Analysis Facilities for the HL-LHC

US CMS

MORGRIDGE
INSTITUTE FOR RESEARCH
CORE COMPUTATION

iris hep

FEARLESS SCIENCE

A bit about me

I have been working on CMS computing since 2006 and the OSG since 2008.

- Helped introduce data federations based on XRootD to the community as part of the AAA project.
- Accordingly, this presentation is going to be strongly colored by my experience (e.g., LHC-centric, CMS-centric).

Currently at the Morgridge Institute for Research where I work closely with Miron Livny and the Center for High Throughput Computing.

- PhD is in Math and CS – will not be in danger of covering any physics topics in this talk!

What is an Analysis Facility, anyway?

When I say 'analysis facility,' what do I mean?

- People, software / services, and hardware meant to support analysis activities for an experiment aggregated into a coherent whole.
- **Services** includes:
  - Access to experimental data products.
  - Storage space for per-group or per-user data (often ntuples).
  - Access to significant computing resources.
- I promised to not talk about physics **software**, but obviously this is ROOT and the growing Python-based ecosystem.
- Computing **hardware** (currently) looks like most of our computing facilities: CPUs and disks. More emphasis on high-data-rates. However, there's a growing need for GPUs.

Great Example – USCMS LPC

What exists today at the USCMS LPC?

- Dedicated staff to help.
- Users can login to hosts via SSH and do modest technical work.
  - Any large-scale computation is done via HTCondor.
- Filesystems for input and output – currently using EOS.
  - Uses the same authentication mechanism as CMS – X.509 (GSI/VOMS).

LPC Services

Compute Interface	SSH host & Batch System
Batch System	HTCondor
Data Interface	POSIX file-like
Input products	EOS disk / offsite XRootD
Data Storage	EOS disk

Why are we interested in Analysis Facilities?

The LHC has been running for years - why are we interested in an AF now?

- HL-LHC will have at least one order-magnitude increase in event count, if not two.
  - Premise: Work previously done on the laptop will now require significant computing resources.
- We want analysts to use significant resources interactively, similarly to how they use their laptops today.
  - Batch jobs – even when they startup quickly – have a distinct experience from interactive use.
- We want new services to be available to users as well as new resource types.

Columnar Analysis

We also want to explore what services are needed in order to support columnar analysis.

- Operations should be expressed on arrays of data, not individual events.
- Represent HEP data as arrays in memory, allowing us to leverage existing libraries that work on vector data (think: numpy).
- Track non-uniformity ("jaggedness") of arrays in a separate array.
- No C++-style objects in memory!

Run Physics Analysis on a GPU

Notebook from Joosep Pata's tutorial at PyHEP 2020

Awkward Array

See also: https://coffeateam.github.io/coffea/

Our Work

The Nebraska CMS-AF

We are working to provide a new facility at the CMS Tier-2 site at Nebraska. Goal:

- Any CMS user can start a JupyterLab interface and get computational resources through the familiar "Dask" interface.
- Have all auth{n,z} be based on web single-sign-on – not X.509 client certificates.
- As the community grows, add new services to the stack.

CMS-AF Services

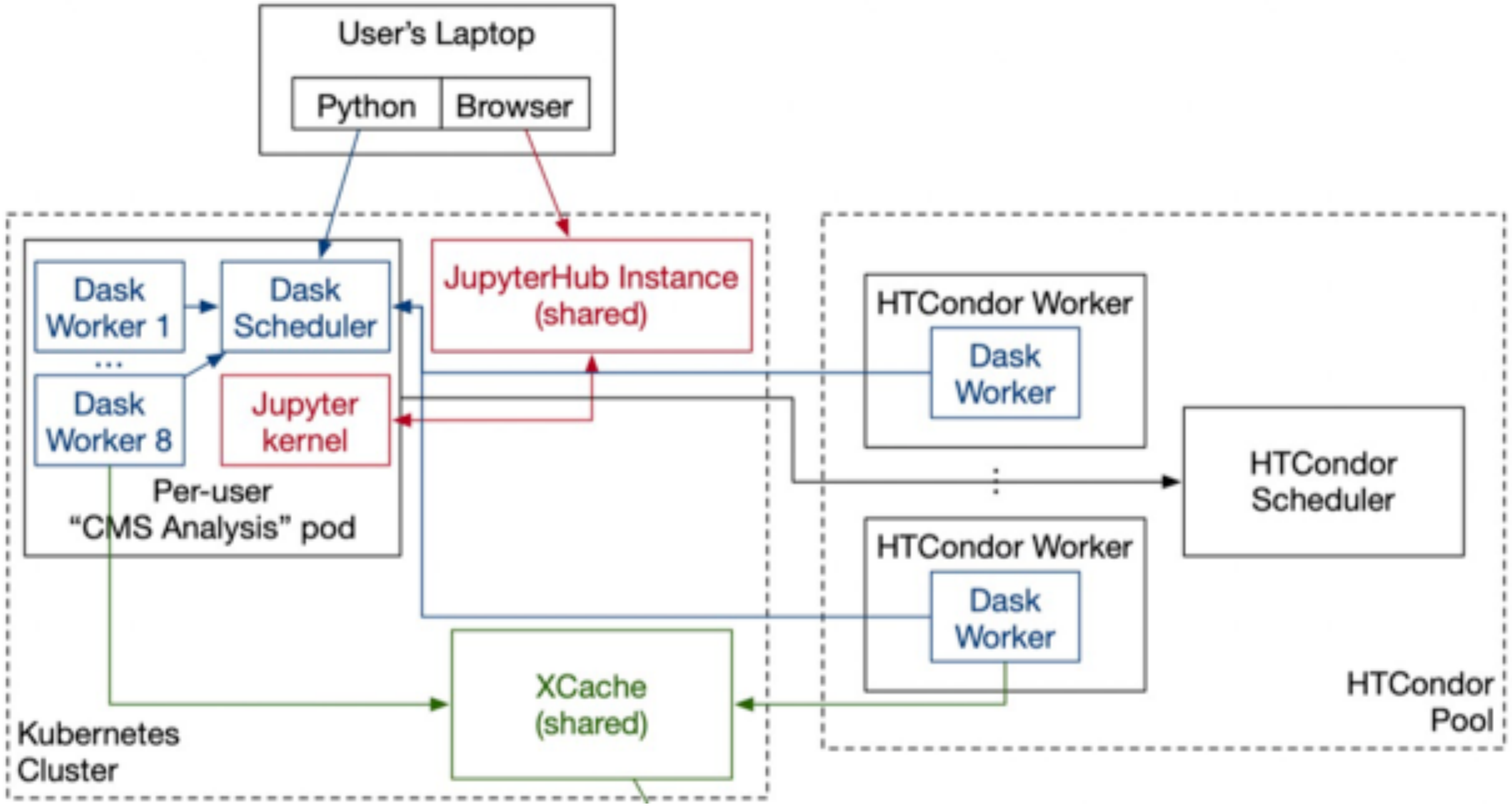
Compute Interface	Jupyter Notebooks & Dask Scheduler
Task Scheduler	Dask
Data Interface	Object-store like
Input products	CMS AAA through XCache
Data Storage	??? Unresolved.

YouTube demo.

First service – the JupyterLab & Dask Scheduler

Our first service – functioning now, but aiming to scale to all of USCMS – is a Dask scheduler.

- Completely focused on interactive use cases. Dask workers are launched as soon as the user's notebook is available.
  - **"A chicken in every pot and 8 cores for every analyst"**.
- Beyond the resources started in Kubernetes, we launch additional workers in HTCondor.
  - Resources beyond the 'standard allocation' are allocated by HTCondor via the standard fairshare mechanism.
- We feel it is essential that these analysis resources coexist with the existing resources – we can't afford partitioning a standalone facility.

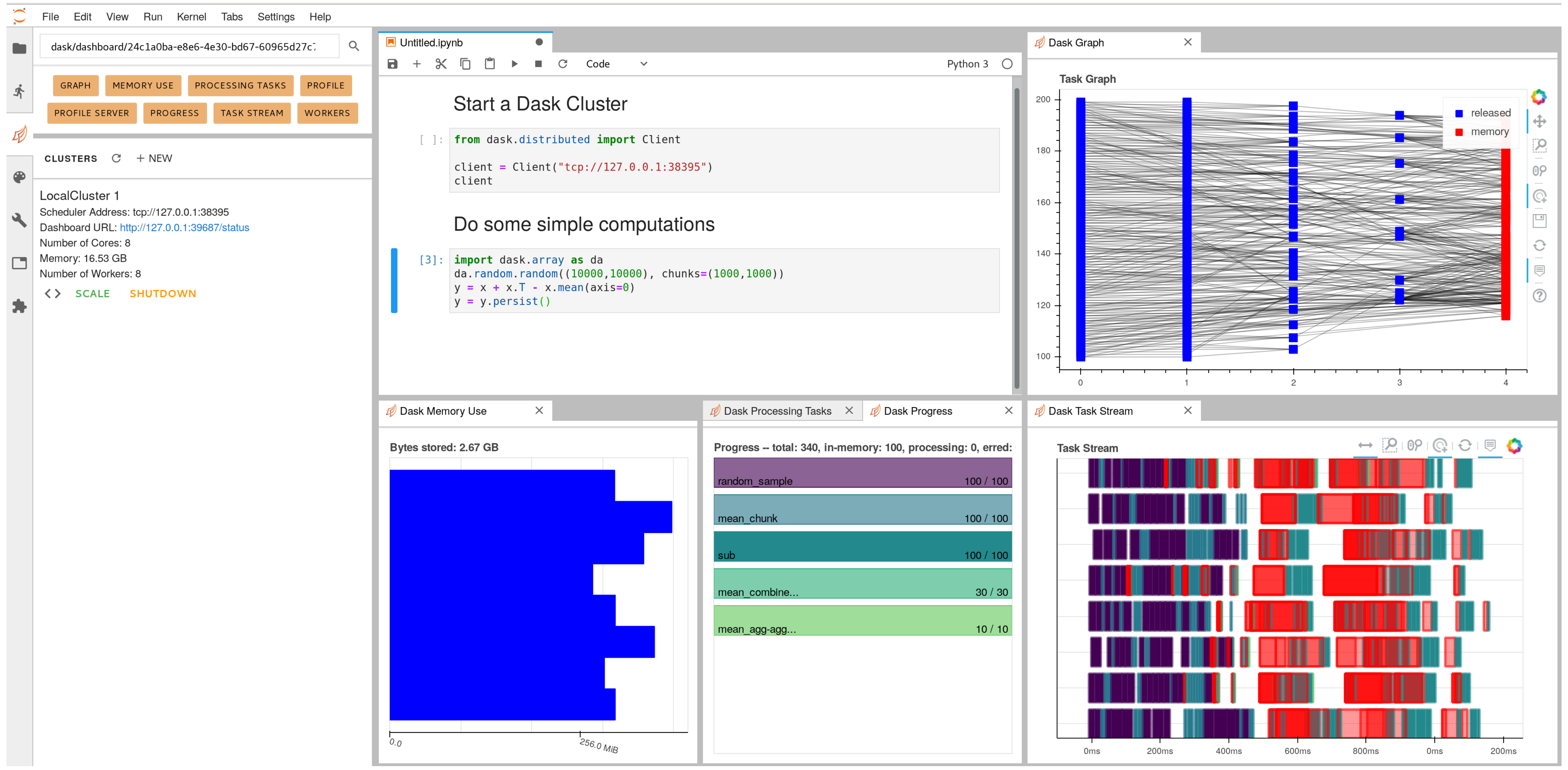


An Aside on Dask

Dask provides a simple task-based parallelism interface in Python.

- Tasks are scheduled out in milliseconds to dedicated worker processes.
- Dask workers keep intermediate results in memory, providing for data locality and minimizing data movement.

Dask provides <u>interactive use</u>, which is perhaps the biggest difference compared to today's batch-system-based solution.



LocalCluster 1
Scheduler Address: tcp://127.0.0.1:38395
Dashboard URL: http://127.0.0.1:39687/status
Number of Cores: 8
Memory: 16.53 GB
Number of Workers: 8

Start a Dask Cluster

```
[ ]: from dask.distributed import Client

     client = Client("tcp://127.0.0.1:38395")
     client
```

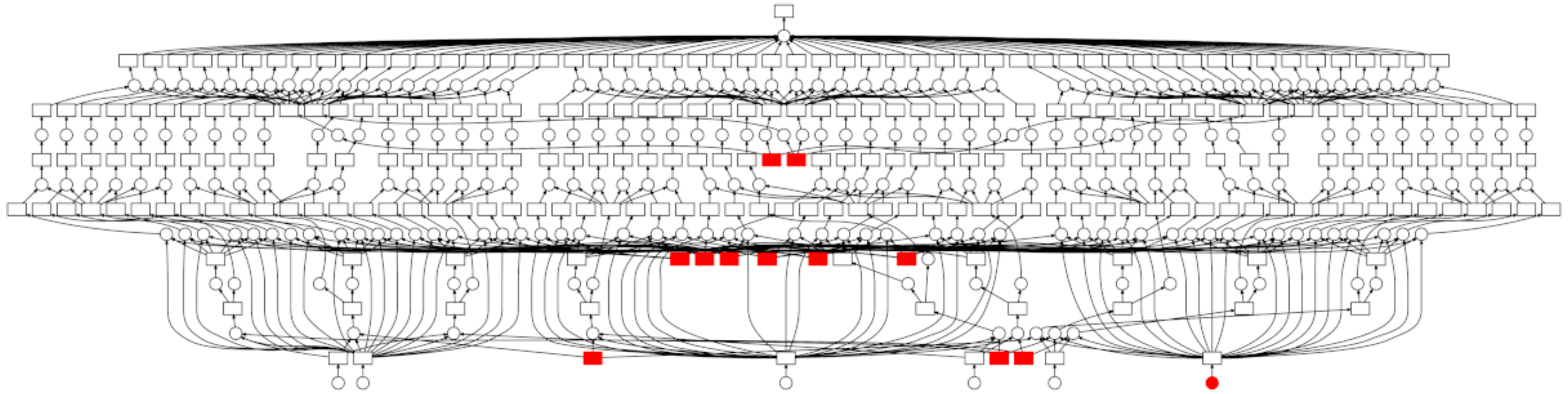
Do some simple computations

```
[3]: import dask.array as da
     da.random.random((10000,10000), chunks=(1000,1000))
     y = x + x.T - x.mean(axis=0)
     y = y.persist()
```

Bytes stored: 2.67 GB

Progress -- total: 340, in-memory: 100, processing: 0, erred:

```
random_sample      100 / 100
mean_chunk         100 / 100
sub                100 / 100
mean_combine...      30 / 30
mean_agg-agg...      10 / 10
```



Beyond Dask

What are the minimal services beyond Dask?

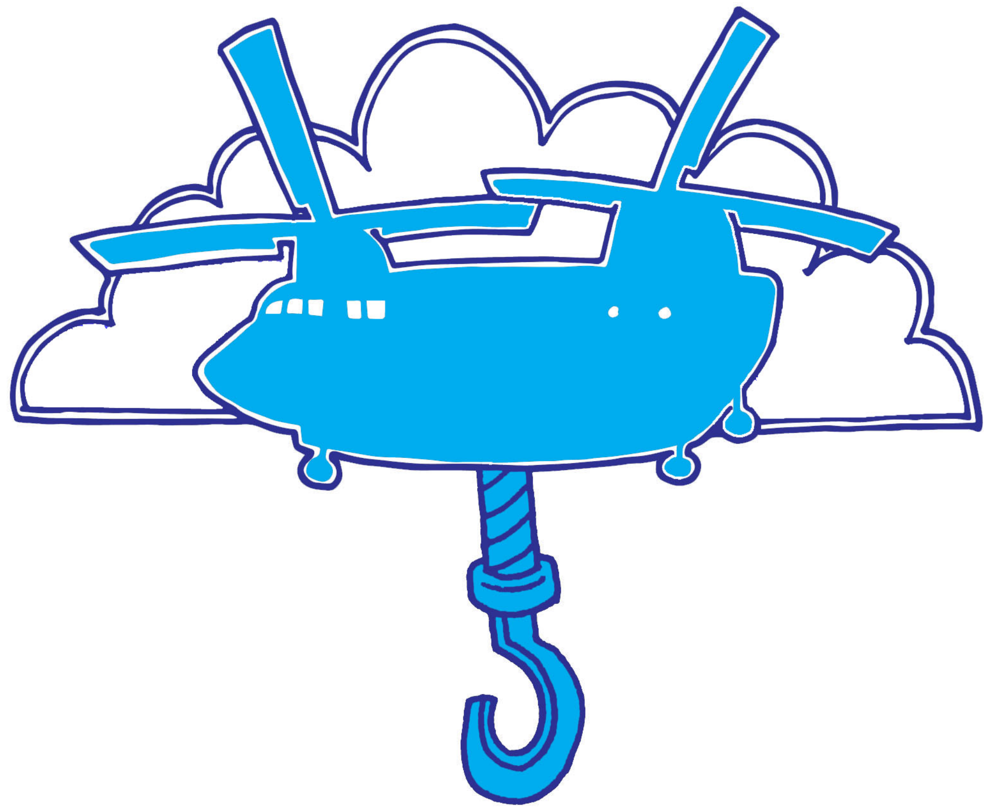
- Authentication and authorization based on OAuth2 – ensure that only CMS users can access the AF.
- Data is delivered through XCache: when tasks are started, they are started with credentials that can read from the onsite XCache, which proxy out to the wider CMS data federation.
- Local storage for code / notebooks.

Big open question: what should be done about user data?

Next up: data handling

We think of the Dask scheduler as the initial service for the CMS-AF. What's next?

- **Derivation of columns & ntuples**: Most analyses can't use experiment-produced ntuples directly but need to generate them from an analysis format. We are looking at ServiceX, developed by IRIS-HEP. ServiceX derives columns on demand from external datasets and can deliver them to object storage.
- **Actual column store**: Right now, all events are still kept in a file structure. We are looking at using SkyHook DM to store ntuples in a dataset as a "table-like" structure -- but add database-like primitives (SELECT/PROJECT/FILTER).
  - Particularly, being able to augment existing tables (datasets) with additional columns is seen as a killer feature.



Take-Home

What’s the message for the larger Snowmass community?

- Analysis facilities have been with us for awhile – they have been important and will be important going forward.
  - Current facilities are largely based on the “batch system + filesystem” paradigm.
- We don’t have to stand still! There are a variety of other approaches (interactive notebooks, task-based, column stores) to explore.
  - Nebraska is working toward a specific vision, developing initially a Dask service.
  - There are several other parallel efforts I’m aware of (CERN’s SWAN, Vanderbilt, FNAL Elastic Compute Facility).

So, how should we all proceed?

MORGRIDGE
INSTITUTE FOR RESEARCH
CORE COMPUTATION

morgridge.org

FEARLESS SCIENCE