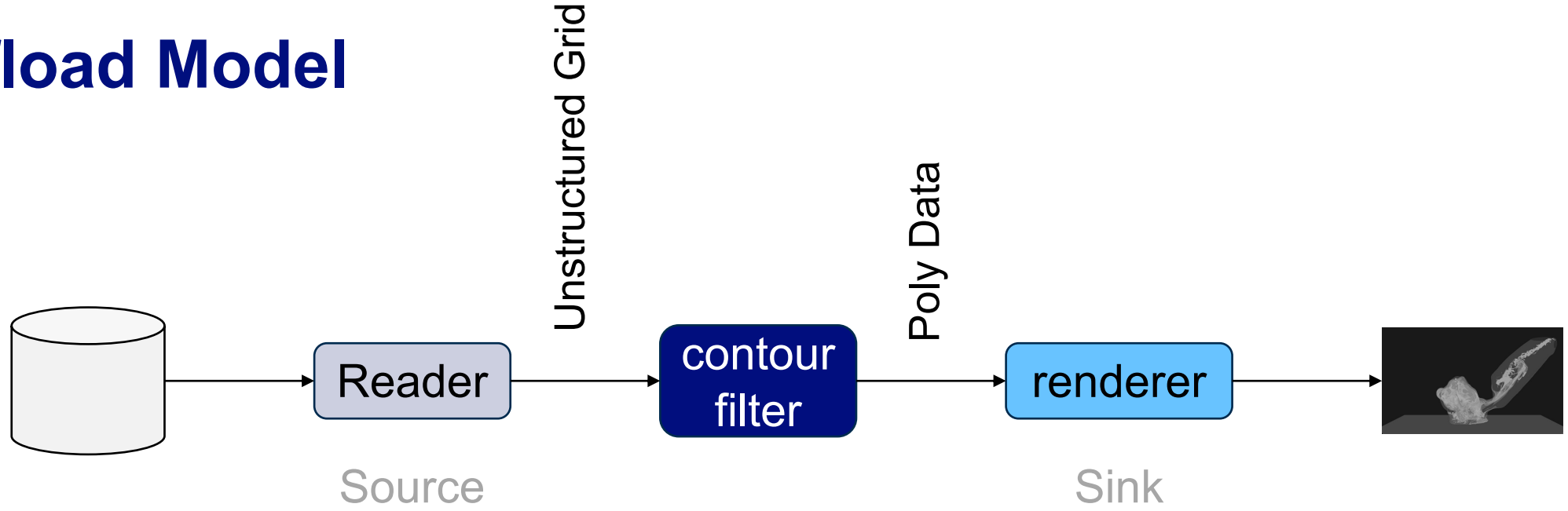
Parallel NFS (*pNFS*) Viz Pushdown

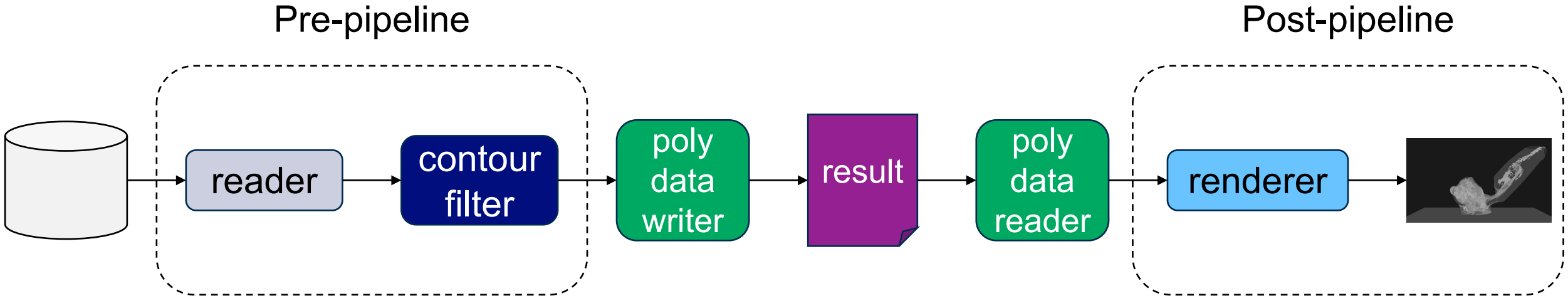


Pushdown Interface

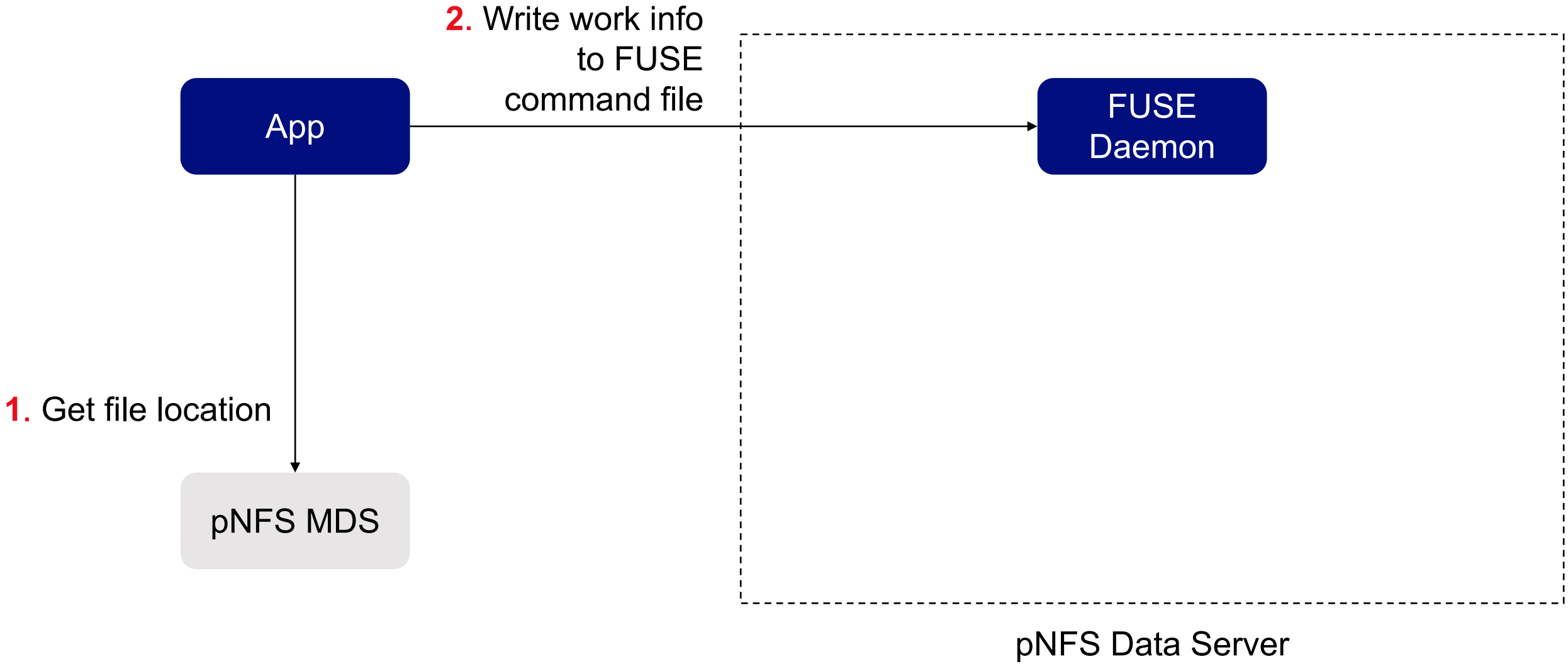


Offload Model

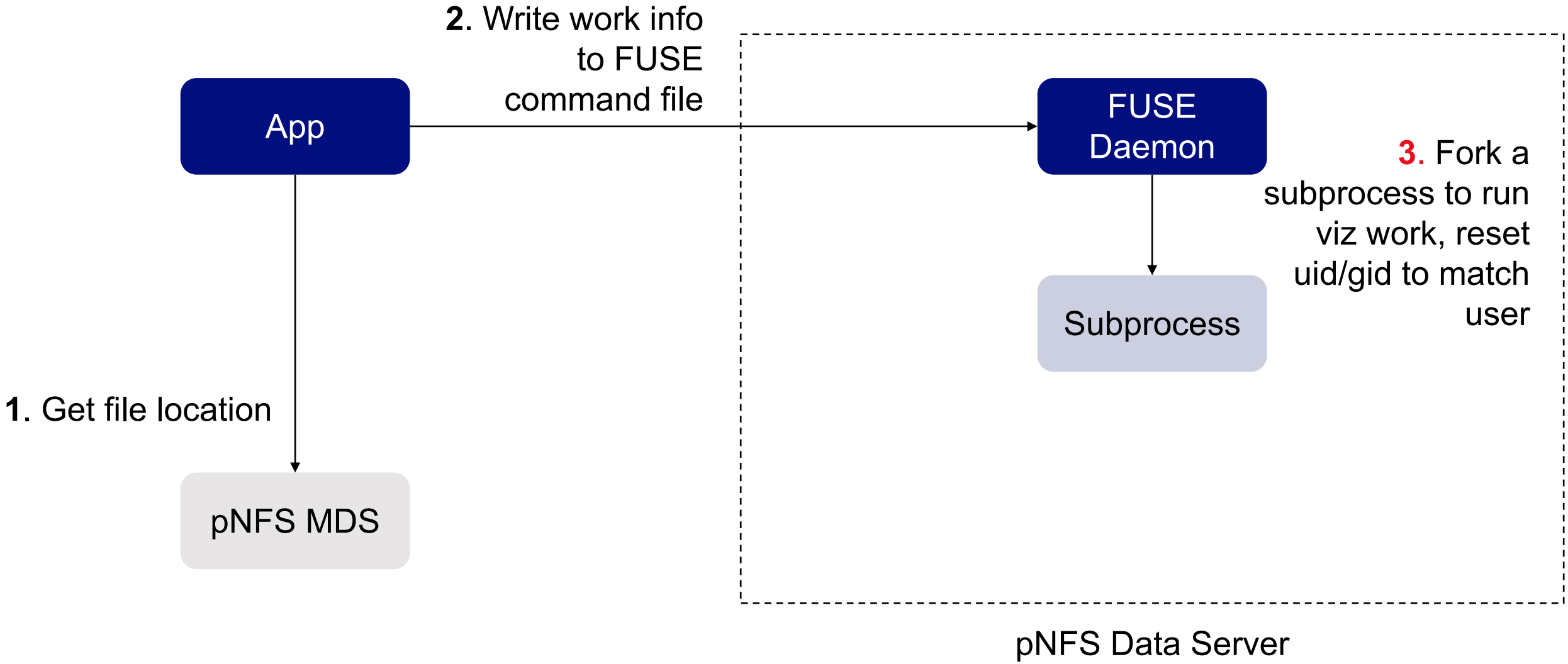




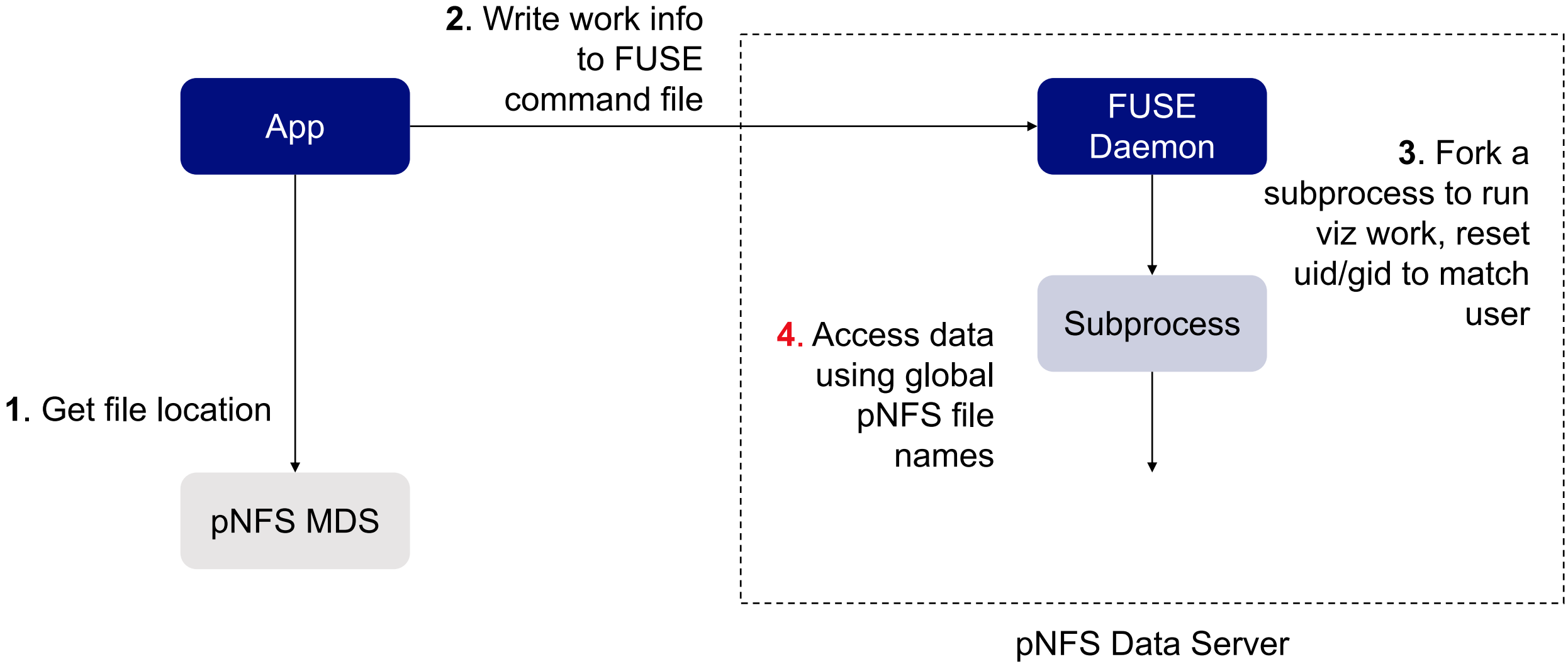
Pushdown Process #1



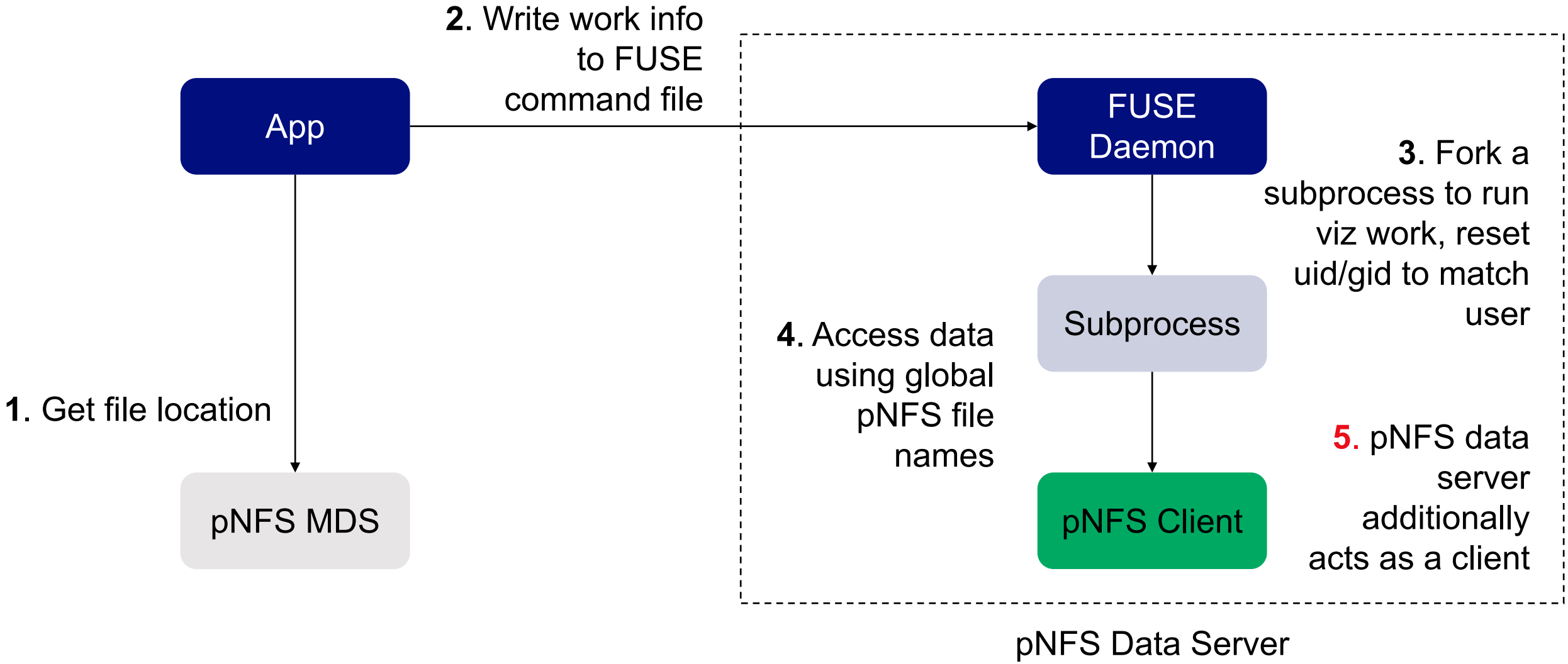
Pushdown Process #2



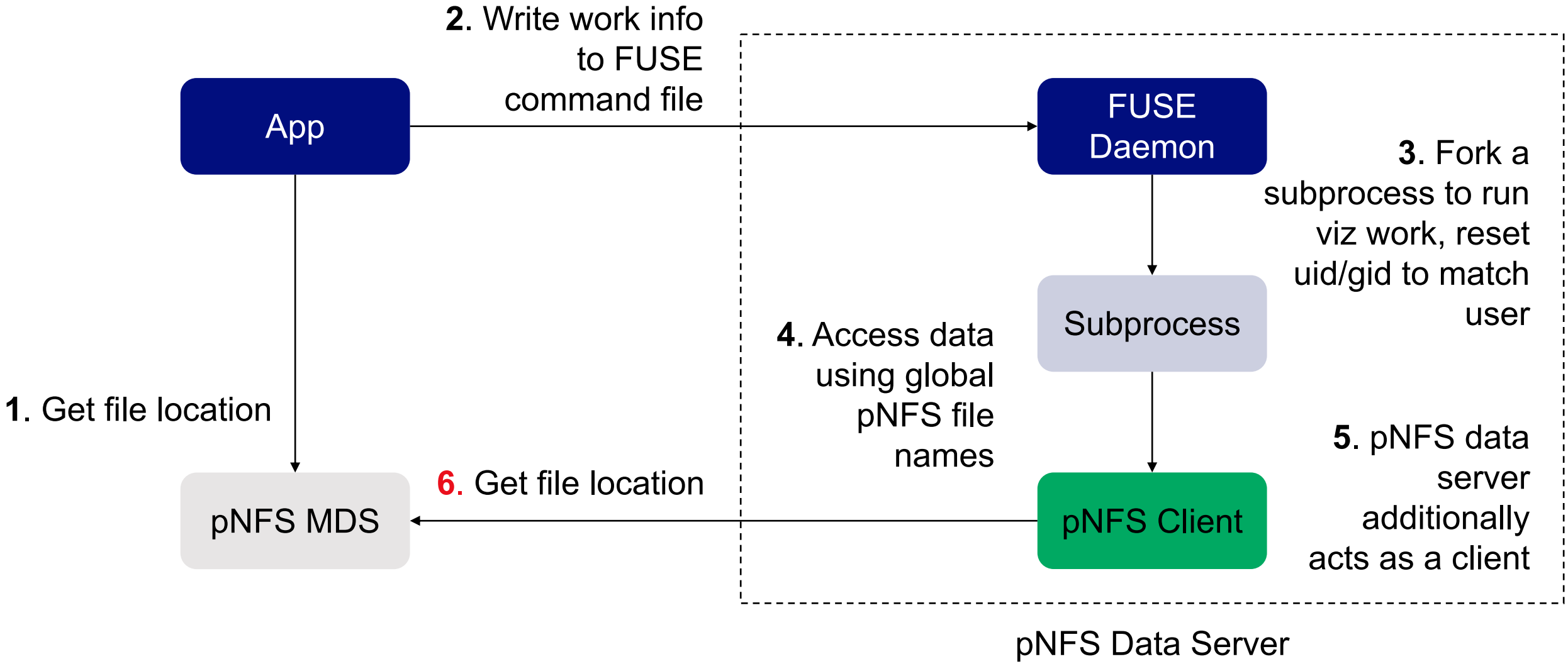
Pushdown Process #3



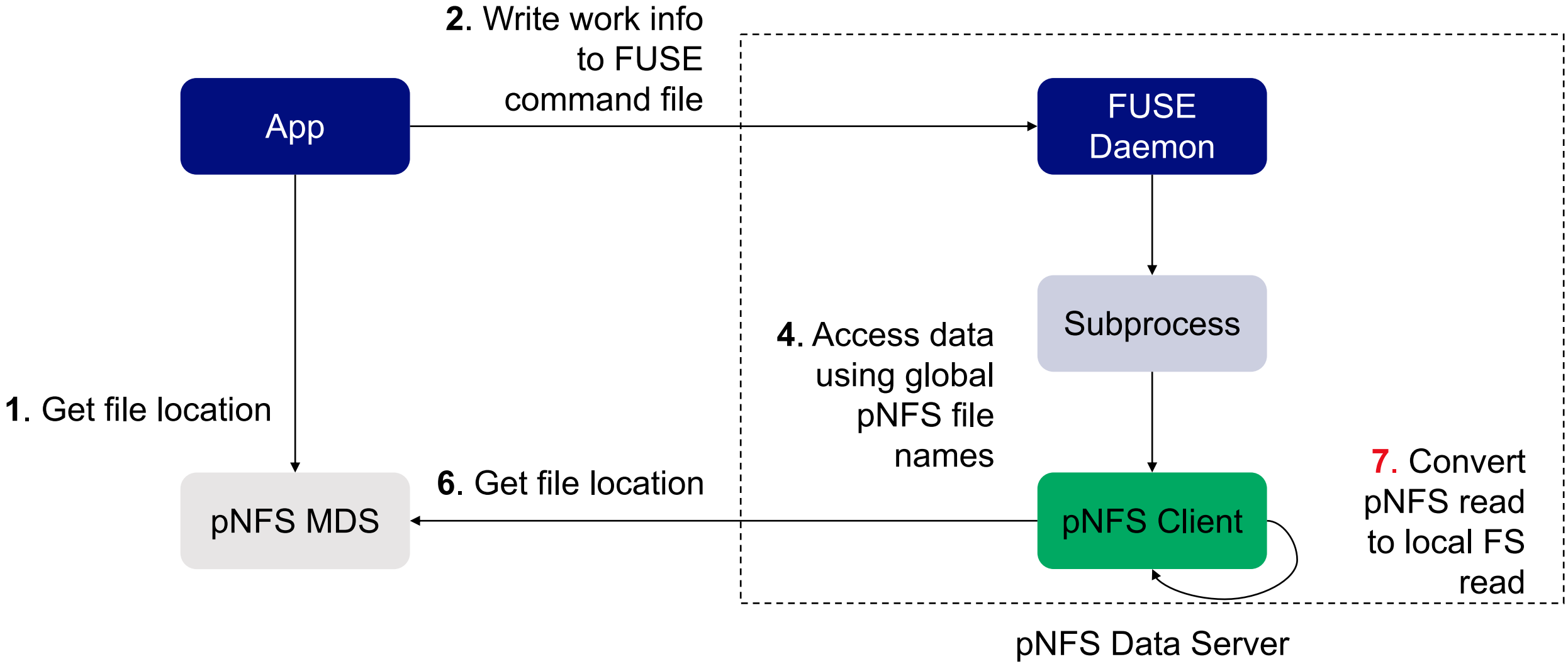
Pushdown Process #4



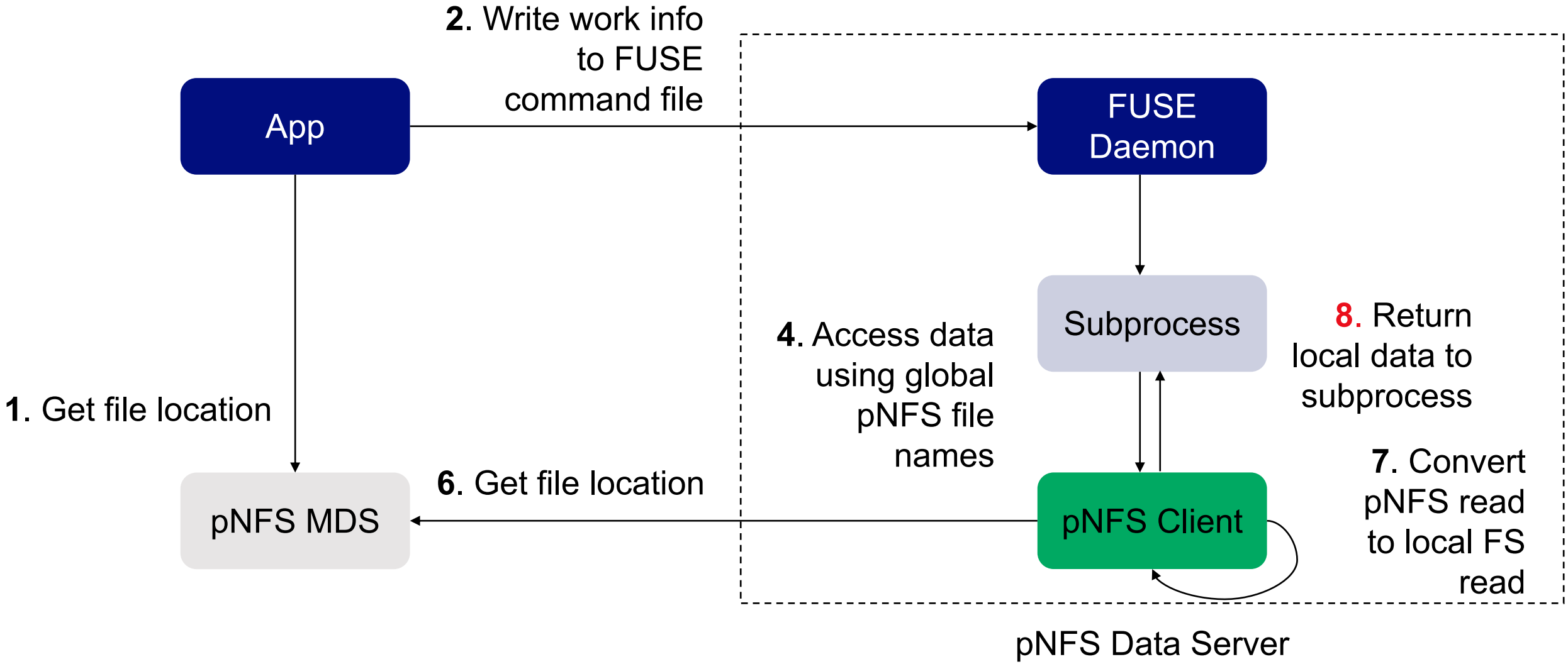
Pushdown Process #5



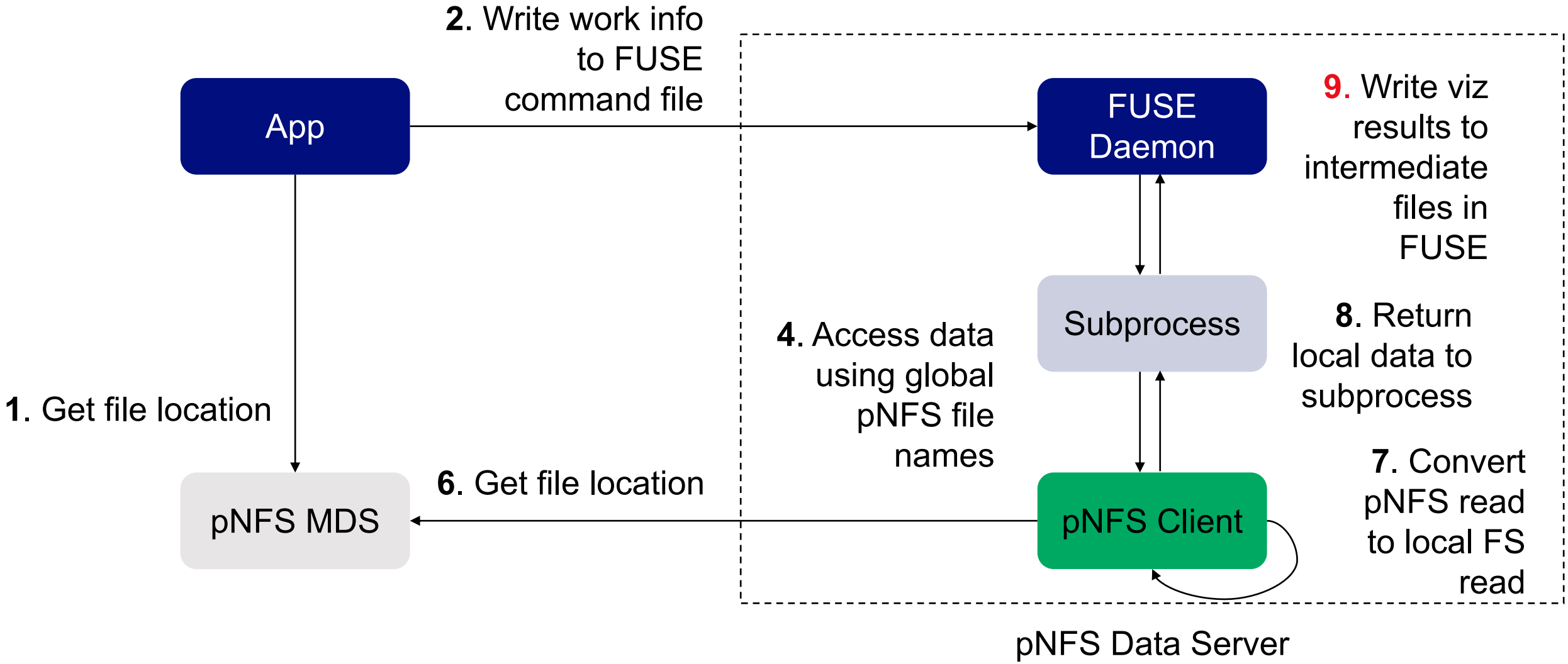
Pushdown Process #6



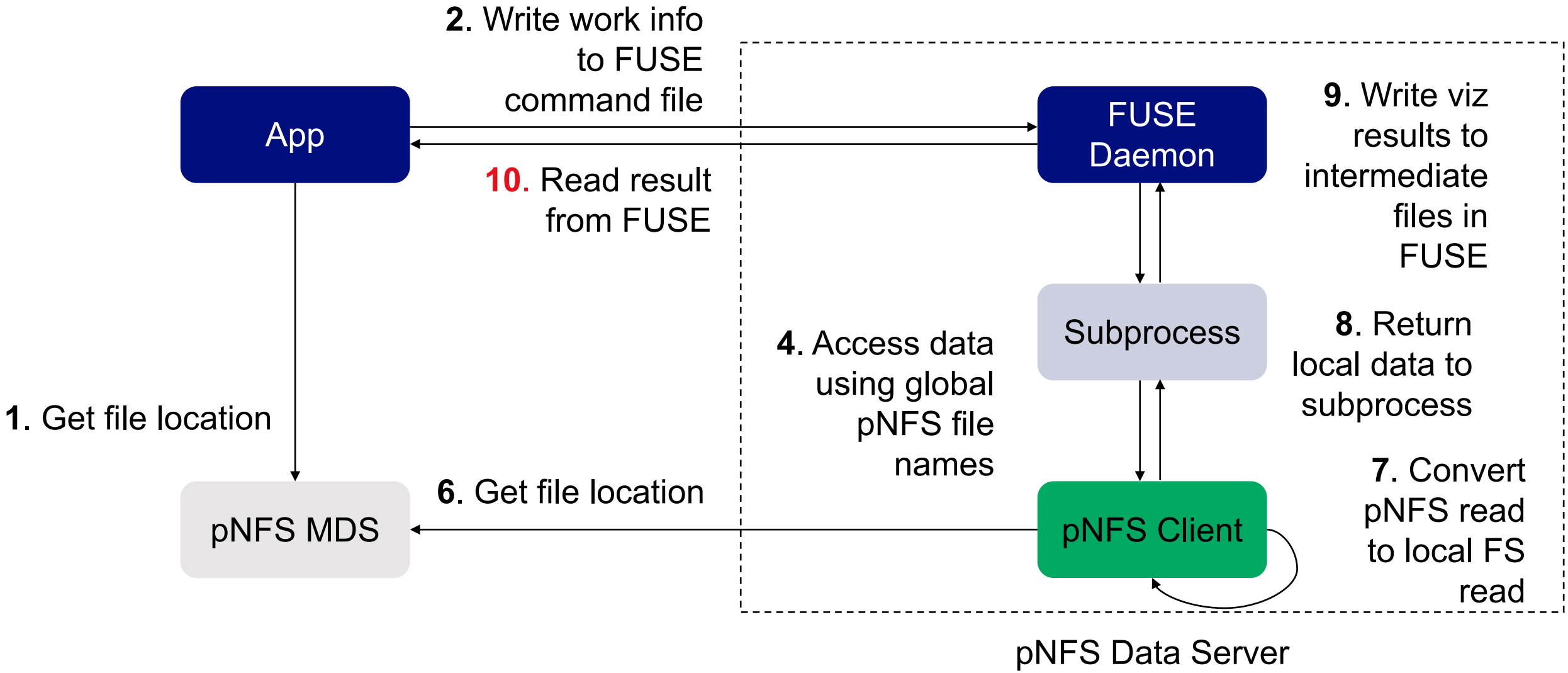
Pushdown Process #7



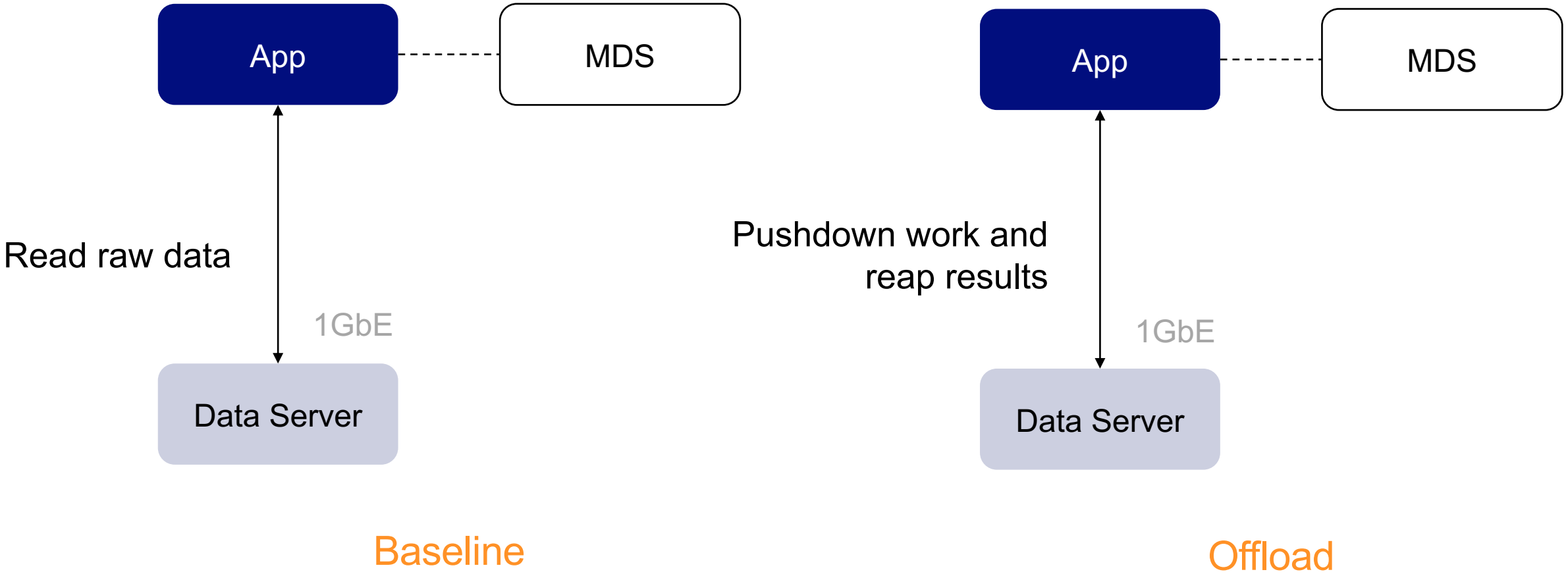
Pushdown Process #8



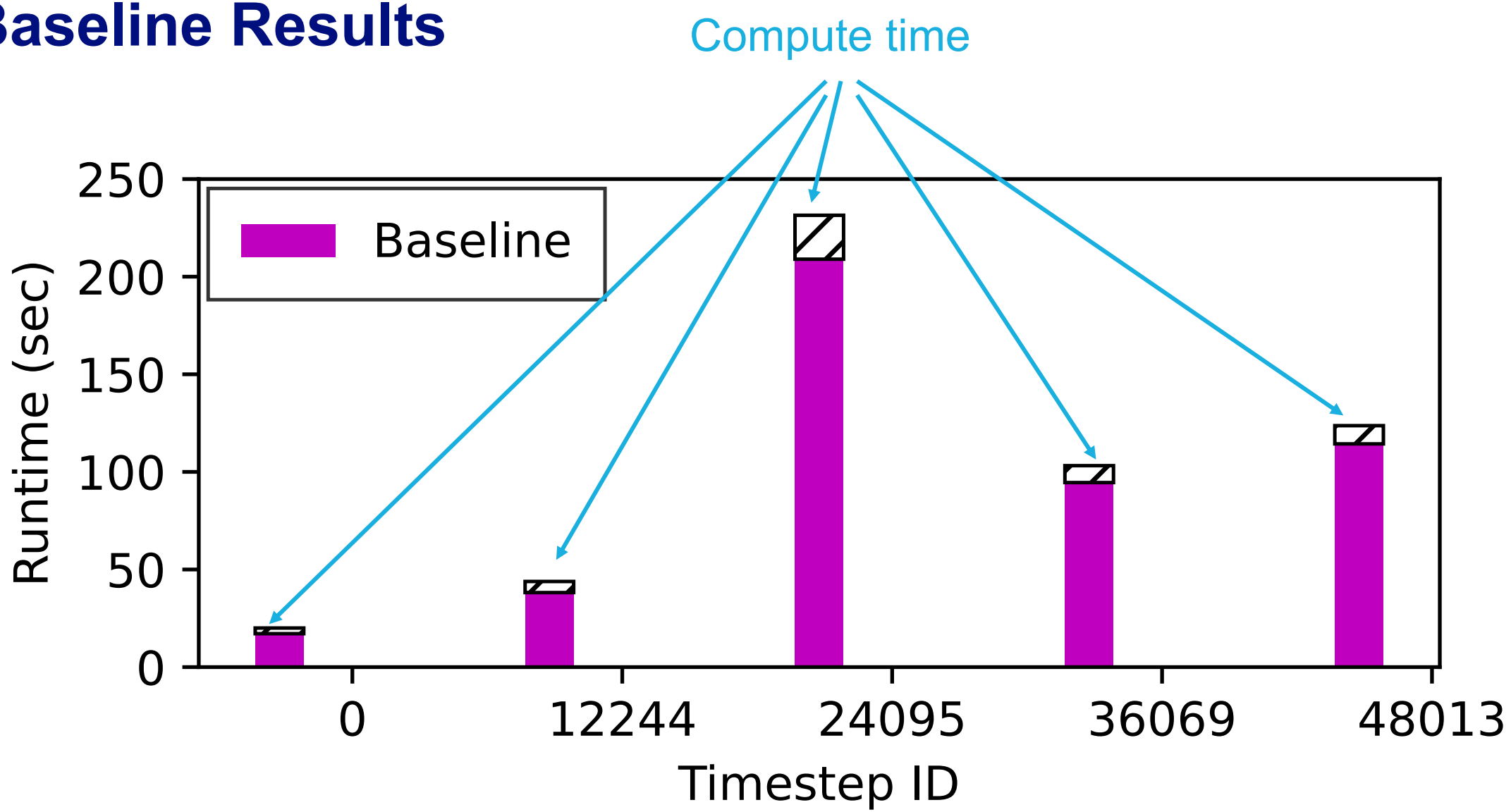
Pushdown Process #9



Experiment (asteroid-impact dataset)



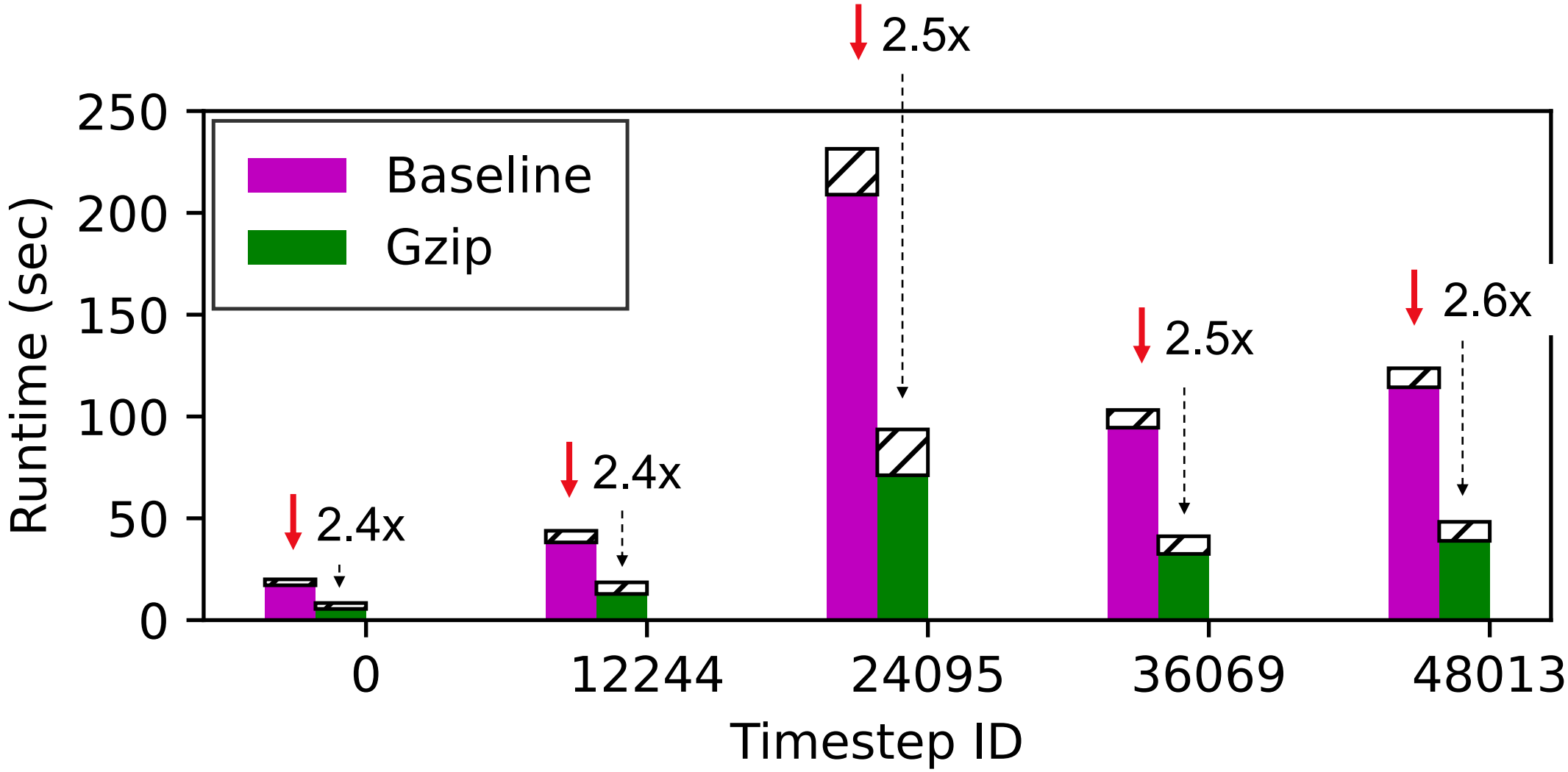
Baseline Results



I/O is a key bottleneck

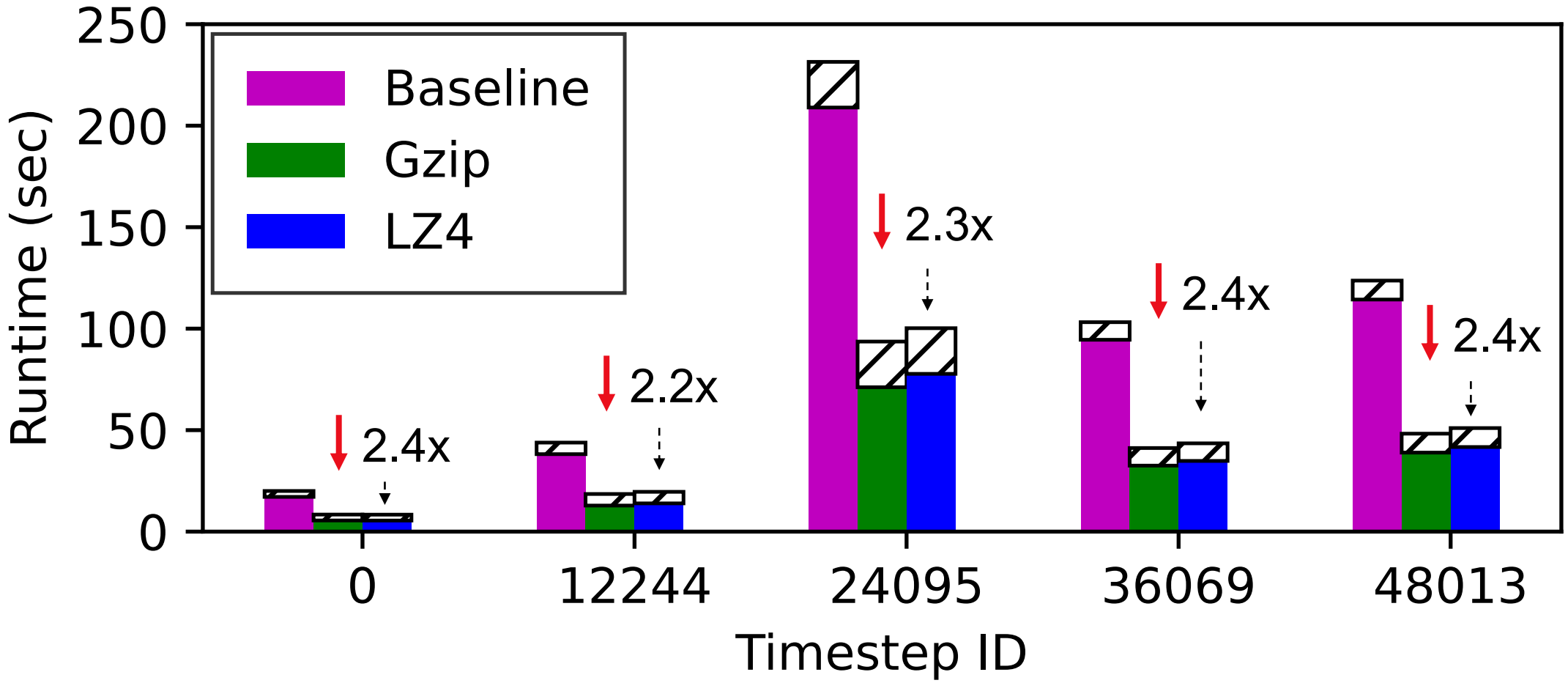
GZIP Compression

Speedups: 2.4 - 2.6x



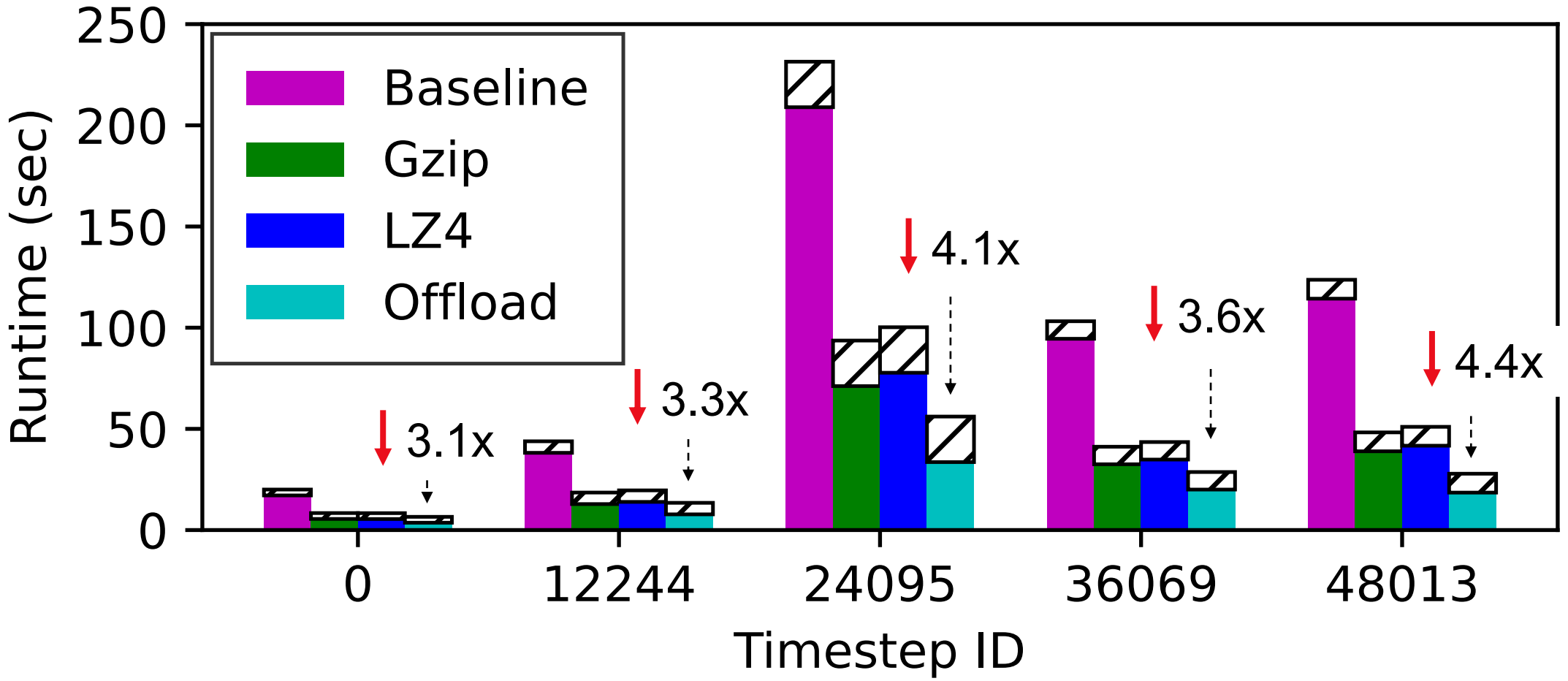
LZ4

Speedups:
GZ: 2.4 - 2.6x, **LZ4**: 2.2 - 2.4x

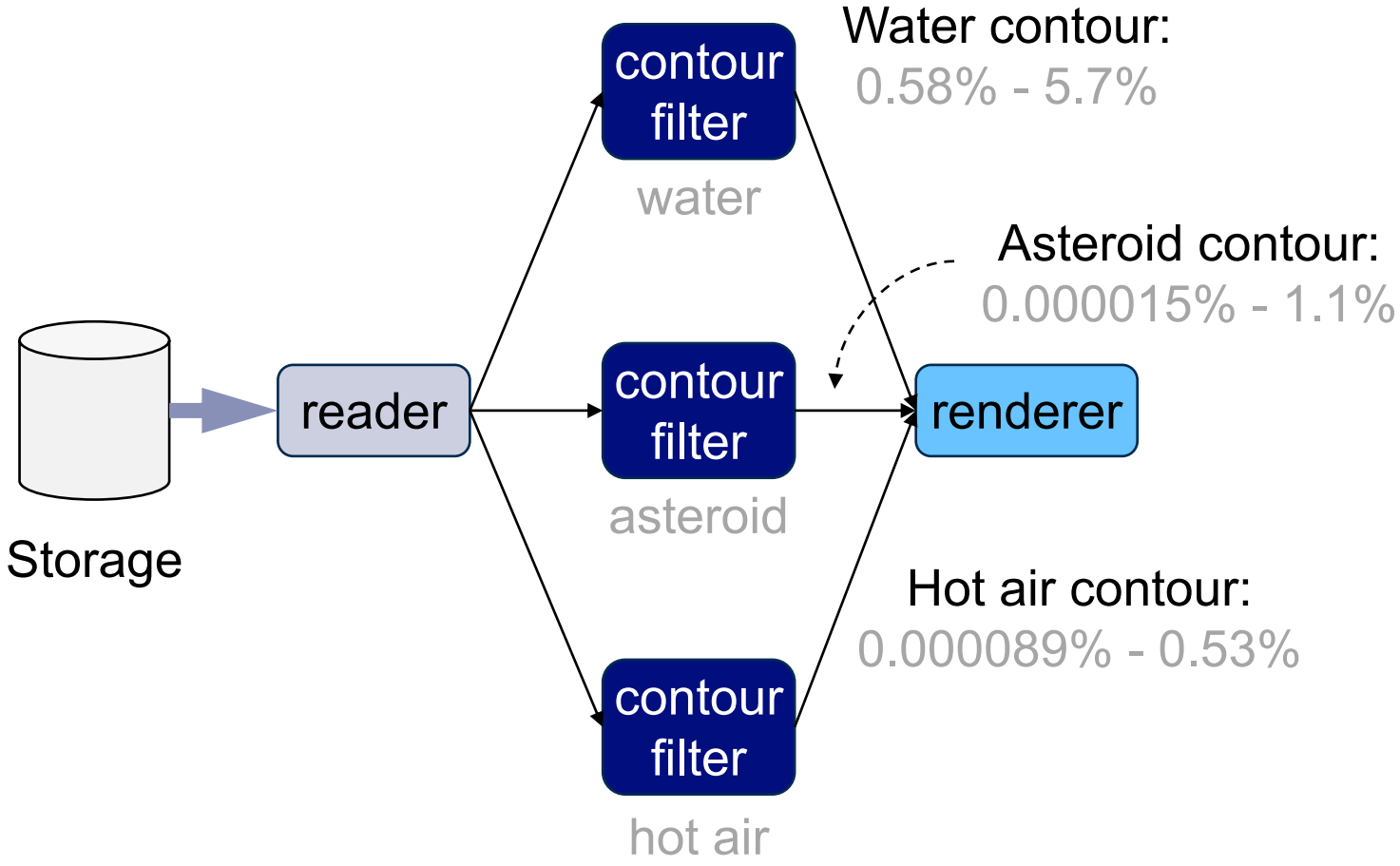


Offload Performance

Speedups:
GZ: 2.4 - 2.6x, **LZ4**: 2.2 - 2.4x
Offload: 3.1 - 4.4x



Speedup Increases with Selectivity



Water, asteroid, hot air

3.1 – 4.4x speedups

Hot air only

5.2 – 6.2x speedups

See paper for results on another real-world app

Conclusion

- Minimizing data movement is essential as datasets grow
- Pushdown lets visualization pipelines use their own selectivity to speed things up
- Pushdown and compression can be combined to reduce both local I/O and network transfer (useful for slower storage)
- The mechanism generalizes beyond visualization to broader analytics (e.g., SQL — see our e-poster)

Future work: resource management on data servers, more applications, …

Thank you!