Analysis Facility Development at Wisconsin CMS T2

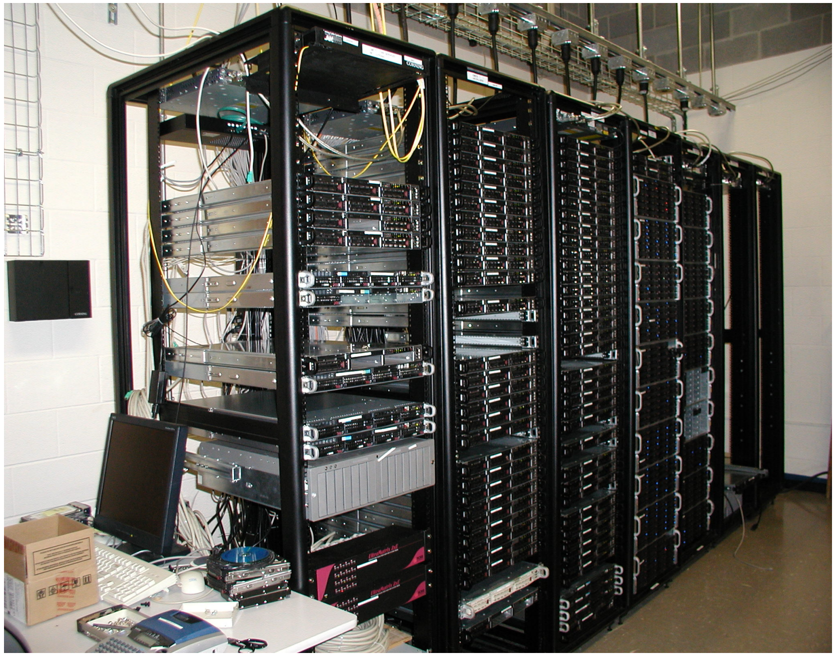
T. Bose, D. Bradley, S. Dasu, A. Mohapatra, C. Seys, C. Vuosalo
(**HEP Computing Group**)

Outline

- Infrastructure
- Analysis with Run1, Run2, and Run3 data
- HL-LHC Computing Resource Needs for CMS
- Expected Analysis model for Run4 and beyond
- Analysis Facility Setup at Wisconsin
- Summary

Infrastructure

- ✓ Became a CMS dedicated T2 computing facility in 2005
- ✓ 3 machine rooms, 17 racks full of hardware
- ✓ Enough power supply (650kW) and cooling is provided by the campus

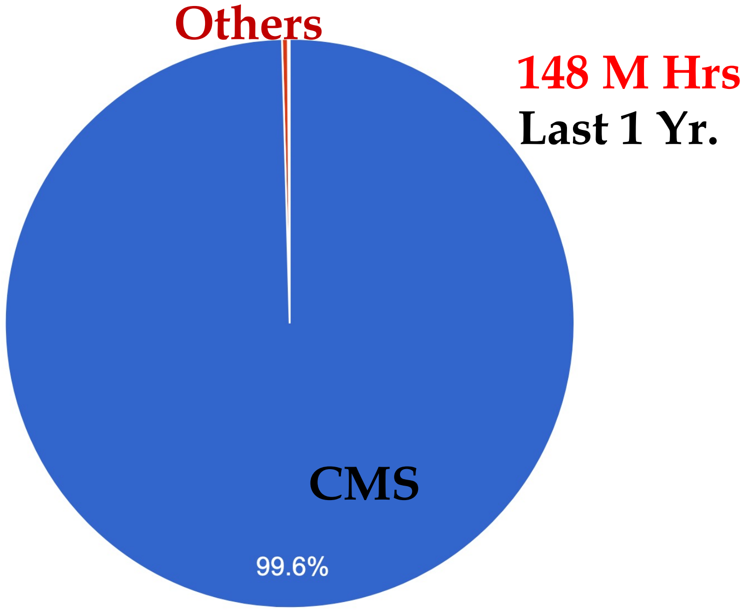


Compute / Storage Resources

✓ Compute (Alma9 OS)
- T2 HEP Pool – @17000 CPU cores
- HS06/HepScore23 → ~200k
- Nvidia T4 GPUs (36)

✓ Storage (HDFS - 3.3.6)
- In use for 13+ years
- Now @12 PB raw
- Supports Erasure Coding (EC) and Replication

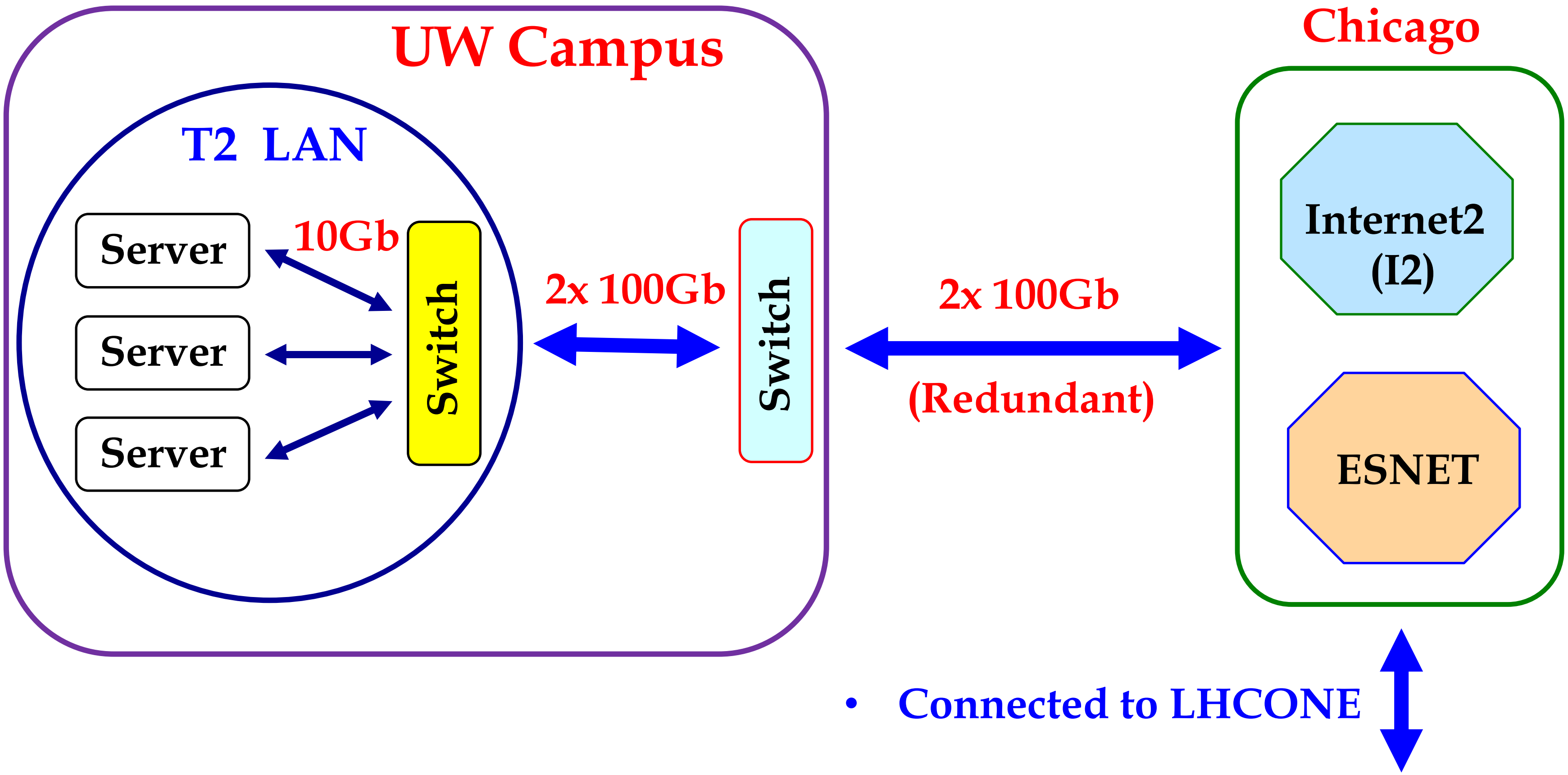
✓ All nodes provide compute and storage services



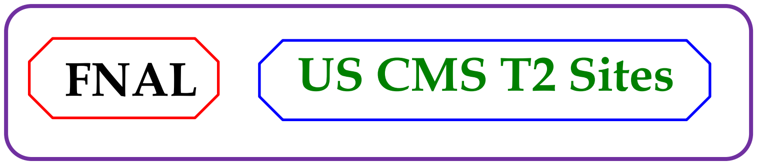
Software and Services

- File systems, authentication, & proxy service
  - AFS, NFS, CVMFS, Kerberos, Frontier/Squid
- Job batch system
  - HTCondor
- Open Science Grid (OSG) software stack
  - HTCondor-CE and various middleware
- Storage and Data Access
  - Hadoop (hdfs) and XRootD
- Cluster management & monitoring
  - Puppet, Local Yum Repo(Ceph), Ganeti, Nagios/Icinga, Prometheus, Grafana

IPv4 and IPv6 Network



- Connected to LHCONE



- UW Campus to ESNET/I2 upgrade plan – 4x 100Gb

Analysis of Run1, Run2 and Run3 data

Analysis Workflow in Run1, 2, and 3

- Data formats: RAW → RECO →AOD (Run1, 480kb/evt) → MiniAOD (Run2, 50kb/evt) → NanoAOD (Run3, 1-2kb/evt). Centrally produced.
- Mini/NanoAOD → suitable user ntuples using analysis support tools i.e. CRAB etc.
- Ntuple processing with user code at the T2/T3s → output for statistical analysis and plots → physics publication.
- MiniAOD and NanoAOD formats significantly reduced storage and cpu resource usage, and user ntuple production with overlapping information.

CMS Remote Analysis Builder (CRAB)

- CRAB was designed by CMS offline and computing support group.
- It allowed physicists to transparently access the global resources of the WLHC Computing Grid.
- CRAB2 was used during LHC Run 1 (2009-2012). Much improved CRAB3 was used during LHC Run 2 (2015-2018).

- CRAB2 limitations experienced by Wisconsin users prompted us (at Wisconsin T2) to develop a less complex analysis support tool (called Farmout).

Farmout Scripts

- ✓ Developed by Wisconsin CMS computing support group – used primarily by Wisconsin T2 affiliated users.
- ✓ Supported users with private MC production and data analysis using Wisconsin T2 and campus computing resources. Output is stored in the T2 storage.
- ✓ Input data was accessible either from the local CMS T2 storage or global CMS T1/T2 sites via XRootD services.
- ✓ Became popular due to efficient and timely user support from local experts.

Data Analysis in the HL-LHC Era

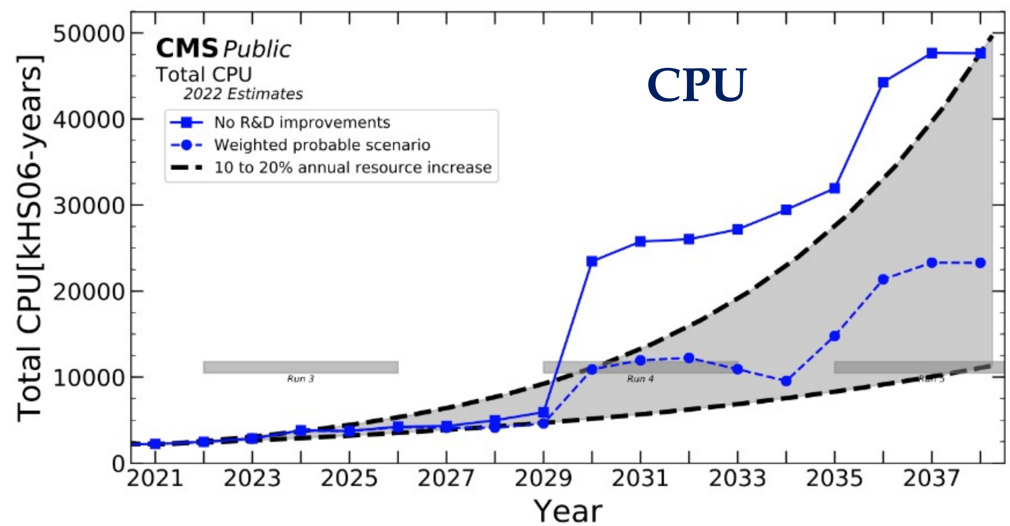
- Analysis scenario is expected to be different for Run4 and beyond due to significantly high data volume and the large computing resources needed to process it for physics.

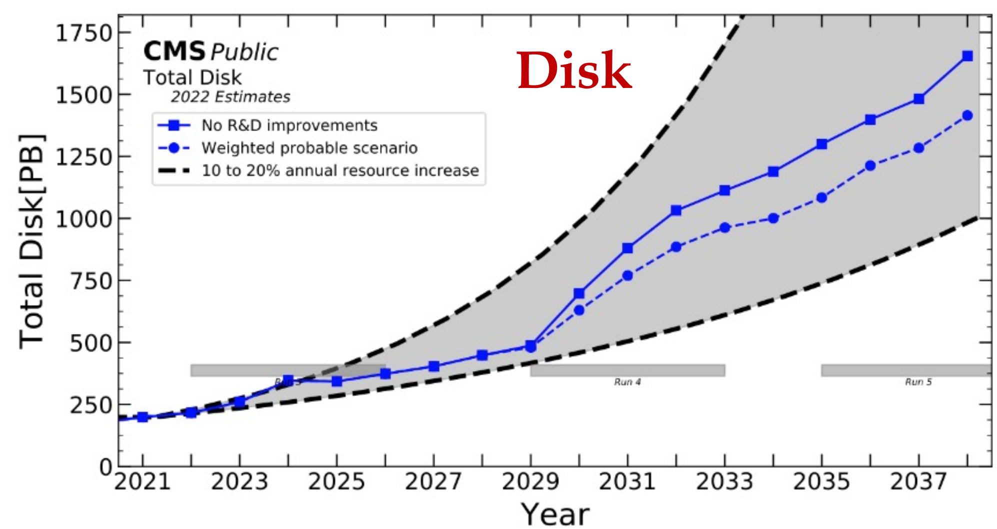
- A combination of much needed hardware resource provisioning, significant software improvements, and innovative analysis methods will be the key.

HL-LHC Computing Resource Needs for CMS

HL-LHC Computing Needs for CMS

- Higher Intensity Proton-proton collisions, and new CMS detector with more channels
- 3x more events to be processed each year → 150 Billion events
- 5x increase in event size → disk storage needs will reach 0.5 exabyte by 2030





How To Meet the Computing Needs

- ## Hardware
  - Maintain scalability for computing facilities allowed within flat budget scenario.
- ## Software
  - Modernize Physics Software and Improve Algorithms using ML/AI, GPUs, accelerators, etc.
  - Build Infrastructure to produce, archive, store, transfer, and provide fast access to Exabyte-Scale Datasets.
  - Transform the Scientific Data Analysis Process.
  - Seize on Industry Advances in Data Science i.e. Use Columnar Data Model and Tools for Analysis.
  - Prepare Highly Efficient Analysis Facilities.

Expected Analysis Model for Run4 and Beyond

Columnar Data Analysis Model

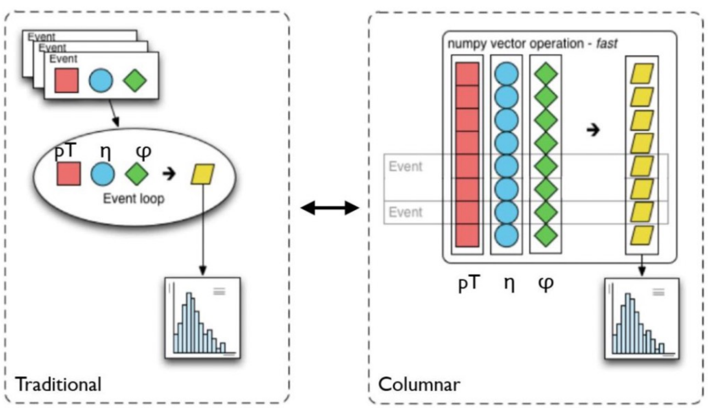
- Switch to fast columnar data processing using highly efficient scientific python packages used in Data Science Industry.
- Translate NanoAOD root branches into awkward arrays and massively parallelize array processing to scale.
- Perform analysis quickly, correctly, and efficiently using the available computing resources.
- (Re)produce physics output in hours instead of weeks by eliminating intermediate steps i.e. skimming etc.

↓

✓ Use "Columnar Object Framework For Effective Analysis (Coffea)"

Columnar Analysis Building Blocks

- Coffea
  - A user interface for processing CMS NanoAOD data
  - Makes use of other packages in the scientific python ecosystem for efficient data processing
  - Works with multiple Task Schedulers



Coffea Analysis Framework

ROOT RDataFrame

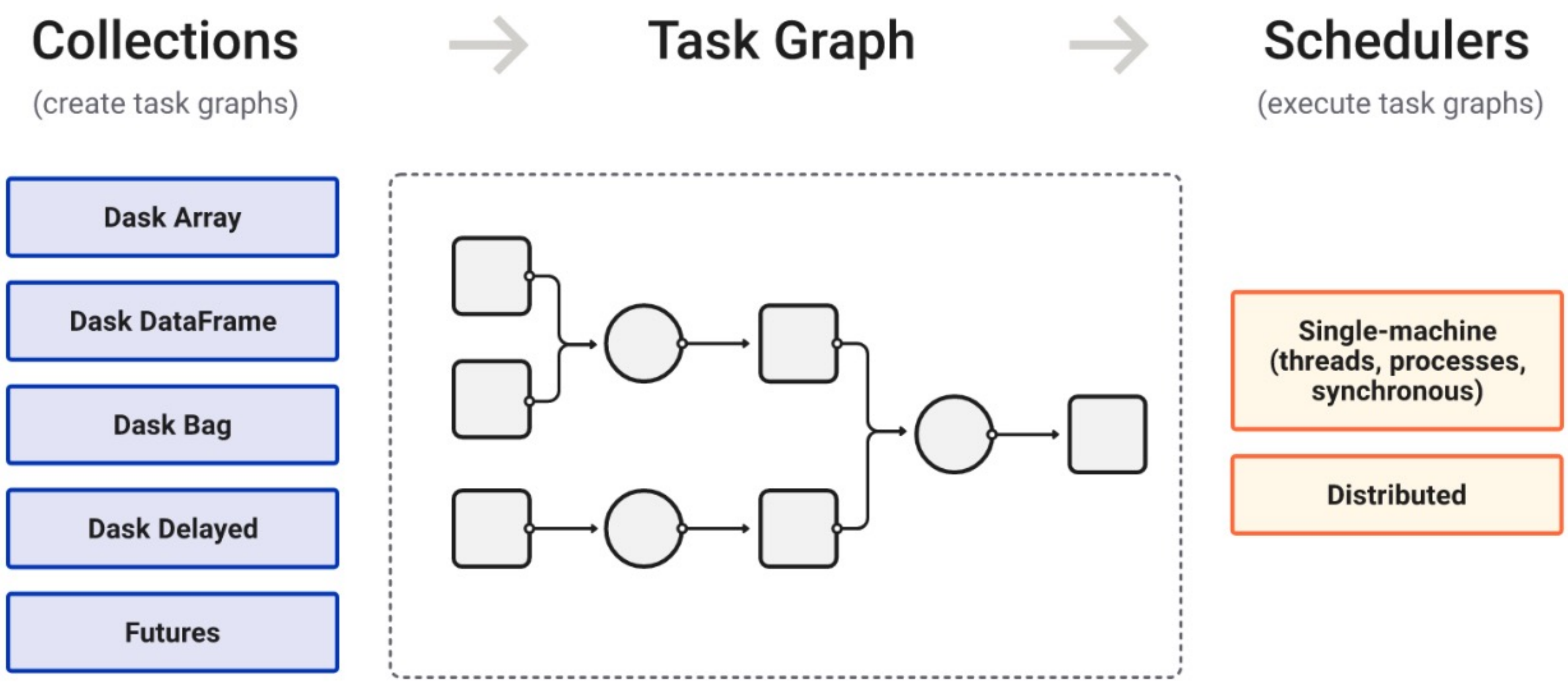
Columnar data analysis concept

Analysis Framework

Task Scheduler

Dask – Task Scheduler

- Dask is a python library for parallel computing
- It scales python code from multi-core local machines to large distributed clusters in the cloud
- Suitable for processing larger-than-memory datasets using Task Graphs



Analysis Facility Setup at Wisconsin

Analysis Facility Setup at Wisconsin

THE UNIVERSITY of WISCONSIN MADISON

- Setup 1 → Analysis with Coffea + Dask using python scripts through a SSH terminal (traditional method)
- Setup 2 → Analysis with Coffea + Dask through Jupyter notebooks managed by a multiuser Jupyterhub installation

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

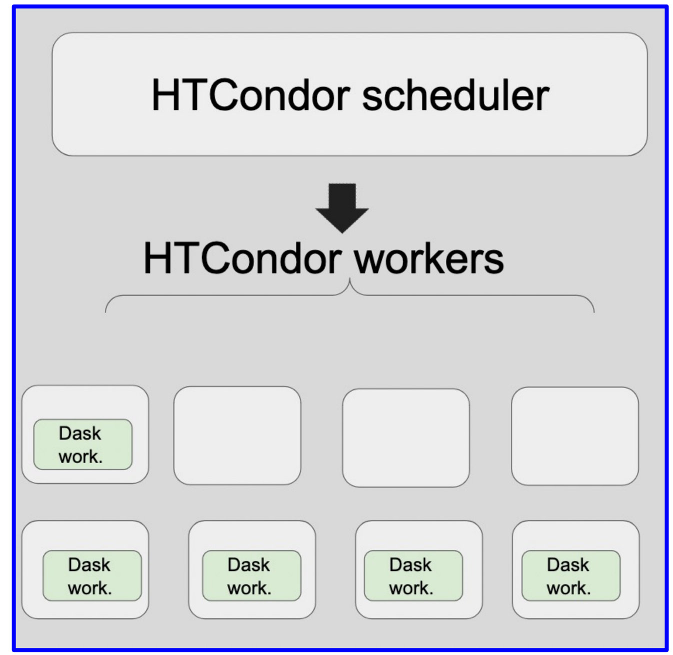
Welcome to the Analysis Facility at the University of Wisconsin - Madison!

The Analysis Facility gives users access to the UW Tier-2 computing cluster, and is configured around the analysis tools and concepts of coffea. So, it is best suited for users looking for access to coffea and associated tools and/or distributed computing resources.

Running analysis with Coffea + Dask (through SSH)

Interactive User
Login Node
(through SSH)

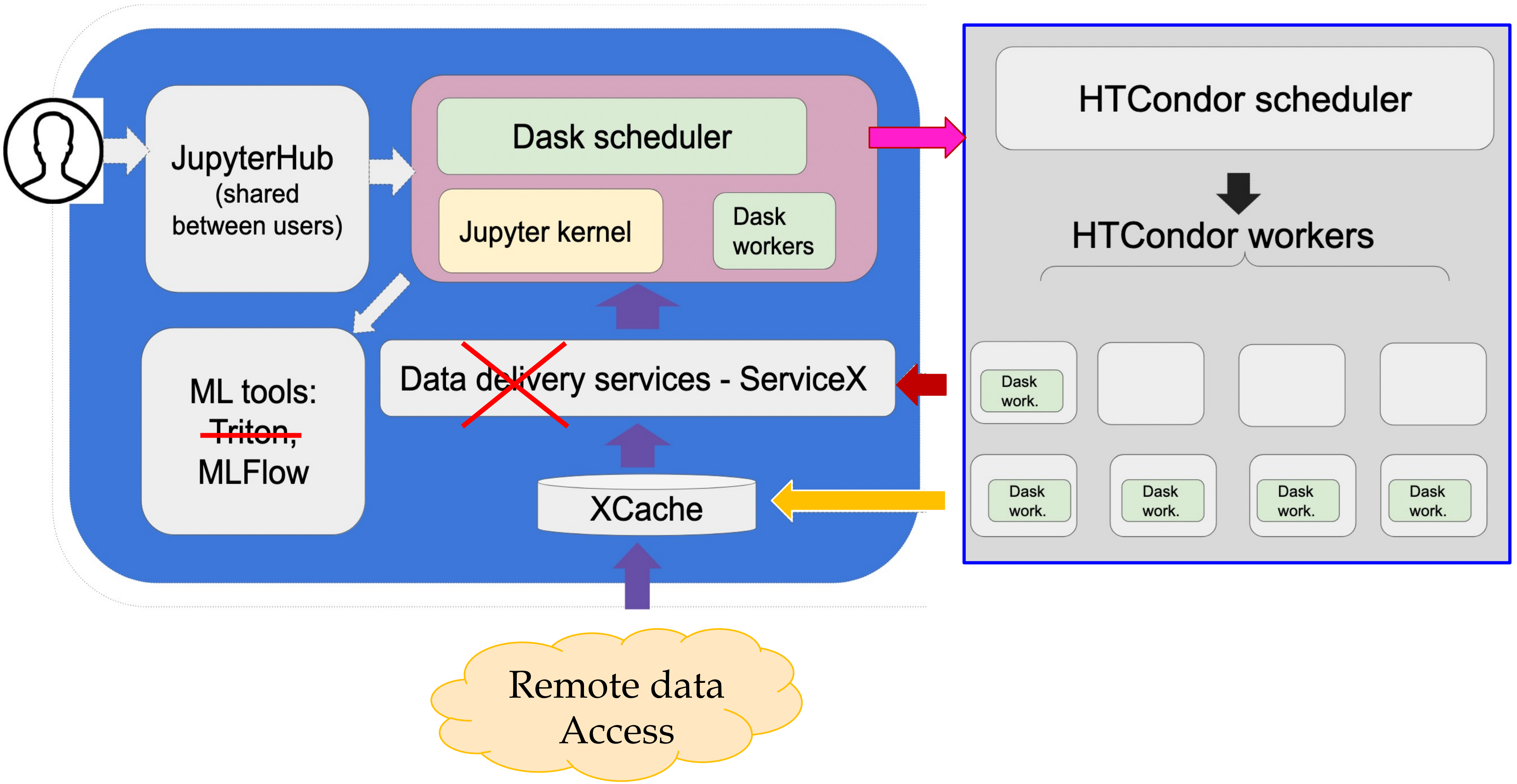
- Prepare Coffea analysis code
- Configure information about dataset, list of files, and # events to process, number of max jobs etc.
- Configure Dask Scheduler



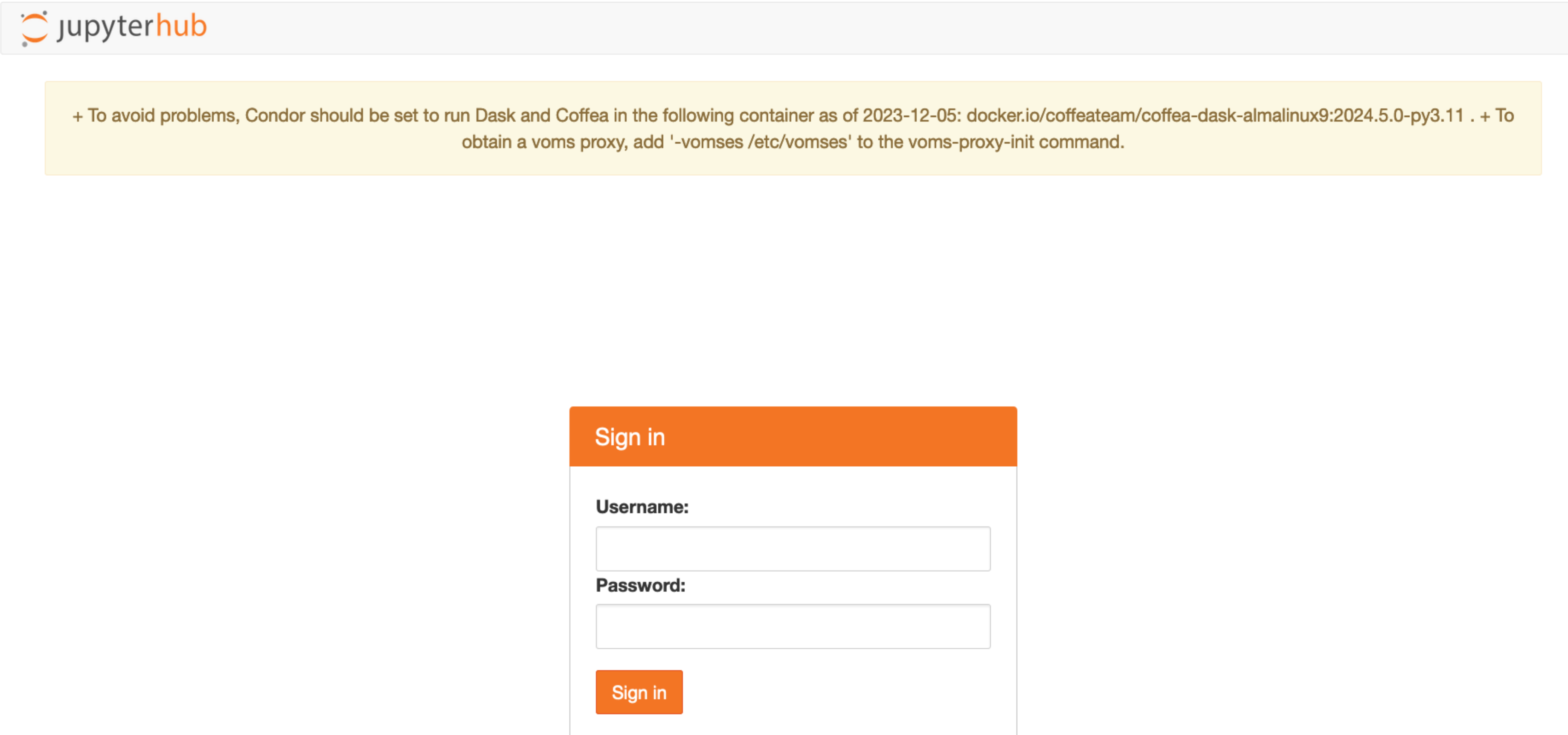
Remote data
Access

Analysis with Jupyter + Coffea + Dask

- Jupyterhub → User home directories (AFS), temp dirs (NFS), storage (HDFS), and authentication (Kerberos).

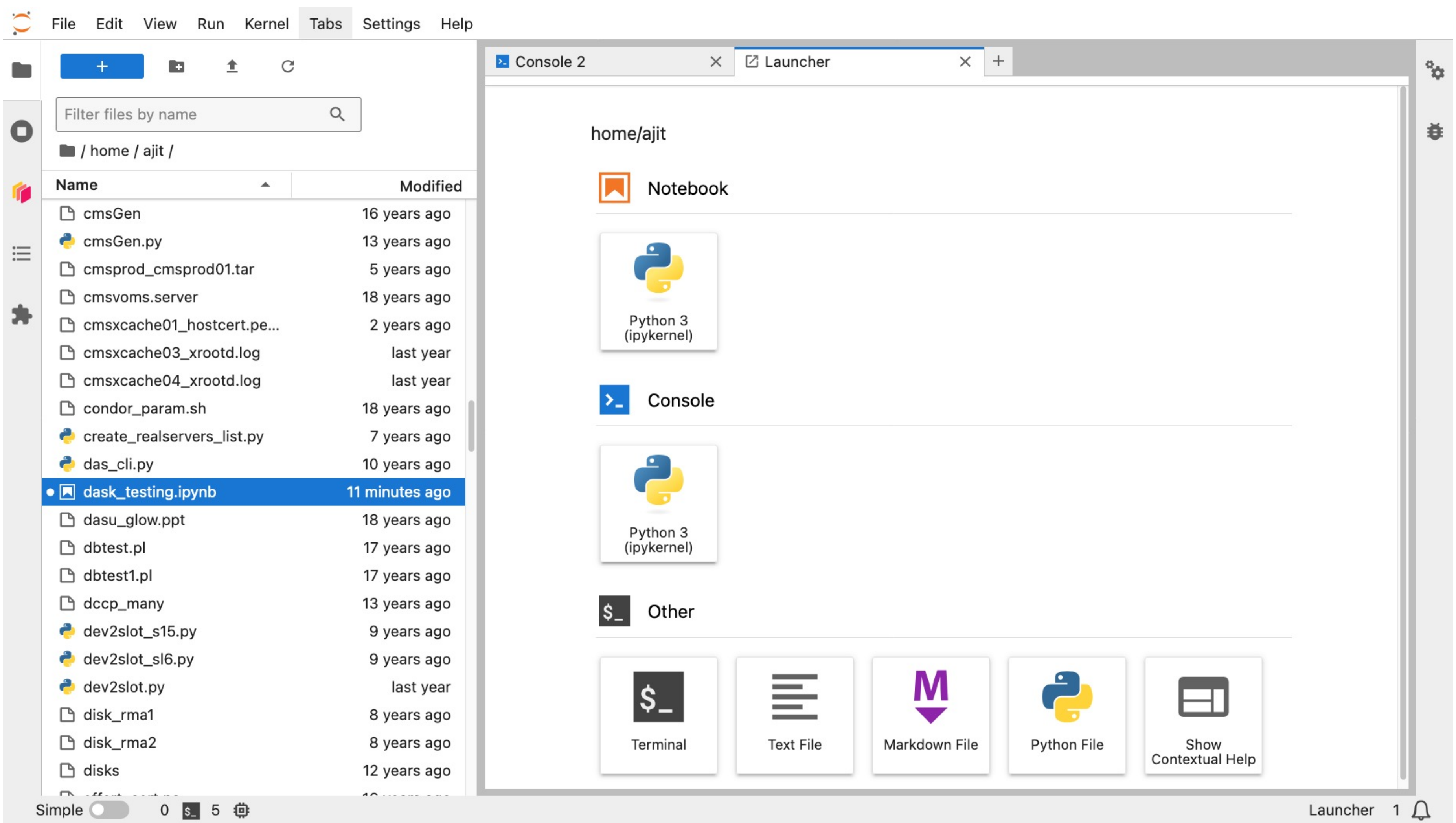


Analysis with Jupyter + Coffea + Dask



The jupyterhub runs on a 128-core node (serves as the local dask cluster)

Analysis using Jupyter + Coffea + Dask



Dask Supports HTCondor Cluster

```
[2]: import os
     from dask.distributed import Client, progress
```

```
[ ]: ## this starts local workers
     ## if this is desired, uncomment and execute instead of the condor workers section
     #client = Client(processes=False, threads_per_worker=4,
     #                n_workers=1, memory_limit='2GB')
     #client
```

```
[3]: # this cell starts workers in condor

     from dask_jobqueue import HTCondorCluster
     from dask import delayed
     from dask.distributed import Client, as_completed

     os.environ["CONDOR_CONFIG"] = "/etc/condor/condor_config"

     cluster = HTCondorCluster(
         cores=1,
         memory='5gb',  # last measured to use 3.5G
         disk='1 GB',
         job_extra_directives={
             "+SingularityImage": '"/cvmfs/unpacked.cern.ch/registry.hub.docker.com/coffeateam/coffea-dask:0.7.22-py3.10-g7cbcc"',
             "Requirements": "Machine == \"cms01.hep.wisc.edu\"",
             #"Requirements": "(HasSingularityJobStart)",
             "InitialDir": f'/var/tmp/{os.environ["USER"]}',
             "log": "dask_job_output.$(PROCESS).$(CLUSTER).log",
             "output": "dask_job_output.$(PROCESS).$(CLUSTER).out",
             "error": "dask_job_output.$(PROCESS).$(CLUSTER).err",
             "should_transfer_files": "yes",
             "when_to_transfer_output": "ON_EXIT_OR_EVICT",
         },
     #    job_script_prologue=[
     #        "export SINGULARITY_TMPDIR=./tmp"
     #    ]
     )
     cluster.adapt(minimum=0, maximum=4)


     client = Client(cluster)
     client
```

Dask Supports HTCondor Cluster

[3]:

Client

Client-ebafaafb-6652-11ef-8f65-3cecefda85c8

Connection method: Cluster object	**Cluster type:** dask_jobqueue.HTCondorCluster
Dashboard: proxy/1691/status	

Launch dashboard in JupyterLab

▾ Cluster Info

HTCondorCluster

11e87460

Dashboard: proxy/1691/status	**Workers:** 0
Total threads: 0	**Total memory:** 0 B

▾ Scheduler Info

Scheduler

Scheduler-1098e9f4-a6ff-42e8-a93f-e62f697d35d4

Comm: tcp://144.92.181.248:27499	**Workers:** 0
Dashboard: proxy/1691/status	**Total threads:** 0
Started: Just now	**Total memory:** 0 B

▸ Workers

Benchmarking with CMS Open Data Analysis

- Physics analysis → Top pair production using the following datasets:
  - ttbar → hadronic
  - single top
  - W+jets
- Dataset size → 1.5 TB
- # of files → 832 (distributed across 32 sites)
- # of events → ~800M
- Max # of concurrent running jobs → 180
- Analysis Pipeline → data delivery and processing, histogram construction and visualization, and statistical inference.

Thanks to our grad student "Ryan Simeon" for this work !

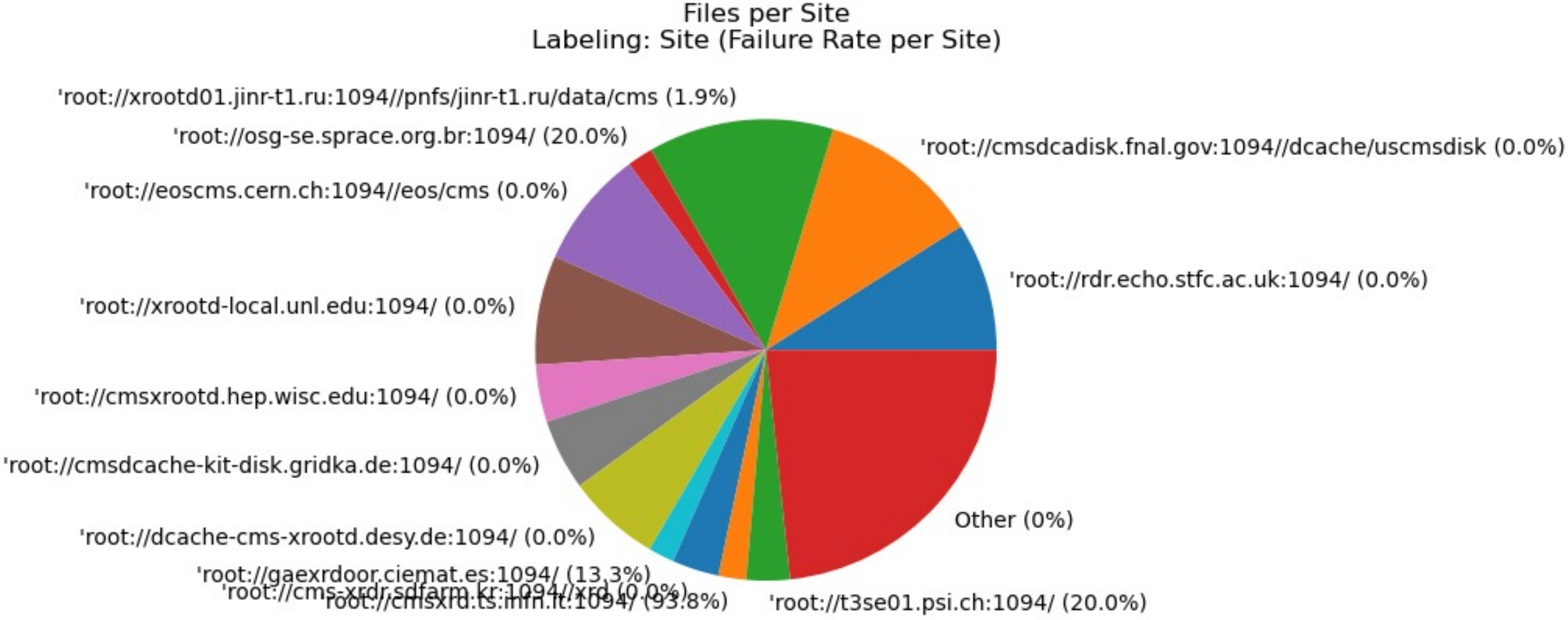
Benchmarking Observables

- How long did it take to run the analysis?
  - ~45 minutes
- Where did the files come from?
  - 32 sites (max)
- Input files read reliability/success rate ?
  - ~95%
- What kinds of error messages encounted ?
  - Various types (next slide)
- Run time with files read from Wisconsin Xcache → ~60 minutes. Need to understand the reason for the slowness and find remedy.

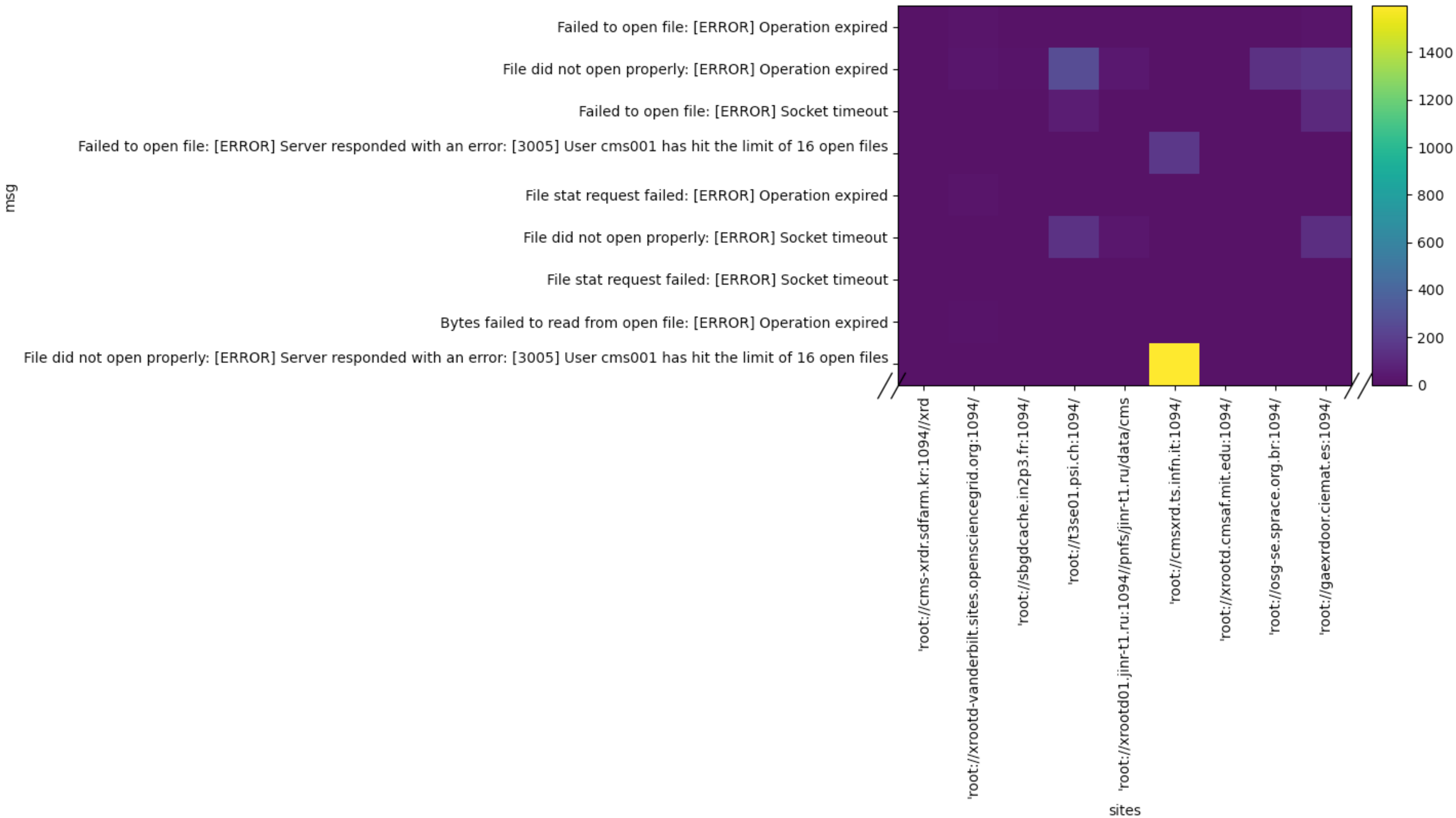
Remote file reading failure analysis

823 Total Files

- Pie wedges proportional to number of files from that site
- Number in parentheses is file read failure rate from that site
- Sites grouped in “other” host < 3% of total files and have zero failures



Error Messages linked to Sites



Summary

- ✓ An analysis facility setup at Wisconsin is in progress.
- ✓ Running coffea analysis through interactive SSH terminal and through jupyter notebook are supported.
- ✓ Users are free to choose the method that's suitable for them.
- ✓ Jupyter notebook way is gaining popularity among new users starting data analysis.
- ✓ Regular tutorials offered by CMS experts related to coffea framework for analysis has been highly beneficial for CMS students, postdocs, and others interested in these tools.
- ✓ Excellent documentation about analysis facility and efficient user support must be maintained.
- ✓ Integration of Kubernetes as the setup matures.

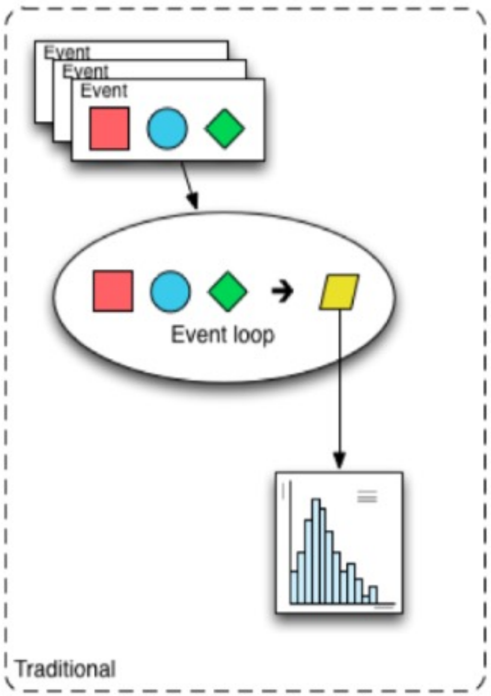
Thank You !

Questions / Comments ?

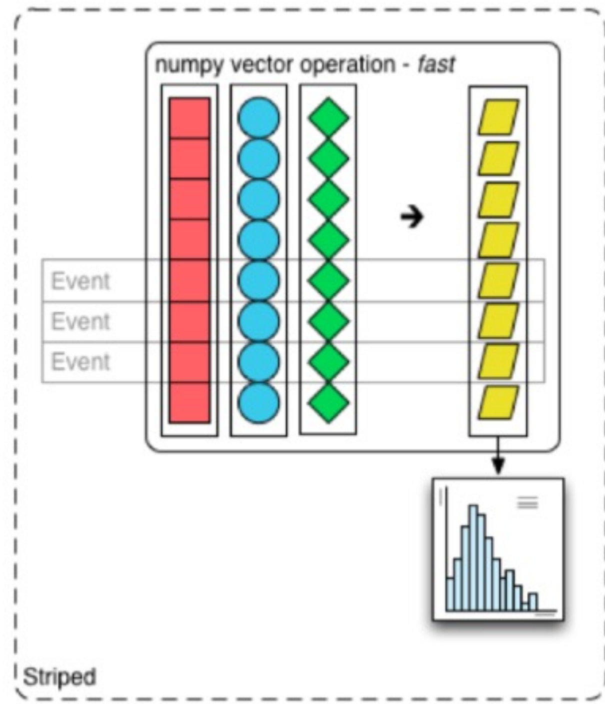
Event Loop → Columnar

- Event Loop: Inefficient, Slow and Expensive

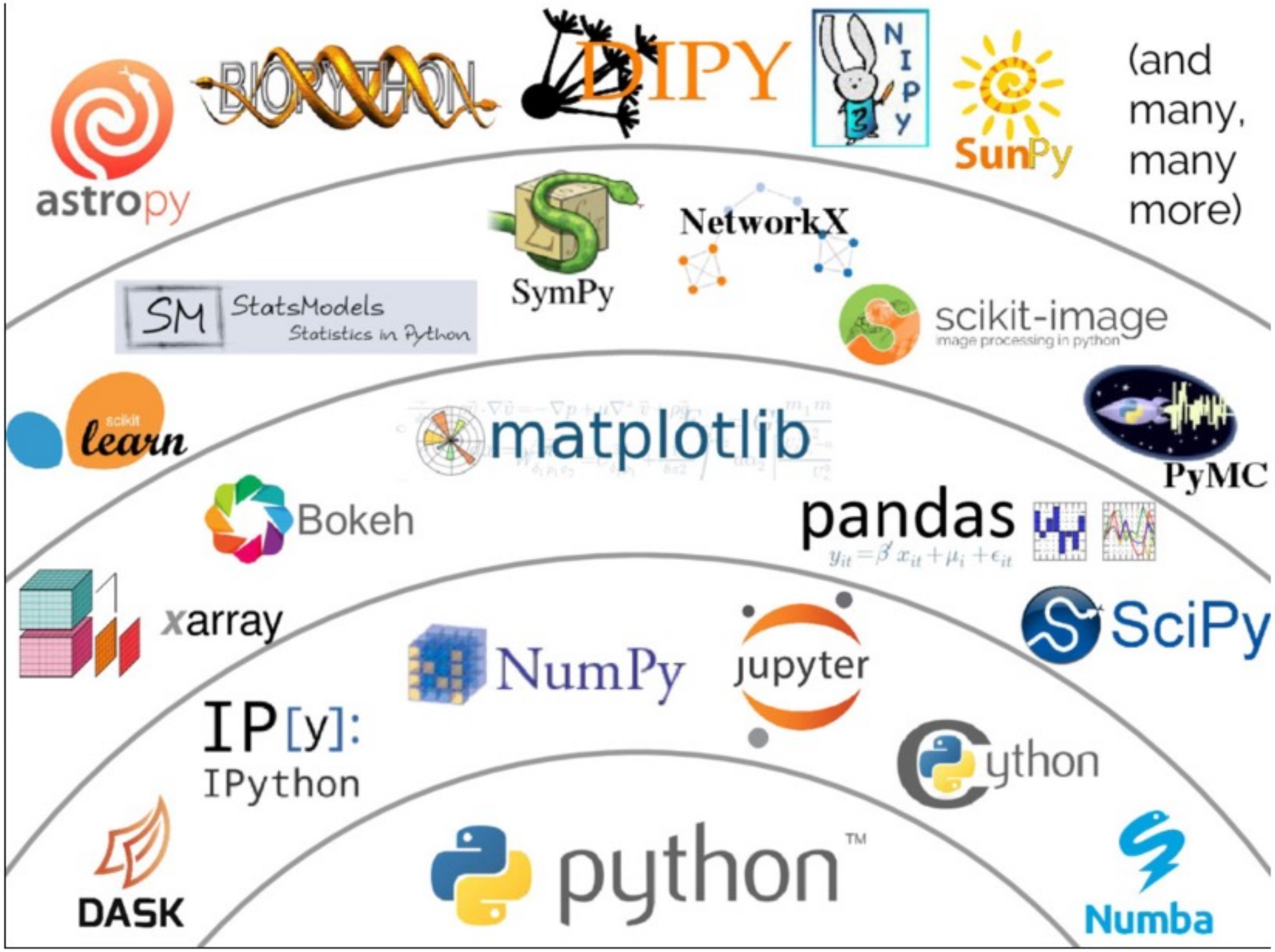
- Columnar processing: Orders of magnitude faster



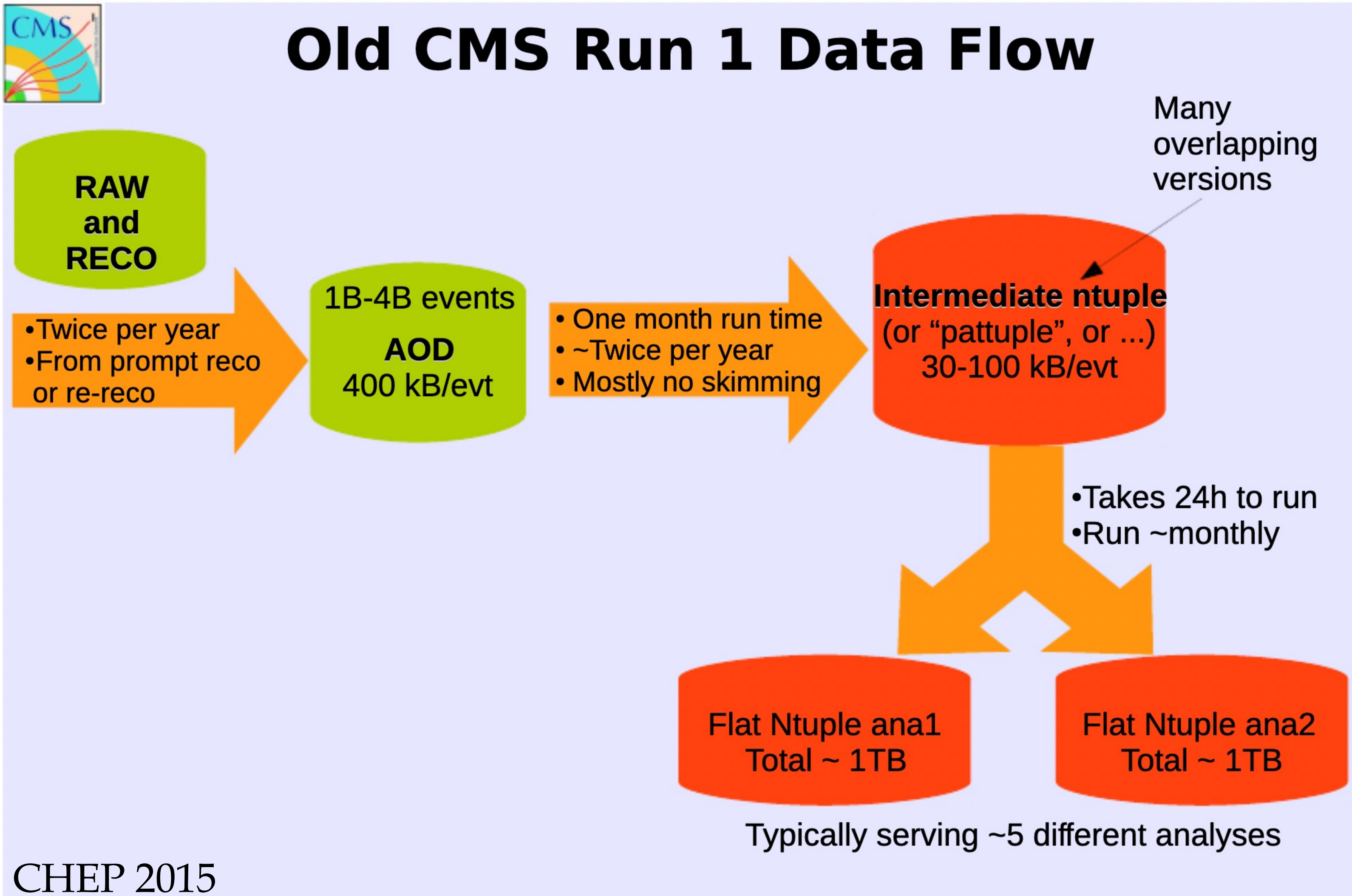
Switch



Scientific Python Ecosystem



CMS Run 1 Data Flow



CMS Run 2 Data Flow