Tuning the CMS Coffea-casa facility for 200 Gbps Challenge

Oksana Shadura, Andrew Wightman, Carl Lundstedt, Derek Weitzel, Garhan Attebury, John Thiltges, Kenneth Bloom, Sam Albin (University Nebraska-Lincoln)

Alexander Held (University of Wisconsin-Madison)

Benjamin Tovar Lopez (University of Notre Dame)

Brian Bockelman (Morgridge Institute for Research)

CHEP 2024, 19–25 Oct 2024

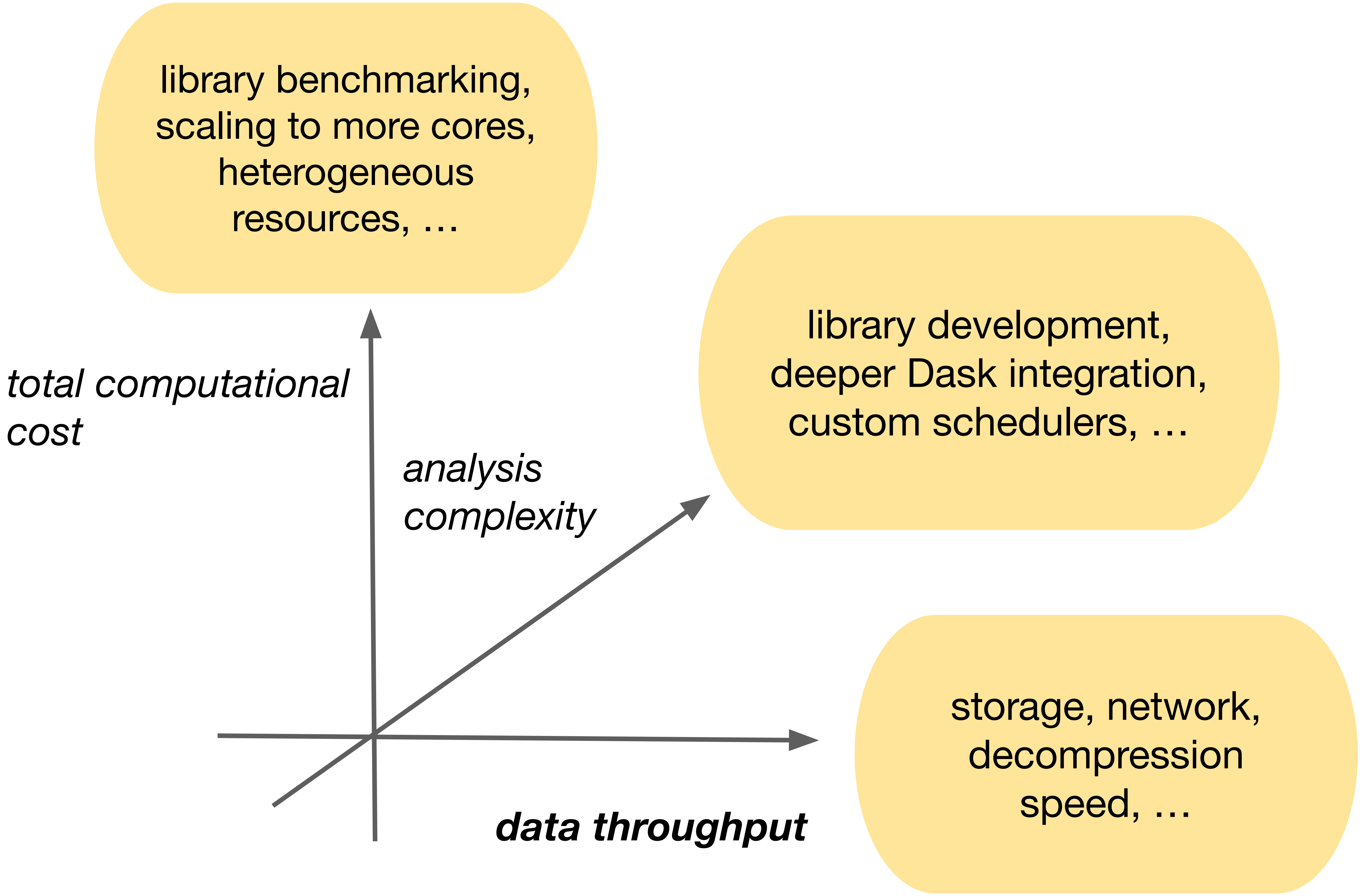
This work was supported by the U.S. National Science Foundation (NSF) cooperative agreements OAC-1836650 and PHY-2323298 (IRIS-HEP).

Analysis Grand Challenge and 200 Gbps

- The goal of the AGC is to test workflows designed for the HL-LHC by running representative analyses.
- **Limited agreement** in the broader field about how HL-LHC analysis will look like

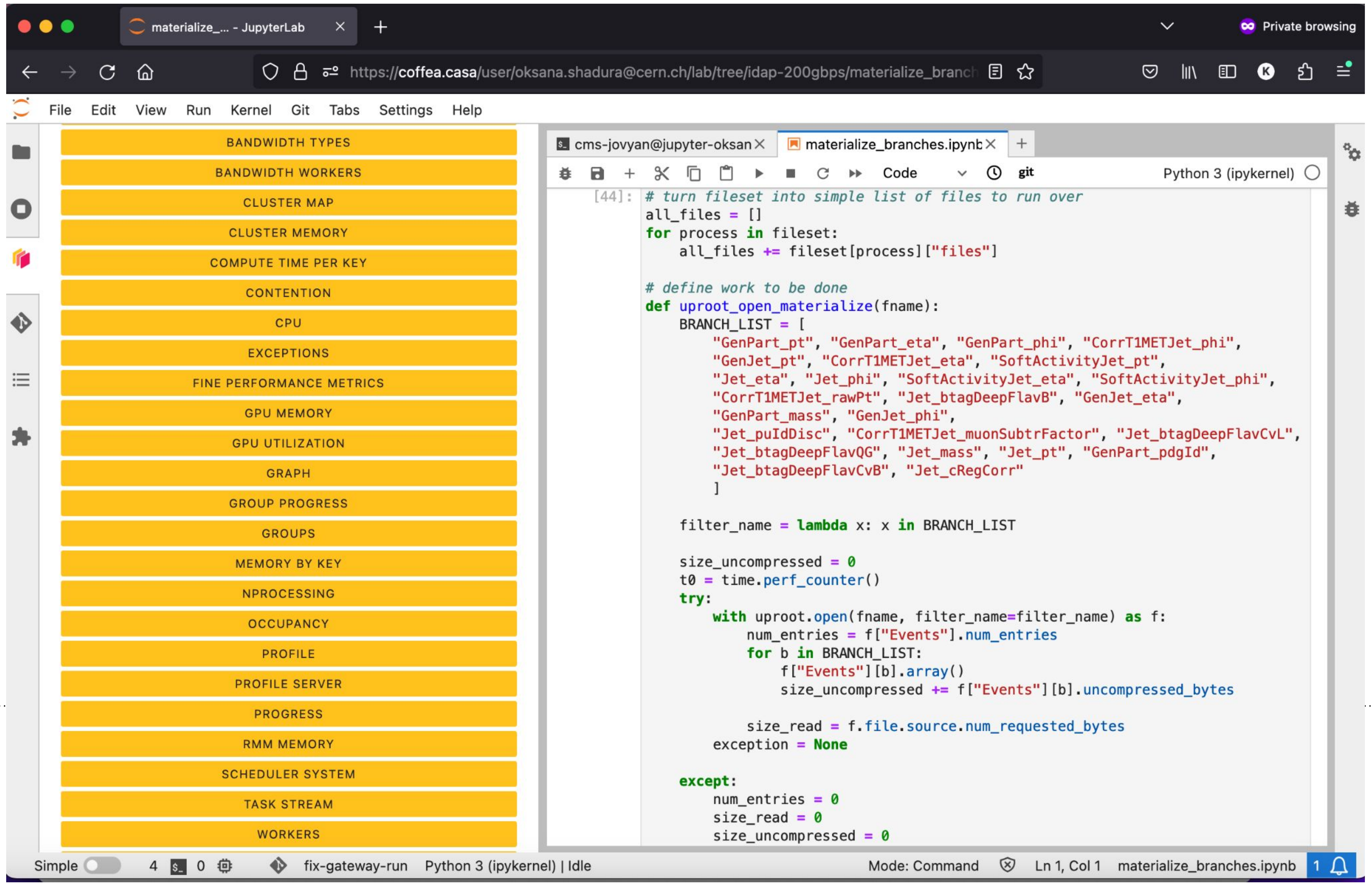
Main idea to factorize the challenges:

  - One of the **projects is focused on data throughput e.g. 200 Gbps**



Scaling to HL-LHC: the 200 Gbps setup

- **Simplified analysis setup** compared to what is done in *Analysis Grand Challenge (AGC)*
  - Analysis “straight from NanoAOD” with all required computations on-the-fly **using CMS Run 3 NanoAOD (90 TB x 2)**
  - **Implemented as** *Uproot and Coffea notebooks*
  - Goal to gradually add back functionality to match AGC



*The goal is to read **25% branches** out of **180 TB dataset** and to process it in **30 minutes***

In case of NanoAOD (~2 kB event size), the dataset size should be **90 B events** and **to analyze it at 50 MHz**

With current rate 25 kHz / core,

we will need 2000 cores

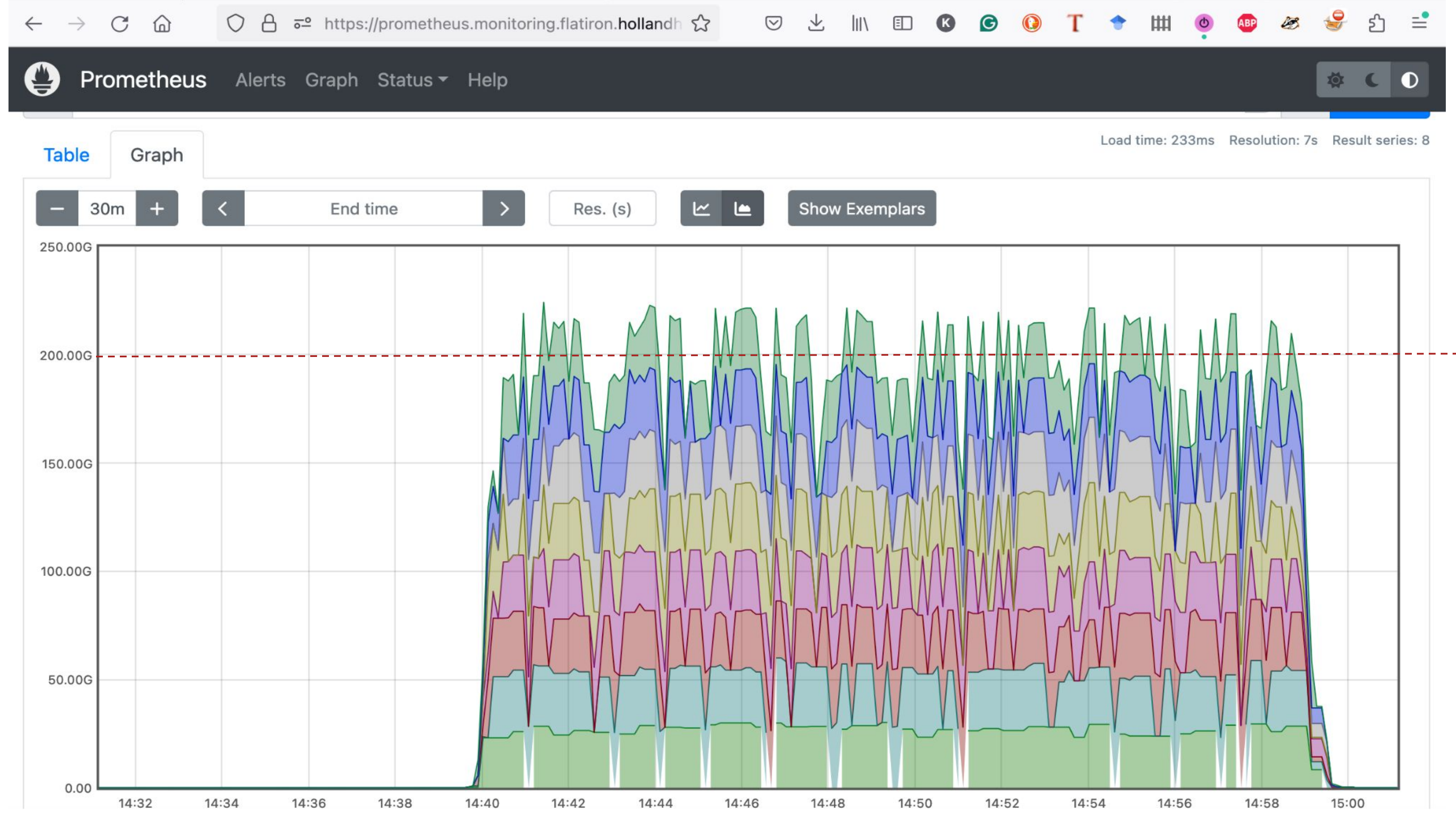
(12.5 MB/s per core)

This will require to tune facility setup to make sure we can reach the requested rate!

Preparing next generation of Analysis Facilities

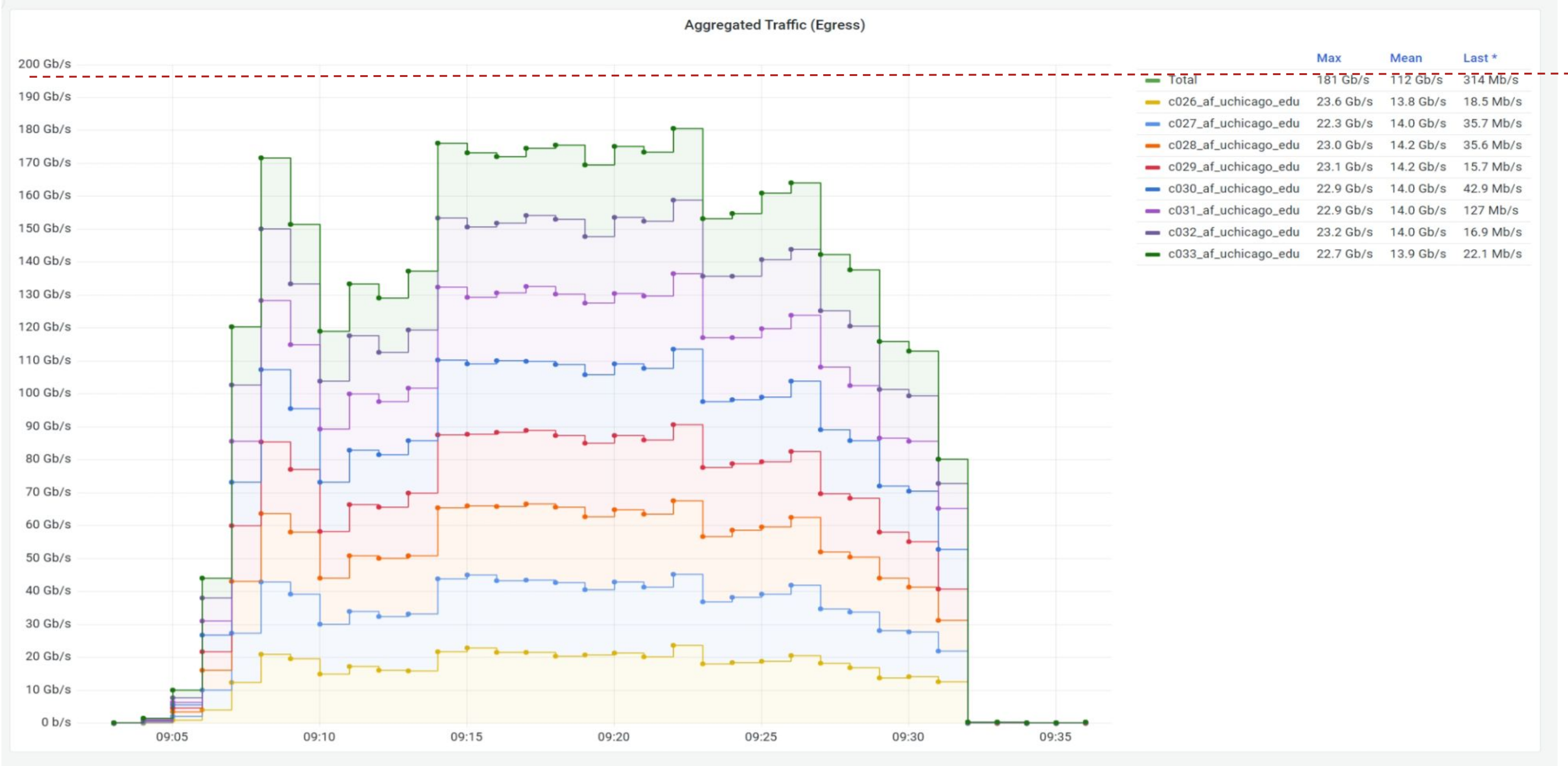
Adding additional services to improve analysis throughput

XCache - service provides caching of data accessed using xrootd protocol



200 Gbps

ServiceX - data extraction and delivery delivery service



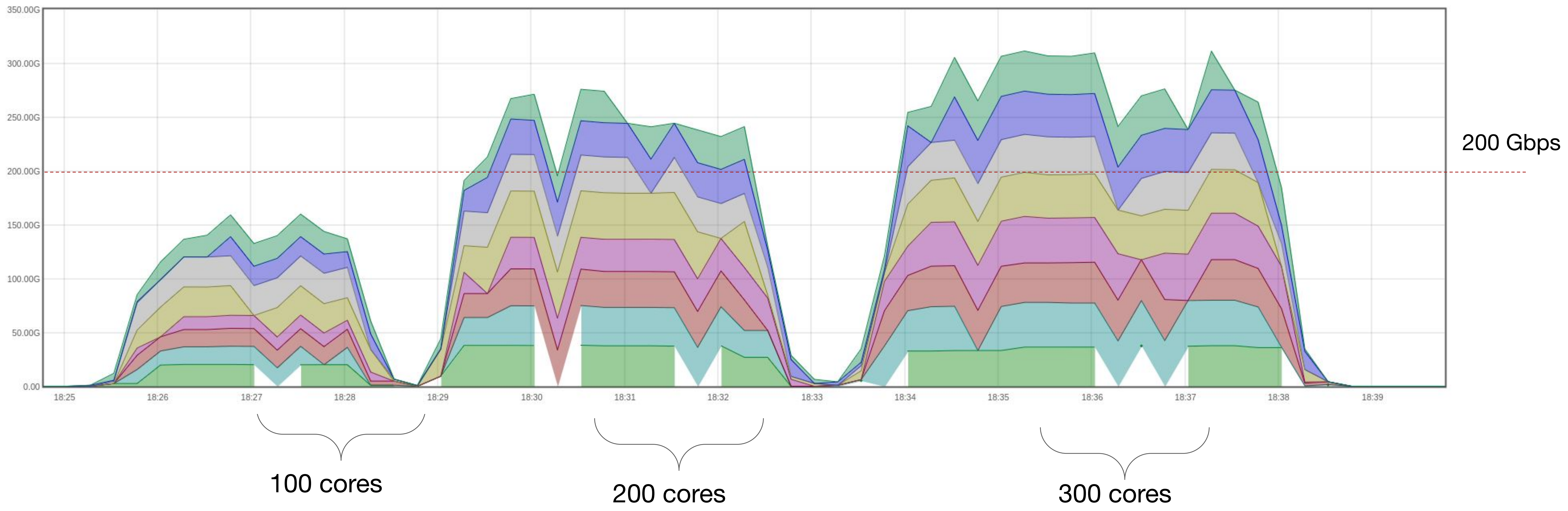
200 Gbps

(Servicex setup was covered in the talk of Rob Gardner)

XCache deployment at Nebraska

- Deployed in Kubernetes using charts
- Total setup *8 XCache servers with node affinity*
- XCache backed by NVMe storage
- Each XCache host has dual 100 Gbps networking to the Kubernetes core switch

xrdcp test on various number of Dask workers



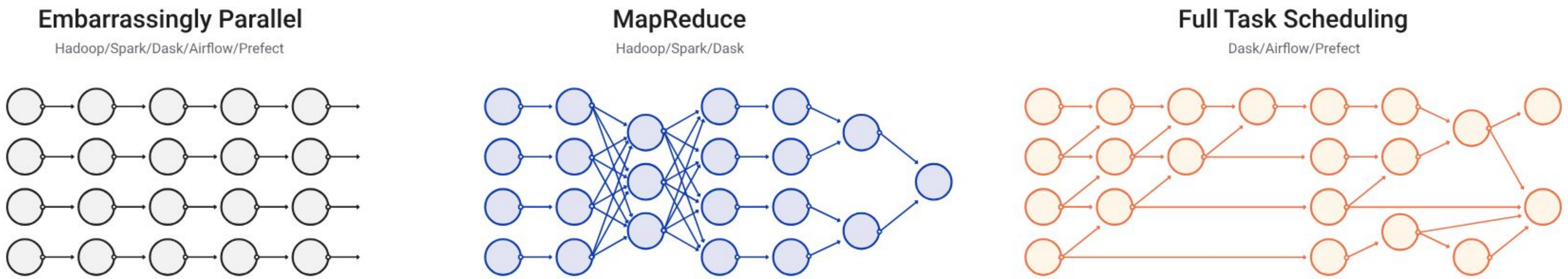
Preparing next generation of Analysis Facilities

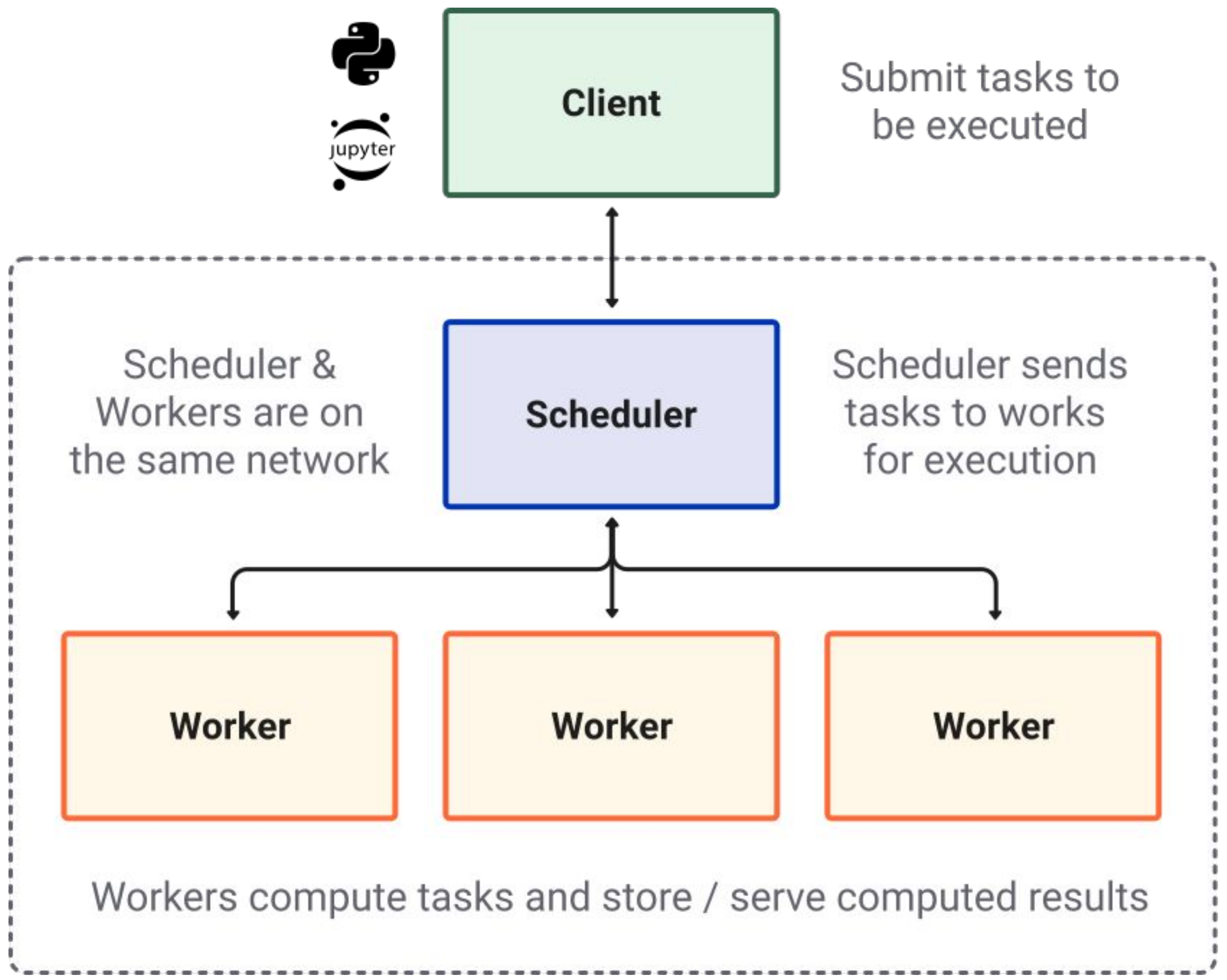
Running analysis frameworks and tools at scale

- **Adopt diverse computing executors** to support execution of complex task graphs
    - **Dask, TaskVine**

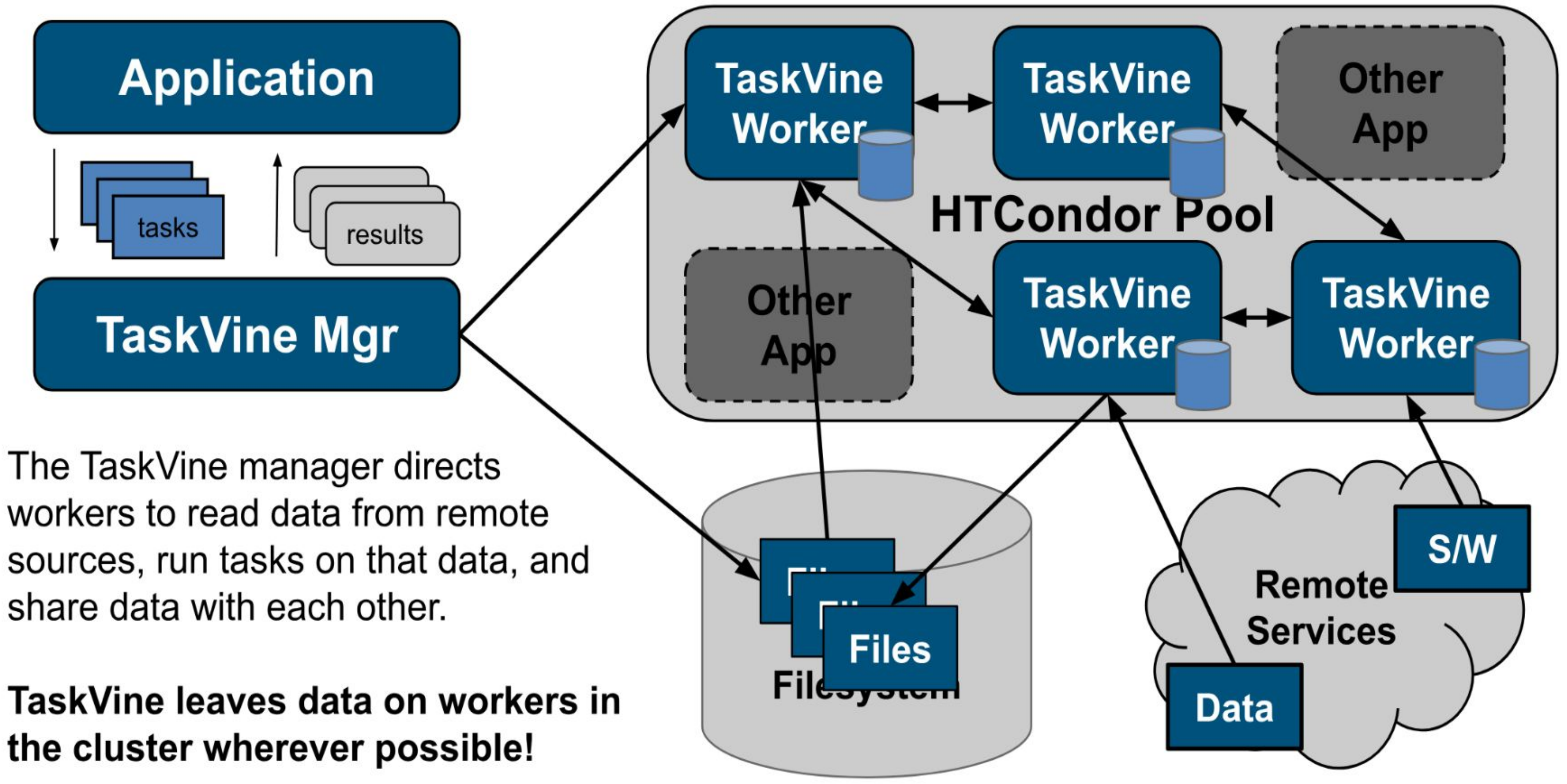
- Flexible computing **resource provisioning model** optimised for given facility
    - **Kubernetes, Tier-2 resources**, HPC cluster
        - e.g. Dask Gateway, dask-jobqueue, Dask Operator

Scaling with Dask and TaskVine





Dask Cluster



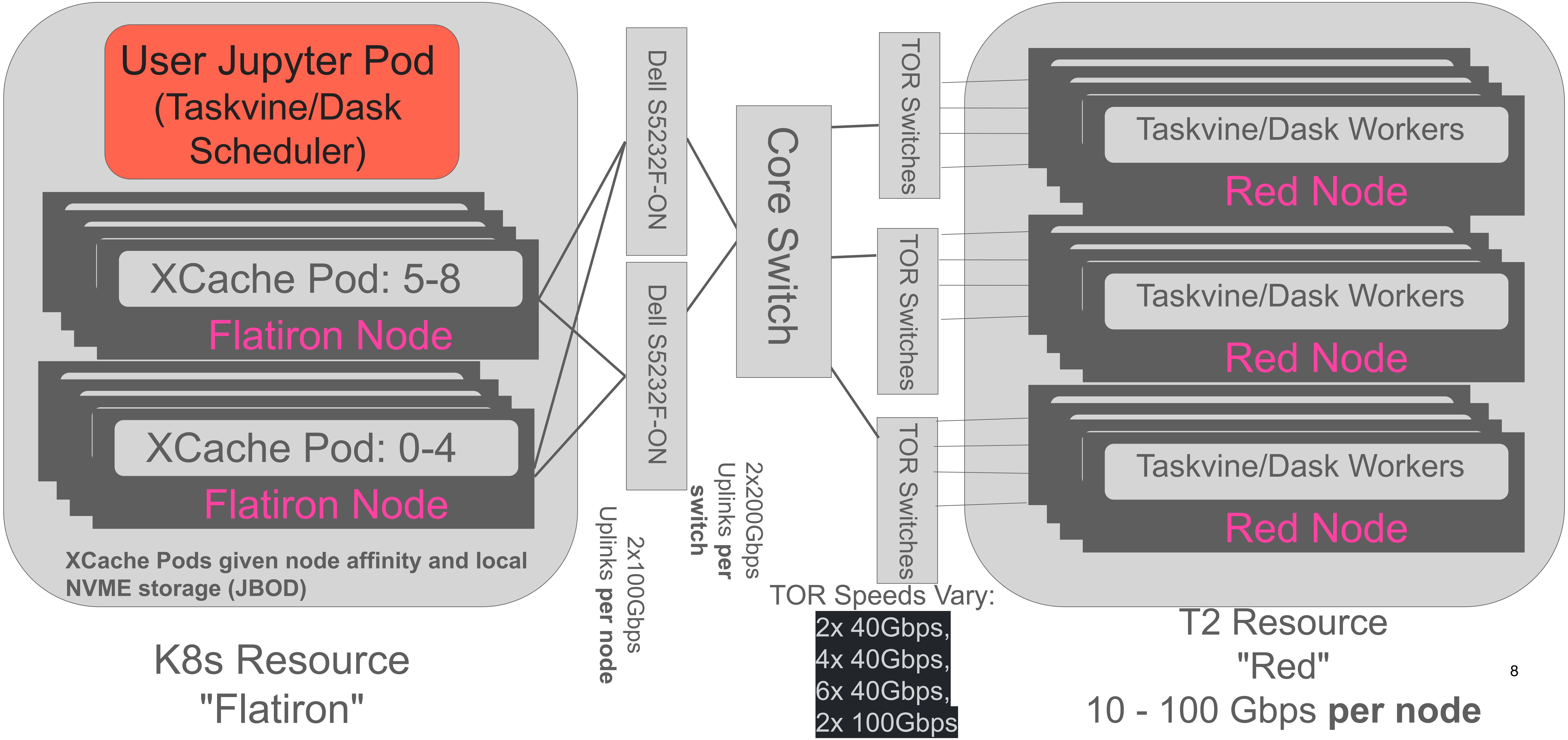
The TaskVine manager directs workers to read data from remote sources, run tasks on that data, and share data with each other.

TaskVine leaves data on workers in the cluster wherever possible!

[Dask Distributed]

Douglas Thain: TaskVine Overview | Video • Slides

Nebraska coffea-casa facility - HTCondor setup

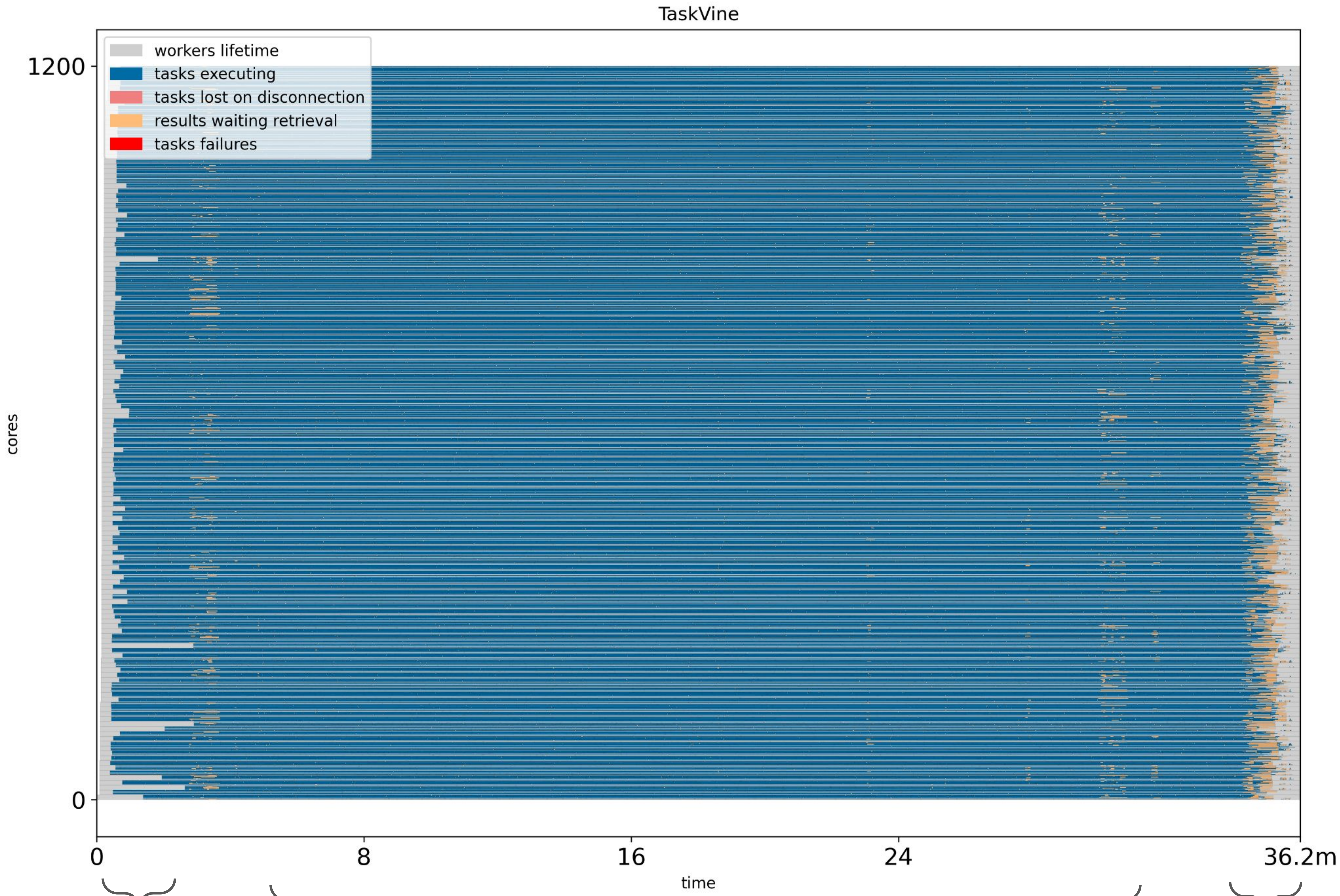


TaskVine stats running over 1200 cores

Data rate (Gbps)	
steady state	224.6 Gbps
overall	221.1 Gbps
max seen	240 Gbps
Time IO	35 min 30s
Time IO + accum results	36min 12s
Number of files	63,762
Files with errors	17
Total read (compressed)	58.33 TB
Total read (uncompressed)	139.03 TB
Total cores	1200 (150 8-core workers)
Core efficiency	92% (1114.67 cores)

event rate
(aggregated time
spent in function):
35.34 kHz

1200 cores across 150 8-core workers



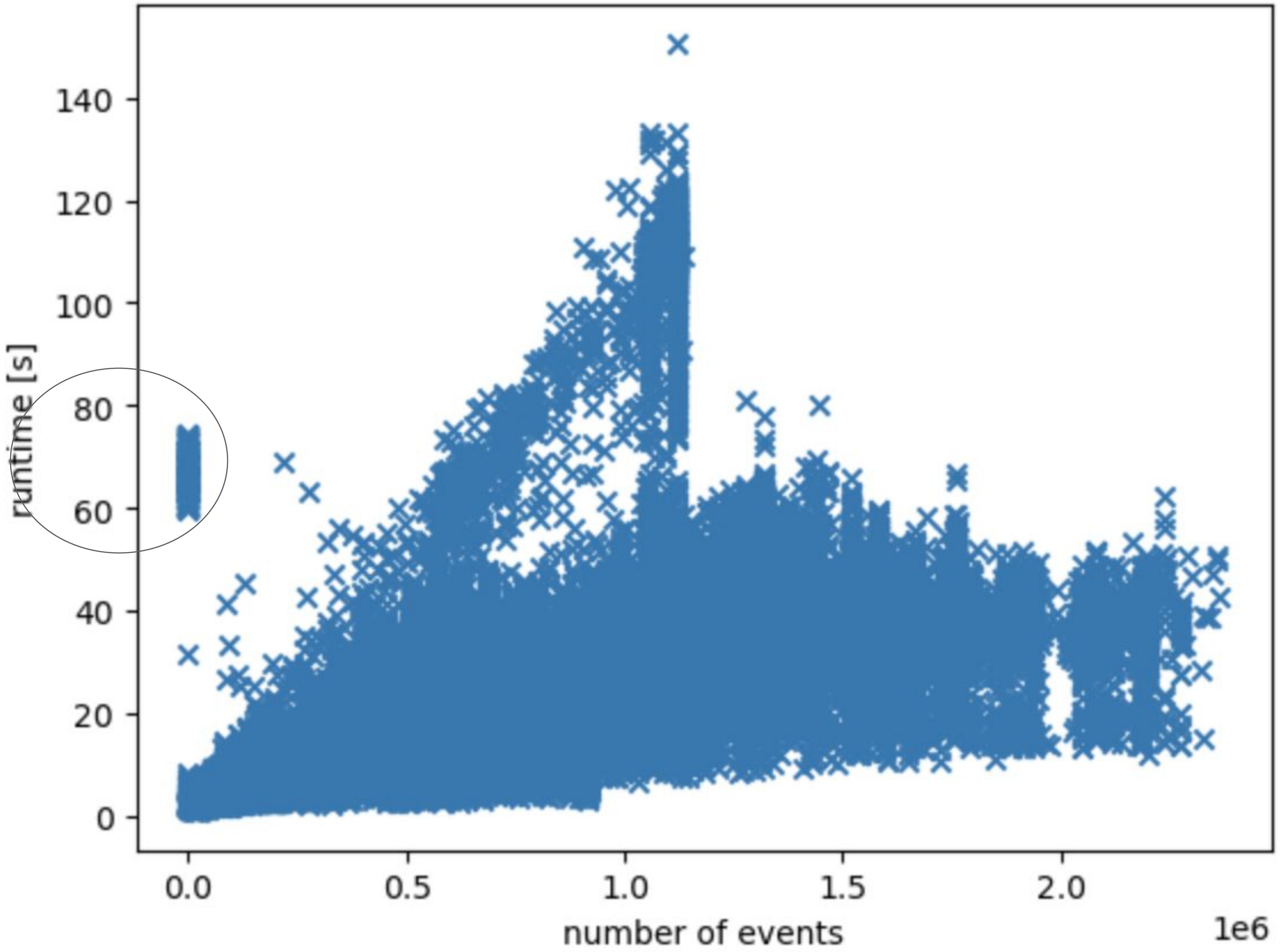
workers connecting +
setting conda env

IO + accumulation

accumulation
+ stragglers

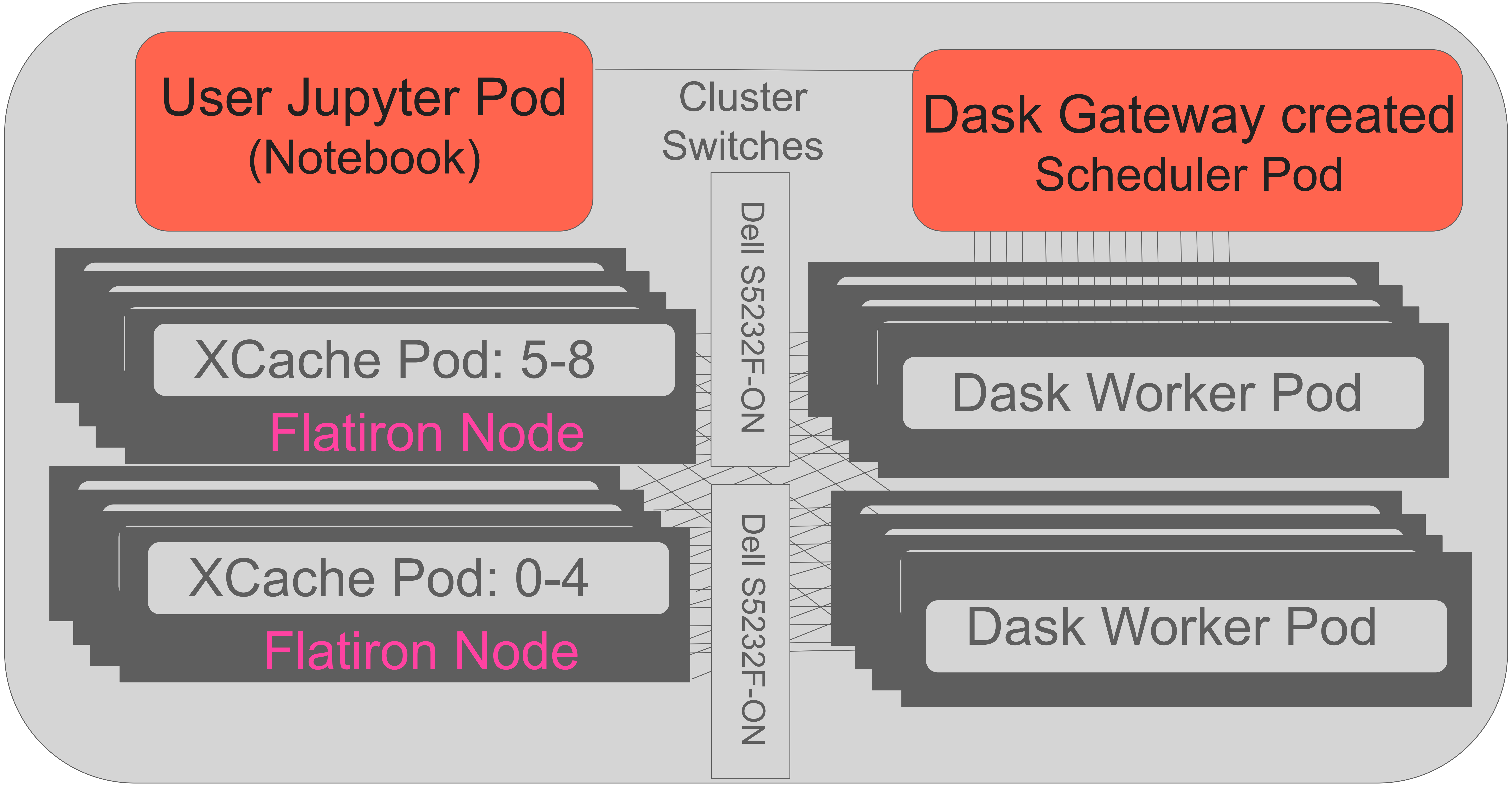
as seen by xcache

Dask + HTCondor stats running over 1300 cores: Rate over time and runtime to access each file



event rate (aggregated time spent in function): event rate (aggregated time spent in function): **27.66 kHz**

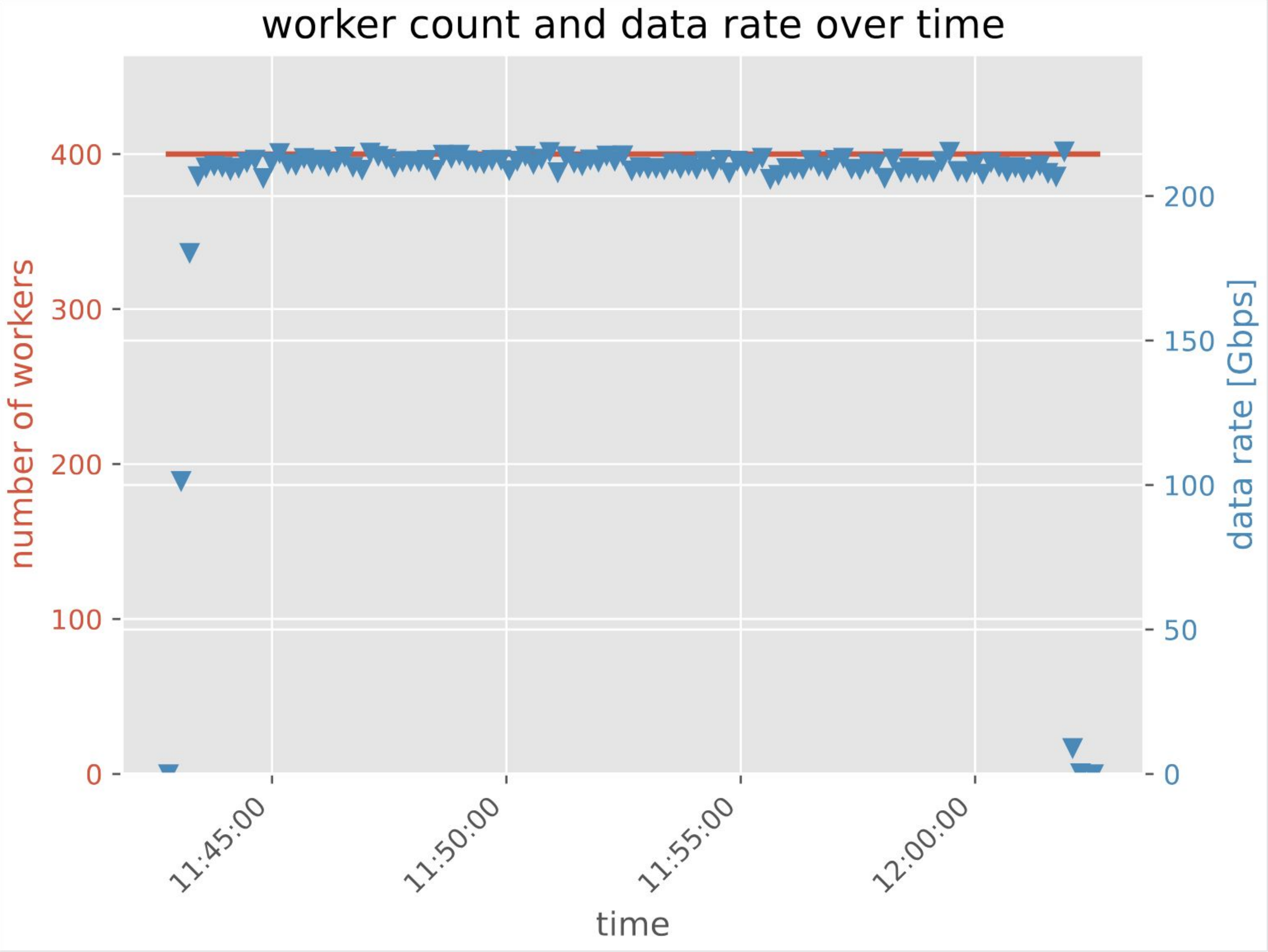
Pure Kubernetes facility: Dask Gateway Networking

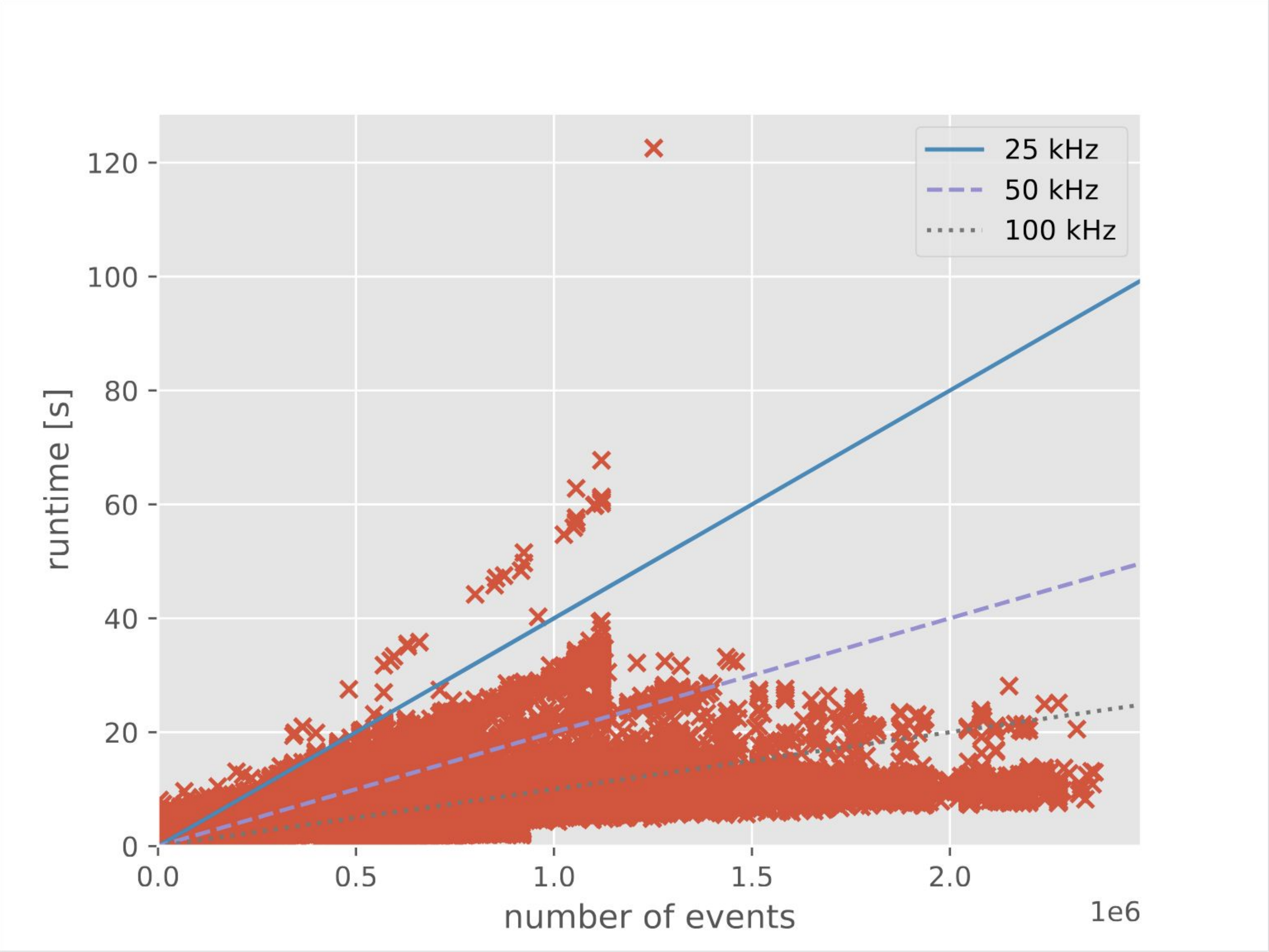


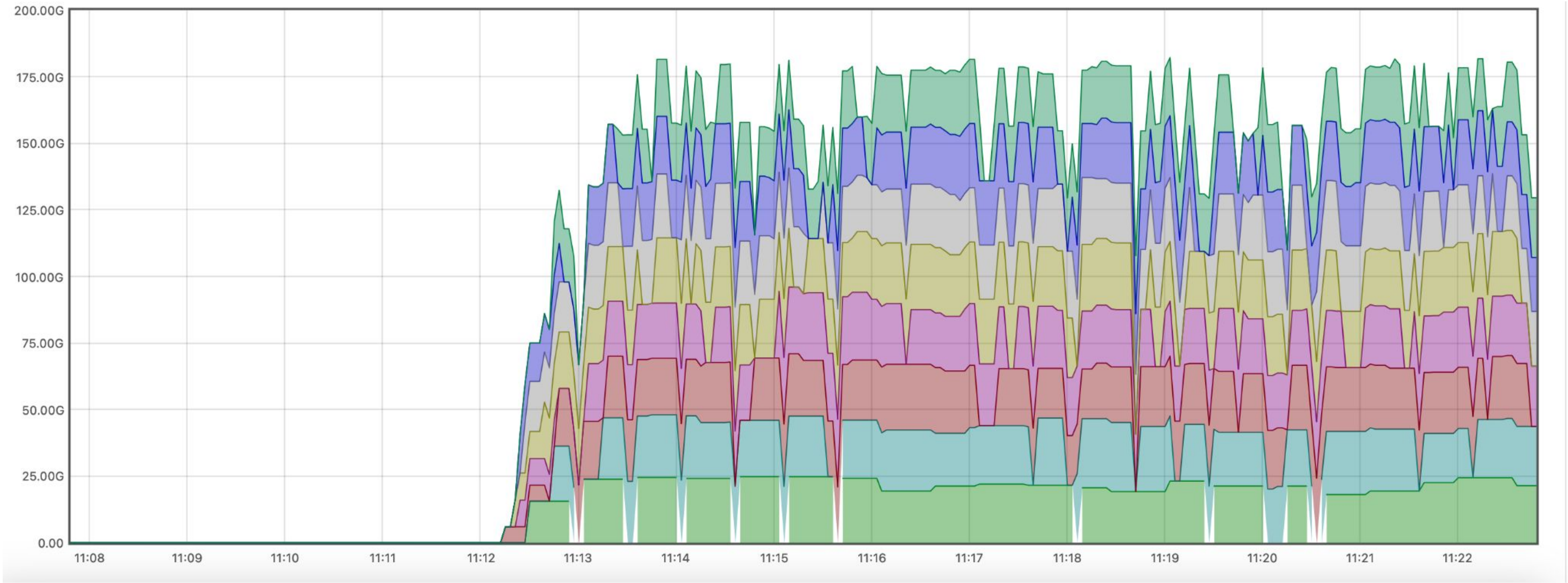
K8s Resource "Flatiron"

- 2 x 100 Gbps uplinks per hardware node
- XCache k8s pods given node affinity and local NVME storage (JBOD)

Dask + Kubernetes running over 400 workers: Rate over time and runtime to access each file

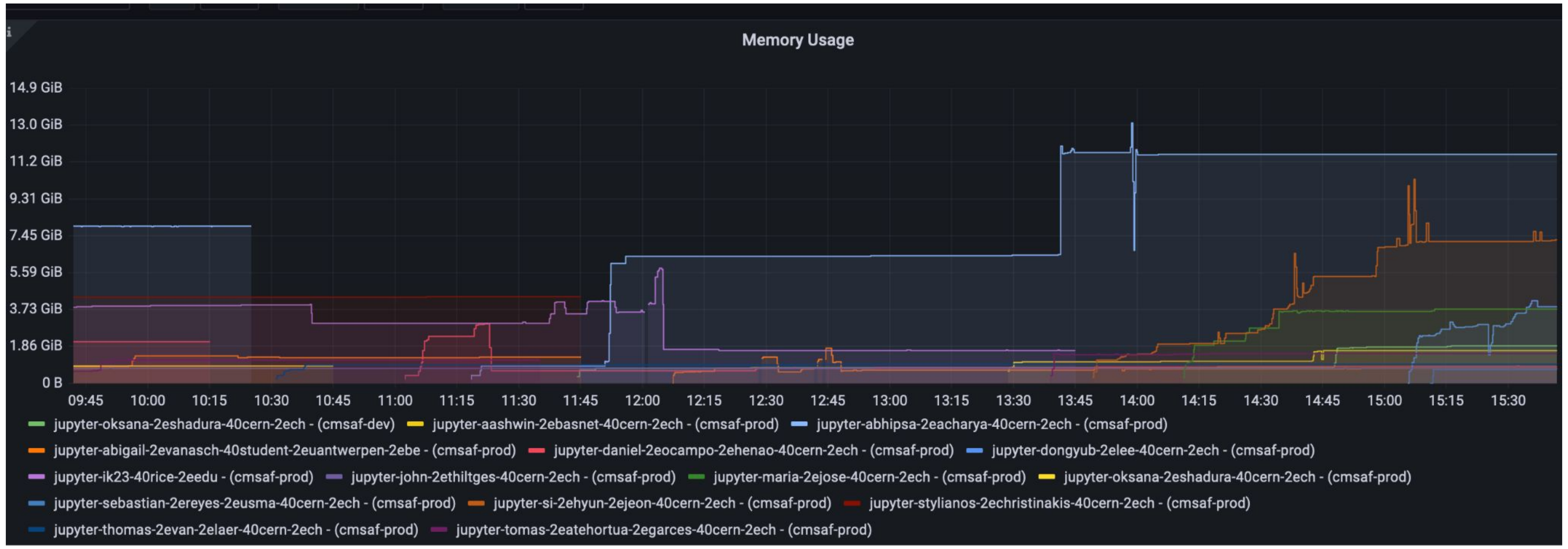




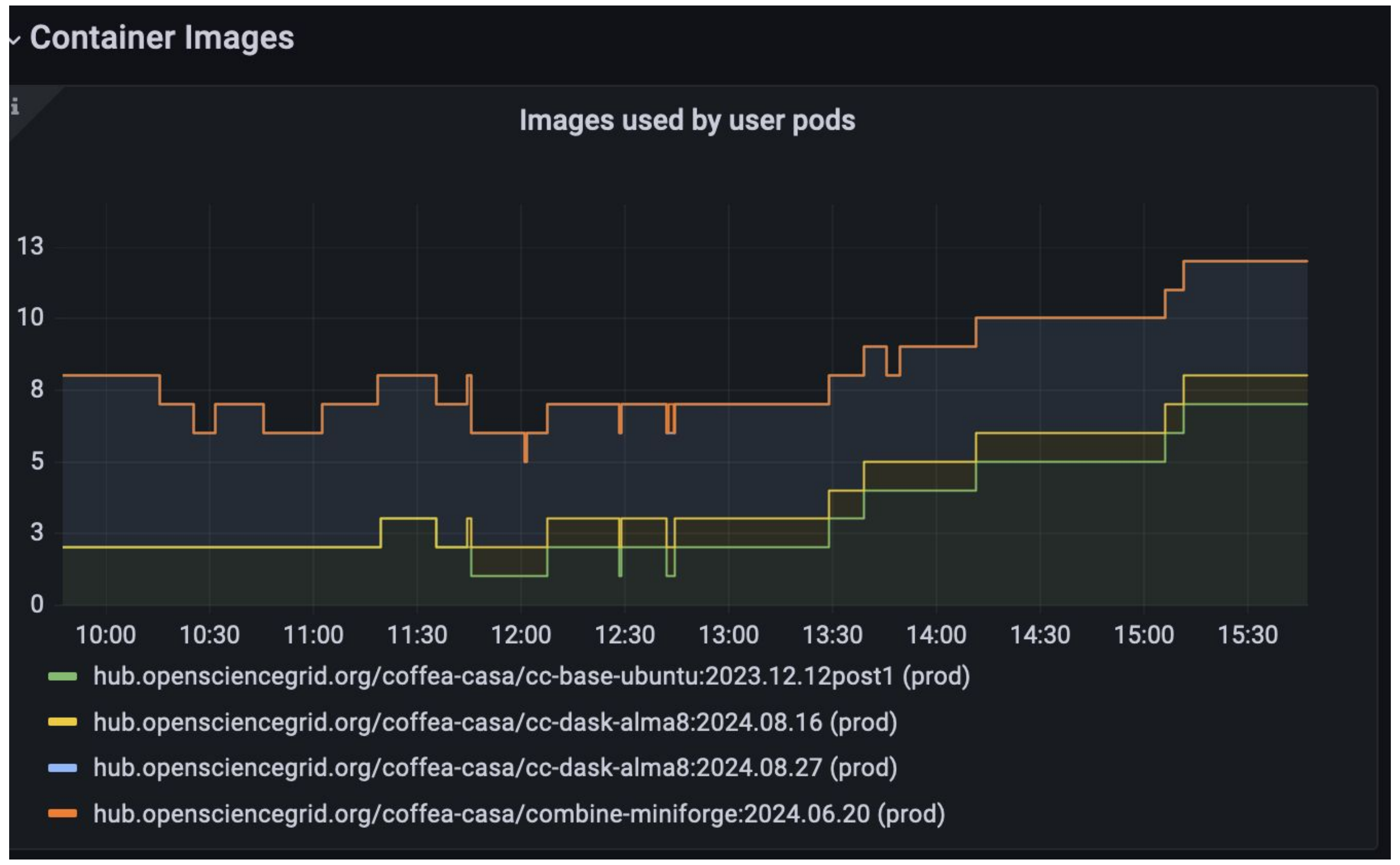


Monitoring

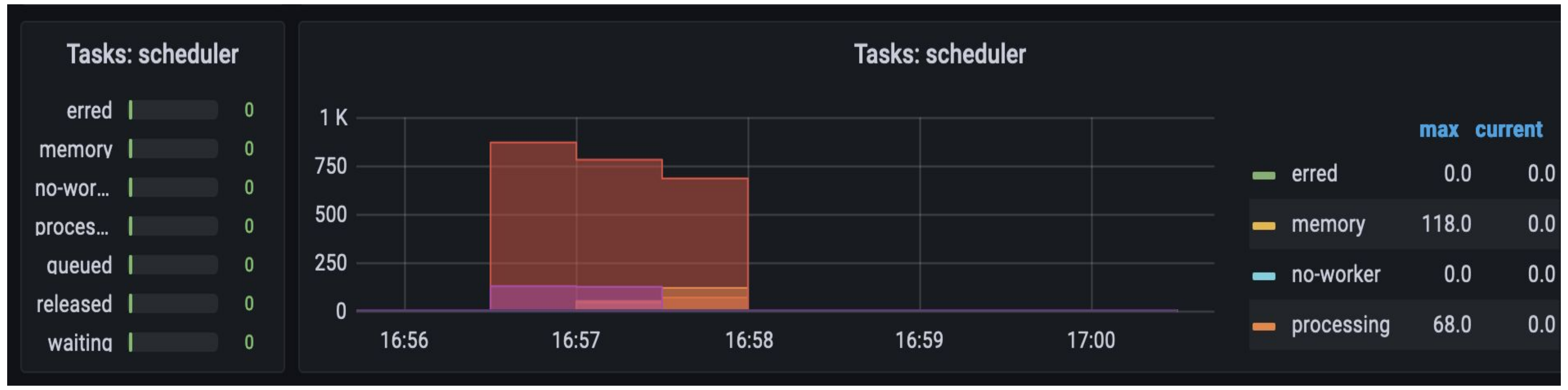
analyzing resource usage patterns (all users)



checking popularity of images between users



tracking dask worker allocation and usage patterns (per user)



(Some of the) lessons learned

- Very **successful exercise format**: huge amount of progress and activity within 8 weeks
- Faced some challenges with memory use and scaling to all available resources
- **NanoAOD:** very large effect of compression algorithm: switching from LZMA to ZSTD brought 2.5x event processing rate improvement
- **Scaling Dask to 2k+ workers** generally works fine, need more testing combining large numbers of workers and very complex graphs
- Good performance observed also with **TaskVine** as alternative scheduler for graphs
- Scale of challenge allowed to identify **new bottlenecks** (many of which have already been fixed)

This presentation summarizes a large body of work across IRIS-HEP and USCMS:

- Fermilab: Lindsey Gray, Nick Smith
- Morgridge: Brian Bockelman
- Notre Dame: Ben Tovar
- Princeton: Jim Pivarski, David Lange
- U. Nebraska: Sam Albin, Garhan Attebury, Carl Lundstedt, Ken Bloom, Oksana Shadura, John Thiltges, Derek Weitzel, Andrew Wightman
- U. Wisconsin: Alex Held (co-coordinator)