# USCMS Analysis Facilities

Lindsey Gray

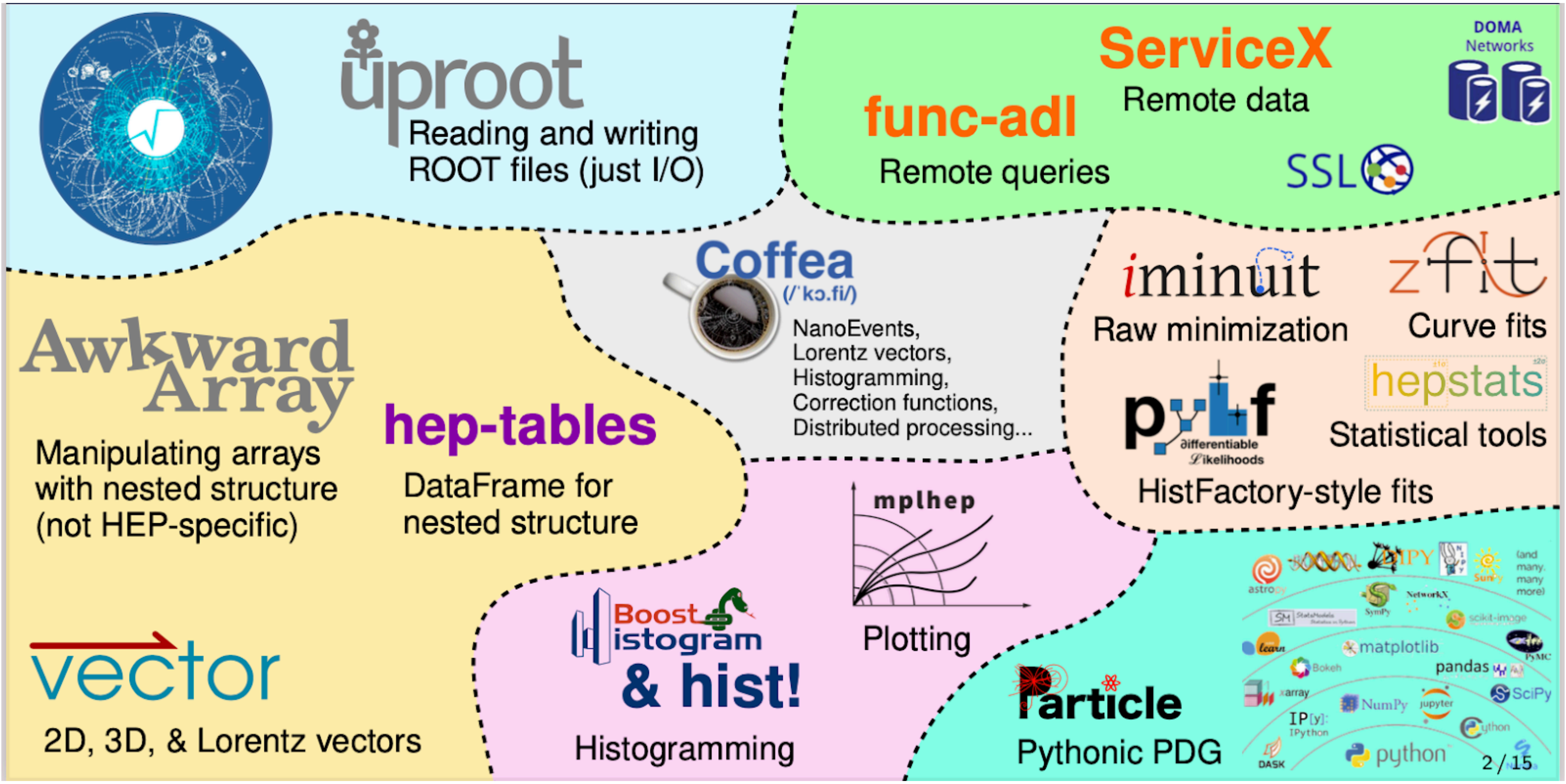
25 March 2022

HSF AF Forum Kick-off Meeting

## Outline

- Modern analysis tools - focusing on columnar tools
  - Analysis facilities then and now
  - Separating resource and compute scheduling
- Ongoing efforts within the US

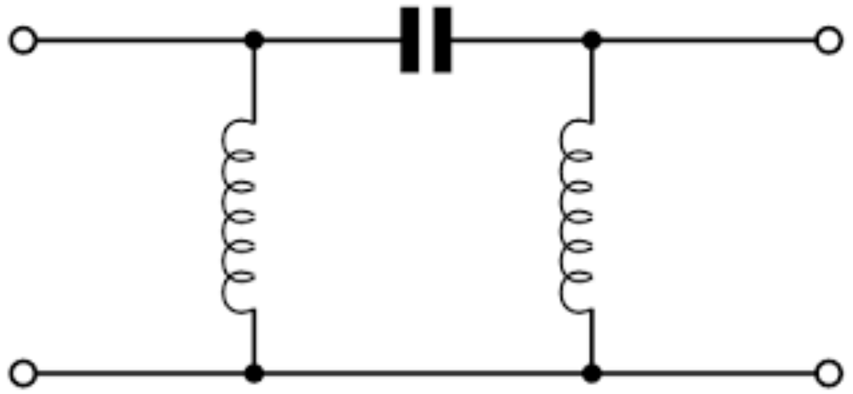
# New Facilities for New Tools



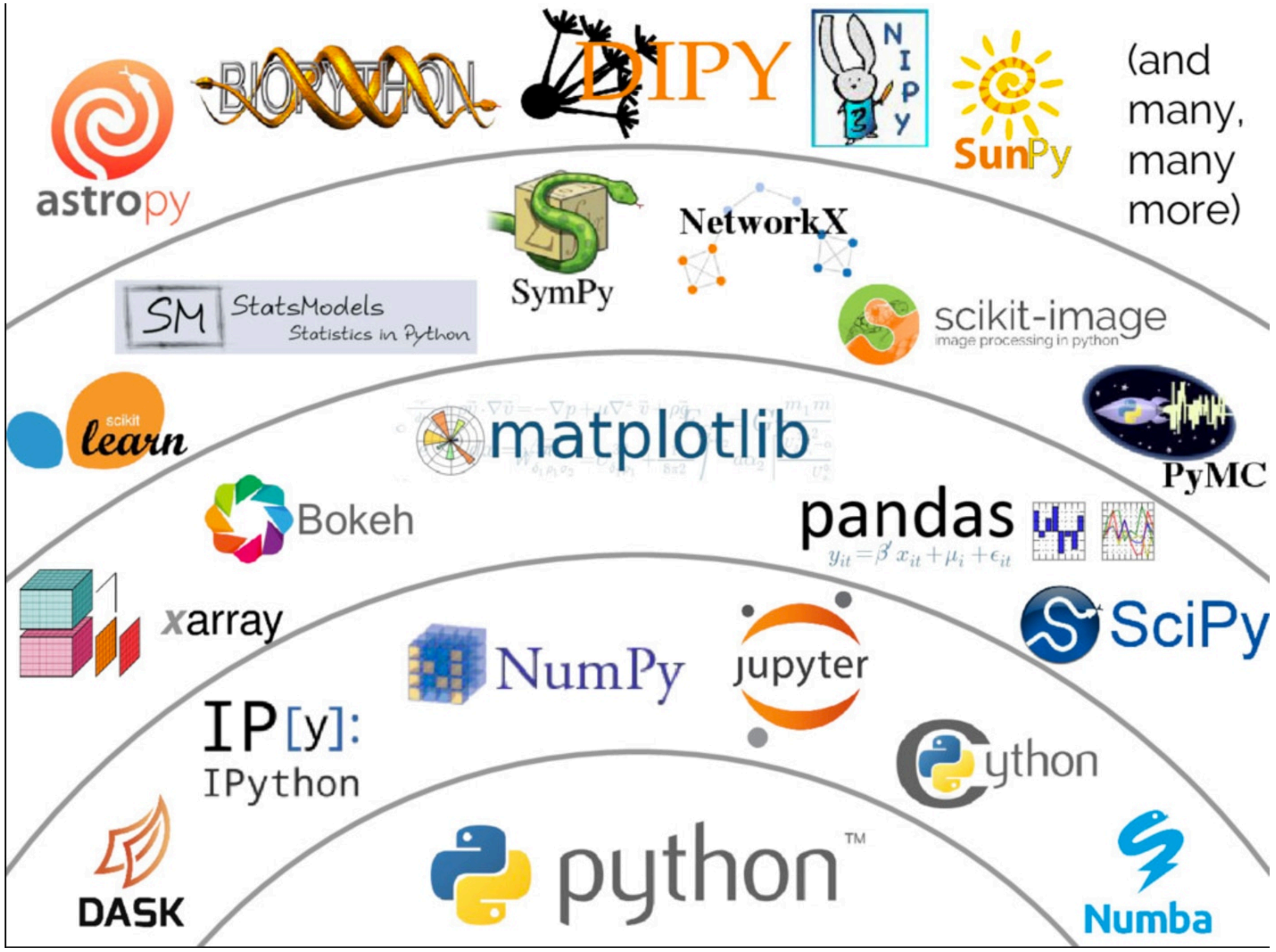
- ◉Generally: AFs provide a curated substrate upon which to easily deploy these (stacks of) tools at scale
  - The exact way in which this service is provided is currently filled with opinion, but there is a large degree of convergent evolution
  - We'll enumerate all the efforts and the directions under study at present
  - Each effort, while similar, does have different foci
- ◉While not in the box above - RDataFrame is within the scope of all AFs discussed in this talk

# Impedance Mismatches

- ROOT File <-> Machine Learning (uproot is everywhere nowadays)
- Big data <-> PyROOT (python for-loops are slow)
- HEP Physicist <-> Industry (we are a subset of wider data science)



# Scientific Python

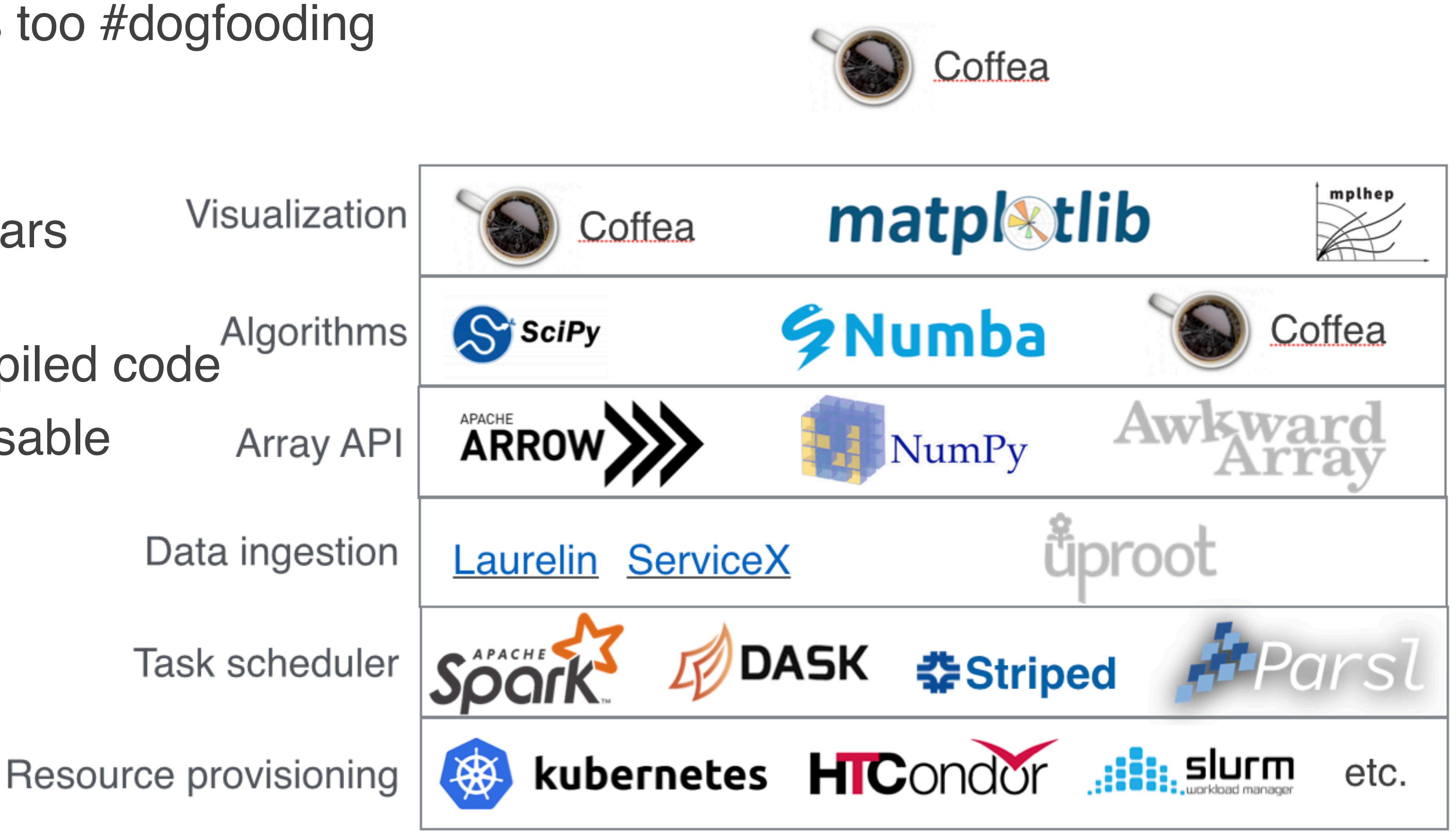


# What's Coffea?

- A package in the scientific python ecosystem
  - `$ pip install coffea`
- A user interface for columnar analysis
  - With missing pieces of the stack filled in
- A minimum viable product
  - We are data analyzers too #dogfooding
- A really strong glue

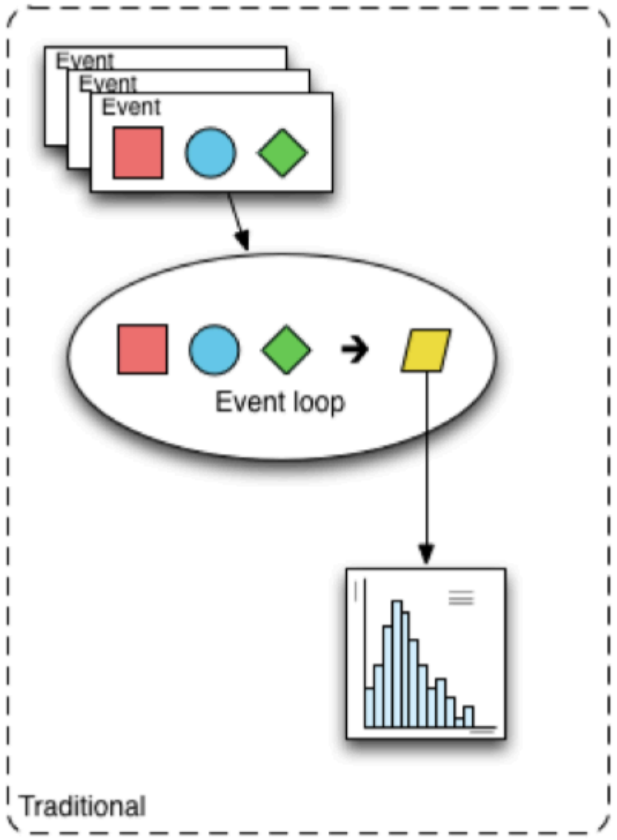
- Going strong for four years

- Roughly as fast as compiled code
  - Significantly more reusable

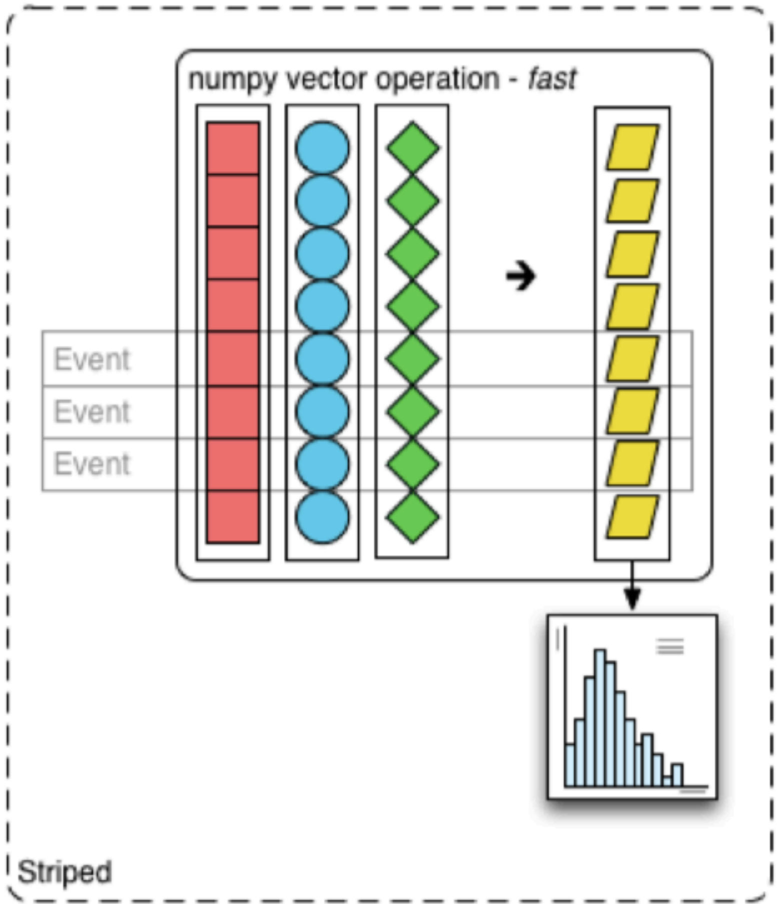


# What is columnar analysis?

- Event loop analysis:
  - Load relevant values for a specific event into local variables
  - Evaluate several expressions
  - Store derived values
  - Repeat (explicit outer loop)



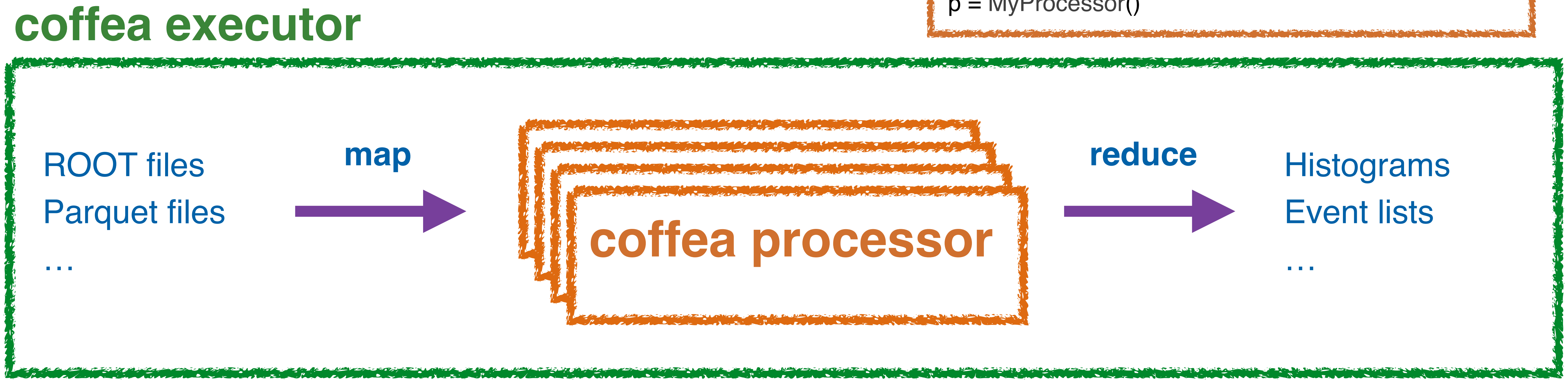
- Columnar analysis:
  - Load relevant values for many events into contiguous arrays
  - Evaluate several **array programming** expressions
    - Implicit *inner* loops
  - Store derived values



## Scaling Out

- User is provided data frame of columns they wish to process
- User fills a defined set of accumulators
  - Histograms, dictionaries of counts, appendable arrays, …
- Coffea executor takes care of the rest
  - Local machine, dask, spark, parsl (and condor)

```
from coffea import hist, processor

class MyProcessor(processor.ProcessorABC):
    def __init__(self, flag=False):
        self._flag = flag
        self._accumulator = processor.dict_accumulator({
            # Define histograms
        })

    @property
    def accumulator(self):
        return self._accumulator

    def process(self, df):
        output = self.accumulator.identity()

        # PHYSICS GOES HERE

        return output

    def postprocess(self, accumulator):
        return accumulator

p = MyProcessor()
```



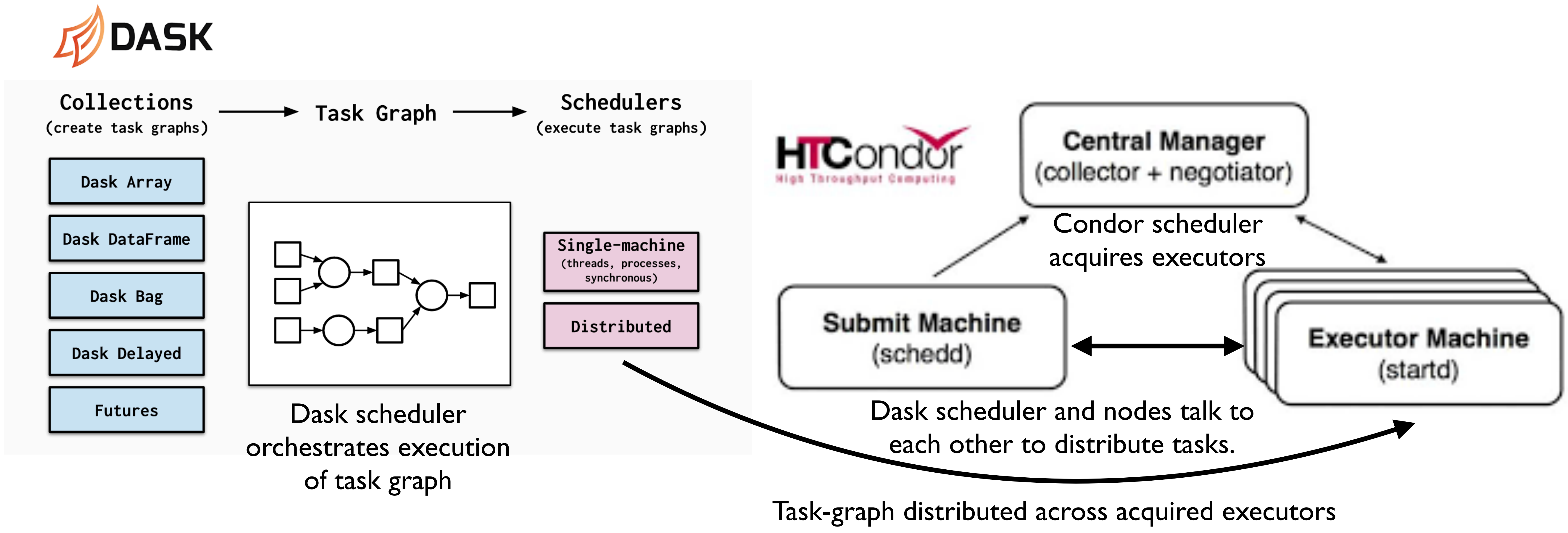
## Analysis Facilities Then and Now

- Physicists can handle an enormous amount of workflow complexity to achieve their goals
  - What's "easy" is incredibly subjective
    - person to person and analysis to analysis
- CMS Analysis facilities circa 2005-2019 have largely been login terminals with batch access
  - Just having that was sufficient to be a facility
  - CMS has published > 1000 papers working this way

## Analysis Facilities Then and Now

- However, it can be easier and less hectic for data analysts
  - Technologies like Dask, Apache Spark, Parsl, and Work Queue encapsulate and abstract physics analysis workflows
  - With this abstraction, administrators are able to determine more optimal resource deployment/usage patterns with a "weaker" binding directly to user code
  - Physicists can focus on physics while also efficiently using clusters
- I co-lead the "Analysis Tools Task Force" on CMS which will make recommendations on the usage of these technologies (n.b. not the facilities themselves) for Run 3 & beyond

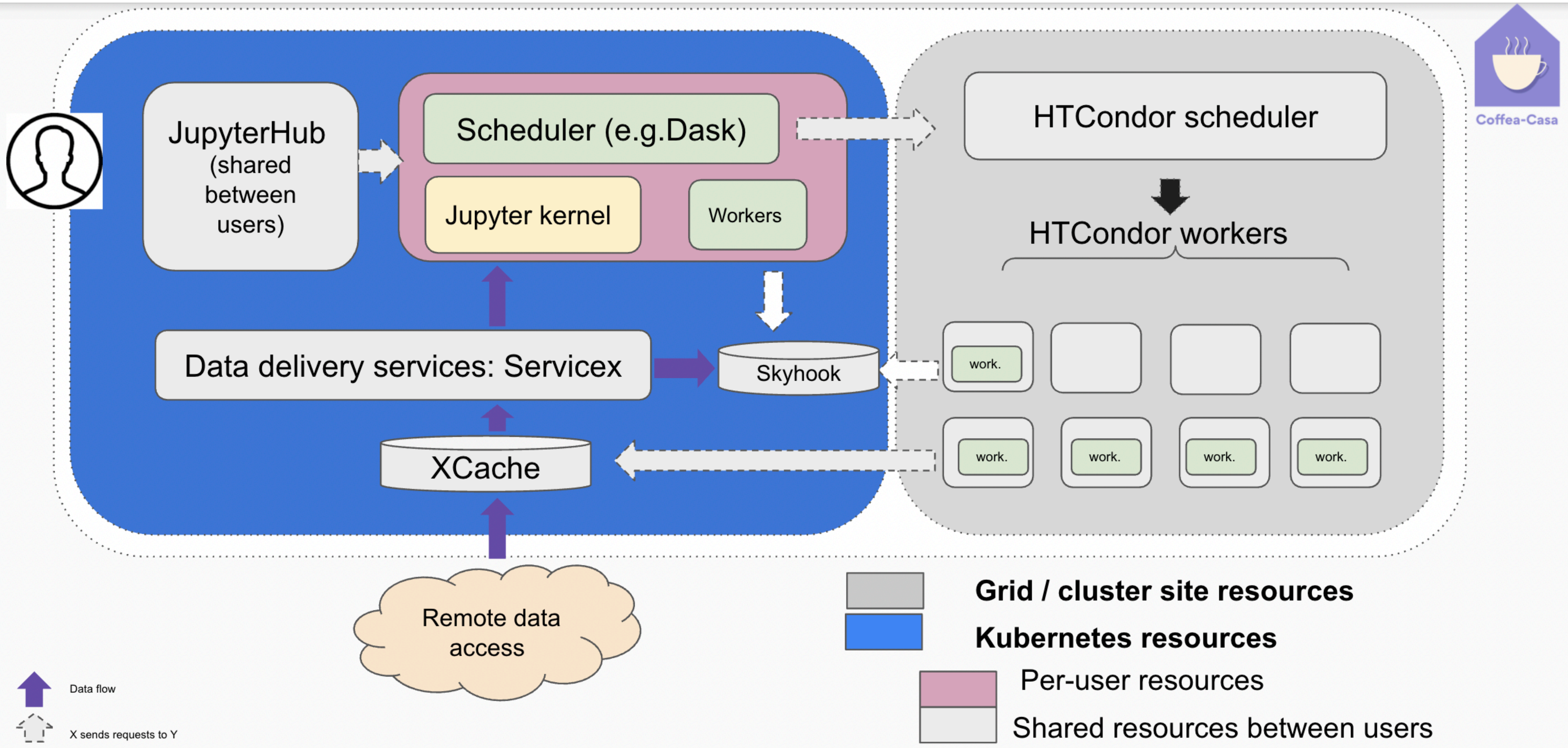
# Separating Resources From Computation



DASK
Collections
(create task graphs)
Task Graph
Schedulers
(execute task graphs)
Dask Array
Dask DataFrame
Dask Bag
Dask Delayed
Futures
Single-machine
(threads, processes, synchronous)
Distributed
Dask scheduler orchestrates execution of task graph
HTCondor
High Throughput Computing
Central Manager
(collector + negotiator)
Condor scheduler acquires executors
Submit Machine
(schedd)
Executor Machine
(startd)
Dask scheduler and nodes talk to each other to distribute tasks.
Task-graph distributed across acquired executors

- The main benefit of tools like Dask is to factorize the accumulation of resources from the execution of a computing workload
  - If your tasks are short (t_task << t_analysis) this factorization removes a significant amount of wall-clock overhead simply from scheduling jobs and starting a worker
  - Additionally, since you know you have a given amount of compute for the entire duration of the analysis you can consider using distributed memory to store results

# Coffea Casa @ UNL



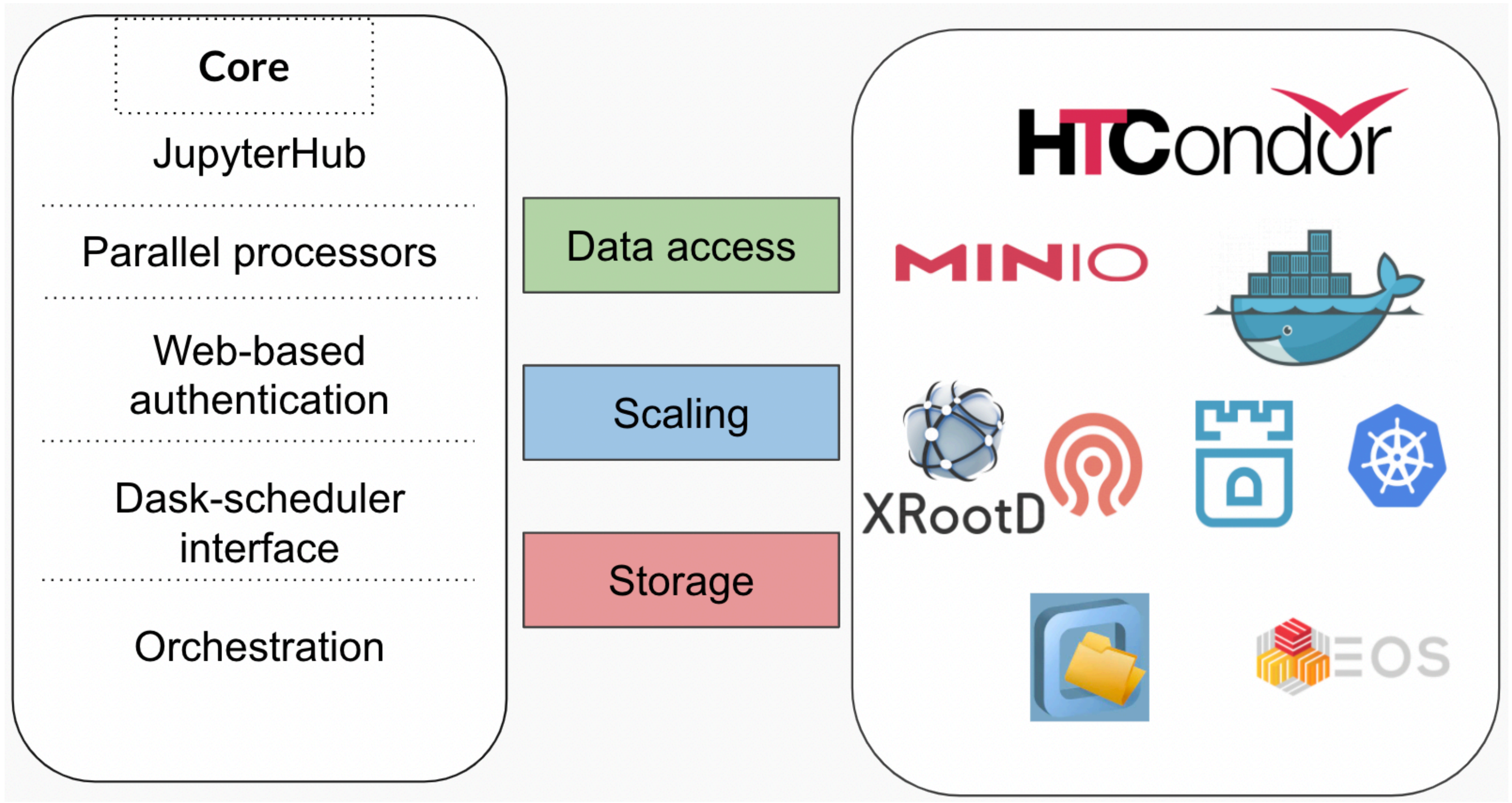
- JupyterHub + kubernetes deployment with spillover to HTCondor for large workloads
  - Supports 3-4 active analyses, have successfully burst to 20 concurrent users
  - Major interest in developing a sharable infrastructure as software, exploiting cloud-native deployment patterns

# Coffea Casa @ UNL

**CMSAF @T2 Nebraska**
**"Coffea-casa"**
**https://coffea.casa**

**OpenData AF @T2 Nebraska**
**"Coffea-casa"**
**https://coffea-opendata.casa**

**ATLAS AF @Scalable System Lab (UChicago)**
**"Coffea-casa"**



## AF @ MIT - Infrastructure

- Computing for login
  - Order of ten beefy machines including, large memory, O(500) CPU cores and O(10) big GPUs (NVidia T100/T4)
- Network
  - 100 Gb/s for all machines, RDMA enabled
- Storage
  - Tiered Storage:
    - Tape storage from MIT Tape Pilot project (being commissioned)
    - Spinning disks: T2 (10 PB) at 100 Gb/s, T3 (300 TB) at 2x10 Gb/s
    - NVMe sticks: Local (50 TB) at 2x100 Gb (waiting for delivery)
  - XCache is planned
- Behind the scenes
  - HTCondor: Tier-2, Tier-3, global pool, OSG
  - Slurm: local HPC resources (old lattice QCD cluster)

## AF @ MIT - Initial Setup

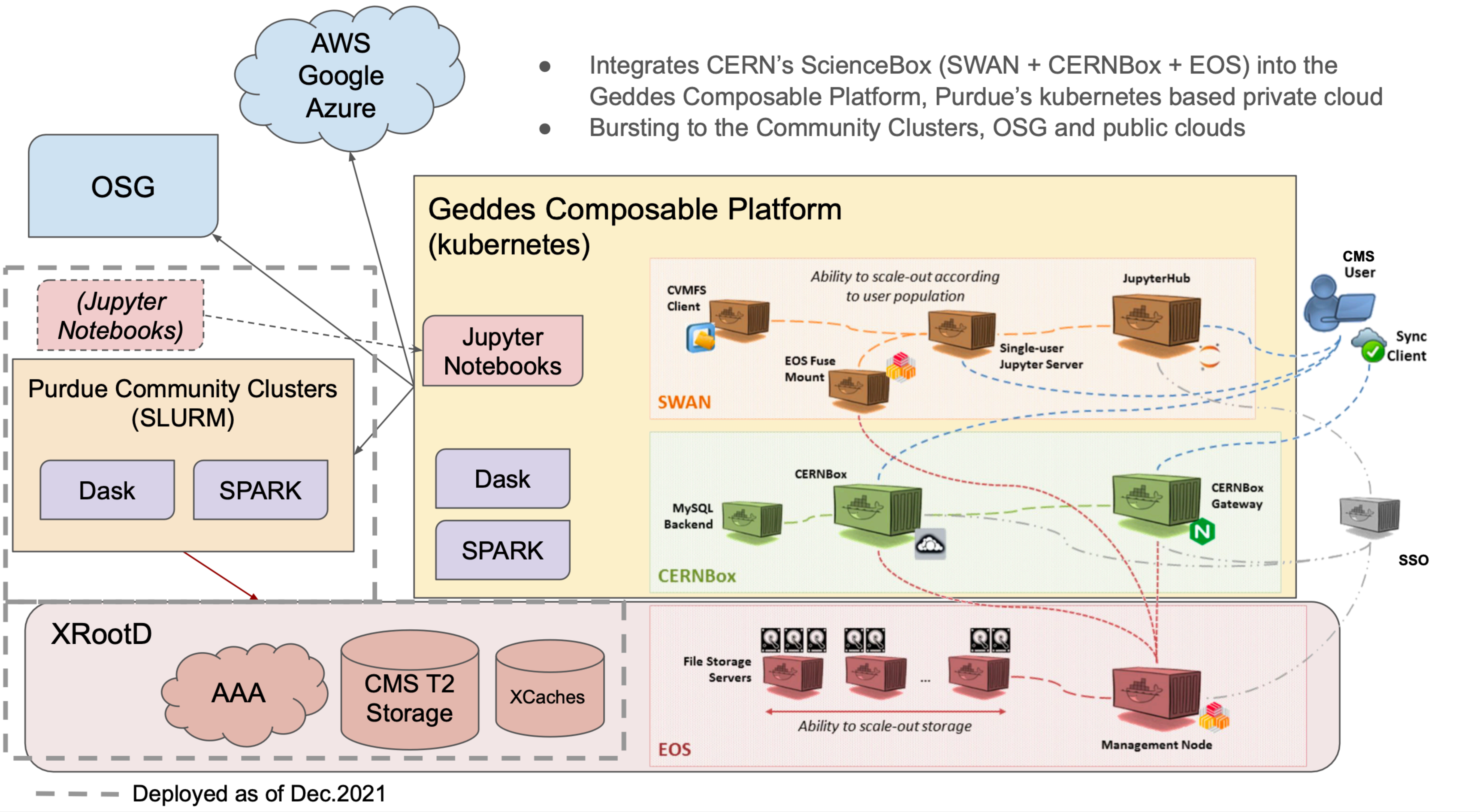
- Login
  - Key based with MIT account (sponsored guest accounts?)
  - CMS data access authentication x509 for now
- Work environment
  - Load balanced JupyterHub access, Coffea type of analysis
  - Dask sitting on top of MIT Tier-3/Tier-2 centers
  - HTCondor and Slurm as batch managers
- Data access optimization
  - Tiered storage seems an obvious candidate for ‘sophisticated’ optimization of storage… work in progress:
    - 50 TB of NVMe should function as a hot cache for most accessed data
    - Tape is ideal candidate for rarely used data or just as safety net to recover from disaster

## AF @ Purdue Status and Plans

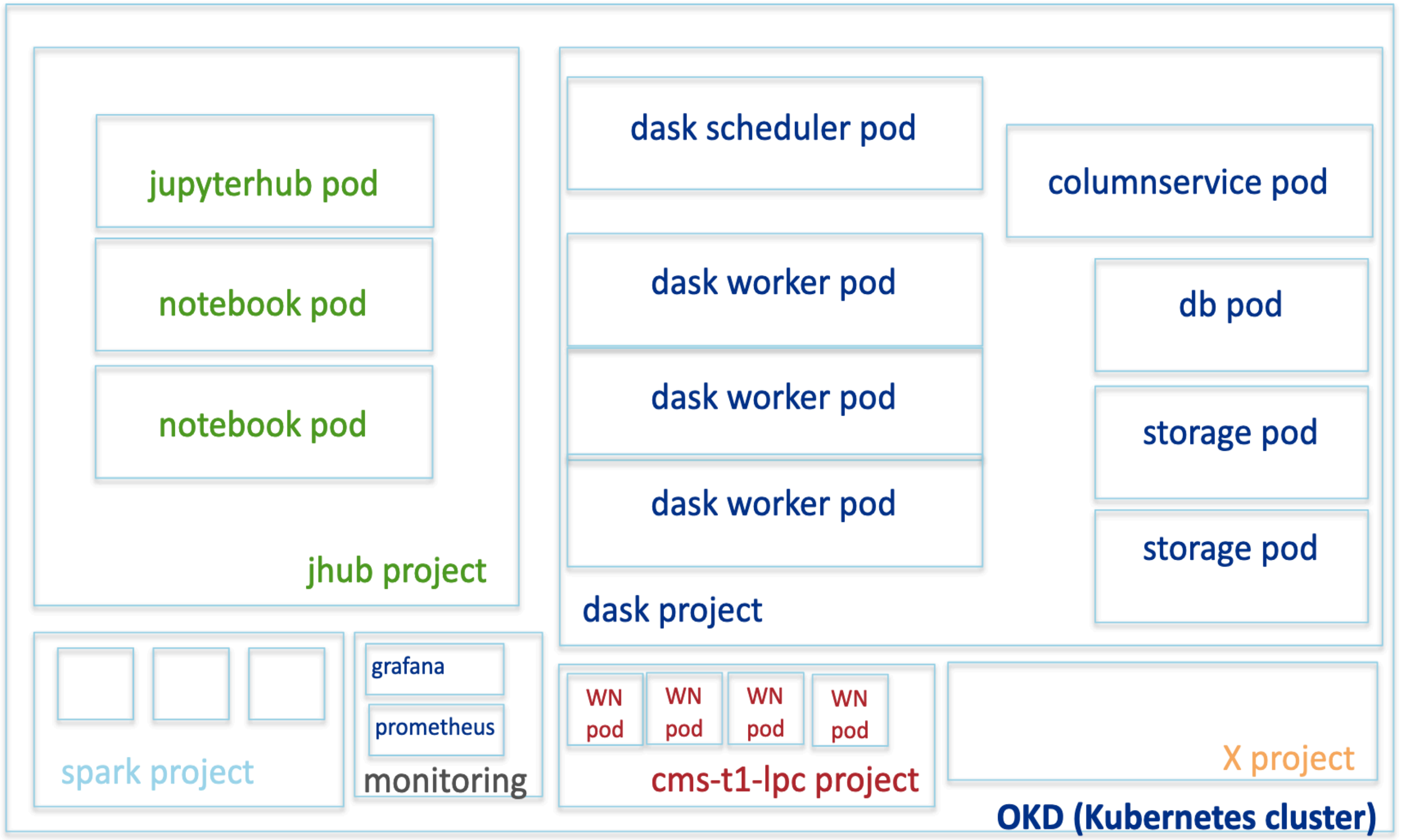
- The Purdue CMS T2 provides interactive AF capabilities for distributed physics data analysis since 2020, utilizing both interactive SSH sessions and JupyterHub to scale DASK/Spark clusters on HPC systems to over 1000 cores in parallel.
  - In that configuration the AF at Purdue was used in a CMS publication
    - DOI: 10.1007/JHEP01(2021)148) and in multiple ongoing analyses
  - MuonHLT upgrades, H→μμ Snowmass, Z'→ll, Top quark spin correlation
- In 2021 Purdue received USCMS funding for dedicated AF hardware, and our design evolved to include new AF capabilities based on CERN's ScienceBox (EOS, CERNBox, SWAN) running in Kubernetes, and leverage the new Geddes Composable Platform, a Kubernetes-based "Community Cloud" resource at Purdue.
  - Provides user-defined virtual clusters via DASK and Spark, for massively parallel user analyses based on coffea framework.
  - Integrates with Purdue's Kubernetes-based private cloud 'Geddes', and Purdue's Community Clusters.
  - Investigating OSG and public cloud integration in the future.
- Geddes Composable Platform
  - Purdue Research Computing has just built the Geddes Composable Platform - a private cloud resource based on Rancher and Kubernetes. This "Community Cloud" resource is a platform for flexible, scalable and reproducible scientific data analysis.
  - In June 2021, Purdue received NSF funding to build out a private campus cloud focused on data analytics and machine learning. Synergies with AF effort funded by USCMS
- The new hardware has already been received, and the upgrade will take place over the course of 2022 in close collaboration with USCMS Operations Program.

# AF @ Purdue Conceptual Layout

- Integrates CERN's ScienceBox (SWAN + CERNBox + EOS) into the Geddes Composable Platform, Purdue's kubernetes based private cloud
- Bursting to the Community Clusters, OSG and public clouds



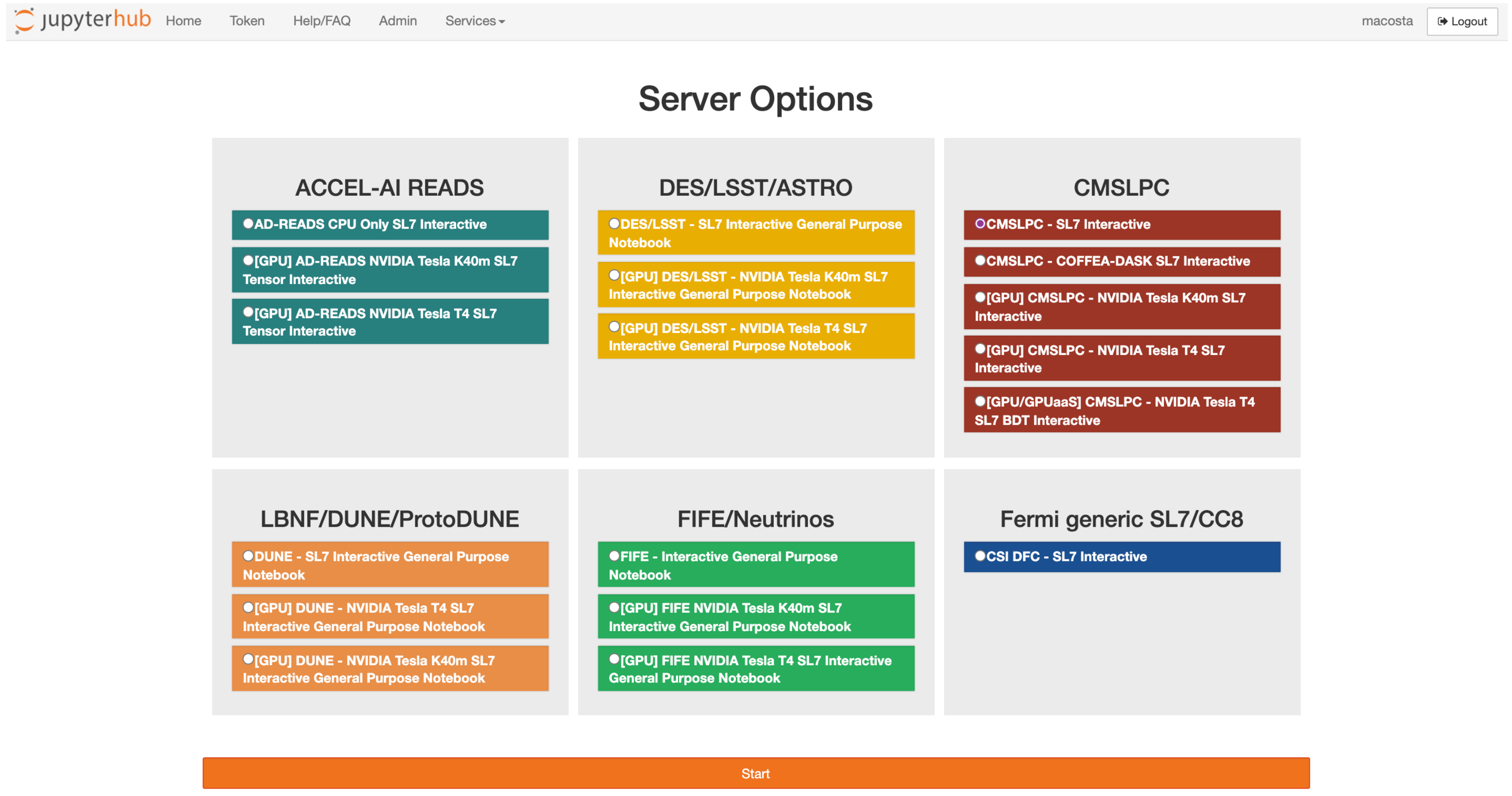
# Elastic Analysis Facility @ FNAL



From Burt's slides at OSG AHM: https://indico.fnal.gov/event/22127/contributions/194934/attachments/133990/165498/Elastic_AF_-_OSG_USLHC.pdf

| Secure | Integrated | Multi-vo | DevOps: | Other horizons: |
|---|---|---|---|---|
| •LDAP and VPN login, Kerberos. Docker image audits, and mitigation strategies put in place for data preservation and least privilege guarantee. | •Ferry, Htcondor, dask-gateway, spark, triton | •user management, centralized authorization, specialized environments, large-ish cvmfs infrastructure in place via NFS auto-scalable pods | •CI/CD pipelines for all environments, CPU and GPU flavors | •Now supporting Fermilab's Accelerator division Edge AI. Hoping to foster effort from SCD and AD on designing analysis facilities beyond SCD<br>•Collaborating with the Dask team on developing a plugin to integrate Dask Gateway with HTCondor, coming soon |

# Elastic AF: A Multi-Experiment AF



Server Options

ACCEL-AI READS
- AD-READS CPU Only SL7 Interactive
- [GPU] AD-READS NVIDIA Tesla K40m SL7 Tensor Interactive
- [GPU] AD-READS NVIDIA Tesla T4 SL7 Tensor Interactive

DES/LSST/ASTRO
- DES/LSST - SL7 Interactive General Purpose Notebook
- [GPU] DES/LSST - NVIDIA Tesla K40m SL7 Interactive General Purpose Notebook
- [GPU] DES/LSST - NVIDIA Tesla T4 SL7 Interactive General Purpose Notebook

CMSLPC
- CMSLPC - SL7 Interactive
- CMSLPC - COFFEA-DASK SL7 Interactive
- [GPU] CMSLPC - NVIDIA Tesla K40m SL7 Interactive
- [GPU] CMSLPC - NVIDIA Tesla T4 SL7 Interactive
- [GPU/GPUaaS] CMSLPC - NVIDIA Tesla T4 SL7 BDT Interactive

LBNF/DUNE/ProtoDUNE
- DUNE - SL7 Interactive General Purpose Notebook
- [GPU] DUNE - NVIDIA Tesla T4 SL7 Interactive General Purpose Notebook
- [GPU] DUNE - NVIDIA Tesla K40m SL7 Interactive General Purpose Notebook

FIFE/Neutrinos
- FIFE - Interactive General Purpose Notebook
- [GPU] FIFE NVIDIA Tesla K40m SL7 Interactive General Purpose Notebook
- [GPU] FIFE NVIDIA Tesla T4 SL7 Interactive General Purpose Notebook

Fermi generic SL7/CC8
- CSI DFC - SL7 Interactive

- Started as a USCMS project but has grown to be a multi-experiment project providing all services to multiple FNAL experiments.
  - EAF heard from and are actively collaborating with YorkU/Compute Canada for a prototype EAF for DUNE.
  - EAF developed more than 15 environments for experiments with dedicated CVMFS mounts, shared storage and specific scientific software, all in compliance with DOE cybersecurity requirements
  - EAF started collaborating with Fermilab's Accelerator Divison and designed an environment for the READS project Accelerator Real-time Edge AI for Distributed Systems (READS)
- BDT/ML analysis on local and remote GPUs via the EAF Triton Server GPU pod

## Conclusions and Outlook

- New tools like RDataFrames and Awkward Arrays make data analysis using the python ecosystem feasible and fast
  - Compiled-code speeds with the flexibility and expressivity of python
  - Notebook integration with these fast tools means we can approach data exploration in significantly more interactive ways
- Four AF efforts within USCMS heavily focused on deploying modern workflows
  - Providing the usual terminal access as well as notebooks predominantly through JupyterHub
  - Coffea-casa, ElasticAF well-advanced on interface & access
- Each effort focusing on different aspects of eventual common goal
  - Healthy (and natural!) split between software infrastructure, hardware infrastructure, and multi-tenancy
- Expect scale-up, benchmarking, (more) publications using these facilities over the course of 2022
  - Exciting times ahead, and best-practices will emerge!

## Extras

## Concrete Example

```
void MyClass::Loop() {
  size_t nEvents;
  // load...

  for (Long64_t iEvent=0; iEvent<nEvents; iEvent++) {
    double MET_pt;
    int nElectron;
    double * Electron_pt;
    double * Electron_eta;
    // load...

    if ( MET_pt > 100. ) continue;

    for(size_t iEl=0; iEl<nElectron; ++iEl) {
      if ( Electron_pt[iEl] > 30. ) {
        hist->Fill(Electron_eta[iEl]);
      }
    }
  }
}
```

Event loop

## Concrete Example

```
void MyClass::Loop() {
  size_t nEvents;
  // load...

  for (Long64_t iEvent=0; iEvent<nEvents; iEvent++) {
    double MET_pt;
    int nElectron;
    double * Electron_pt;
    double * Electron_eta;
    // load...

    if ( MET_pt > 100. ) continue;

    for(size_t iEl=0; iEl<nElectron; ++iEl) {
      if ( Electron_pt[iEl] > 30. ) {
        hist->Fill(Electron_eta[iEl]);
      }
    }
  }
}
```

Event loop

```
void MyClass::Loop() {
  size_t nEvents;
  double * MET_pt;
  int * nElectron;
  size_t nElectron_flat;
  double * Electron_pt;
  double * Electron_eta;
  // load...

  bool * eventmask = allocate(nEvents);
  for (size_t i=0; i<nEvents; i++)
    eventmask[i] = MET_pt[i] > 100.;

  bool * entrymask = allocate(nElectron_flat);
  for (size_t i=0; i<nElectron_flat; ++i)
    entrymask[i] = Electron_pt[i] > 30.;

  bool * entrymask2 = allocate(nElectron_flat);
  size_t * parents = get_parents(nEvents, nElectron);
  for (size_t i=0; i<nElectron_flat; ++i)
    entrymask2[i] = eventmask[parents[i]] & entrymask[i];

  double * take_result = allocate(nElectron_flat);
  size_t idx = 0;
  for (size_t i=0; i<nElectron_flat; ++i)
    if ( entrymask2[i] )
      take_result[idx++] = Electron_eta[i];

  for (size_t i=0; i<idx; i++)
    hist->Fill(take_result[i]);
}
```

Columnar

## Concrete Example

```
void MyClass::Loop() {
 size_t nEvents;
 // load...

 for (Long64_t iEvent=0; iEvent<nEvents; iEvent++) {
  double MET_pt;
  int nElectron;
  double * Electron_pt;
  double * Electron_eta;
  // load...

  if ( MET_pt > 100. ) continue;

  for(size_t iEl=0; iEl<nElectron; ++iEl) {
   if ( Electron_pt[iEl] > 30. ) {
    hist->Fill(Electron_eta[iEl]);
   }
  }
 }
}
```

Event loop

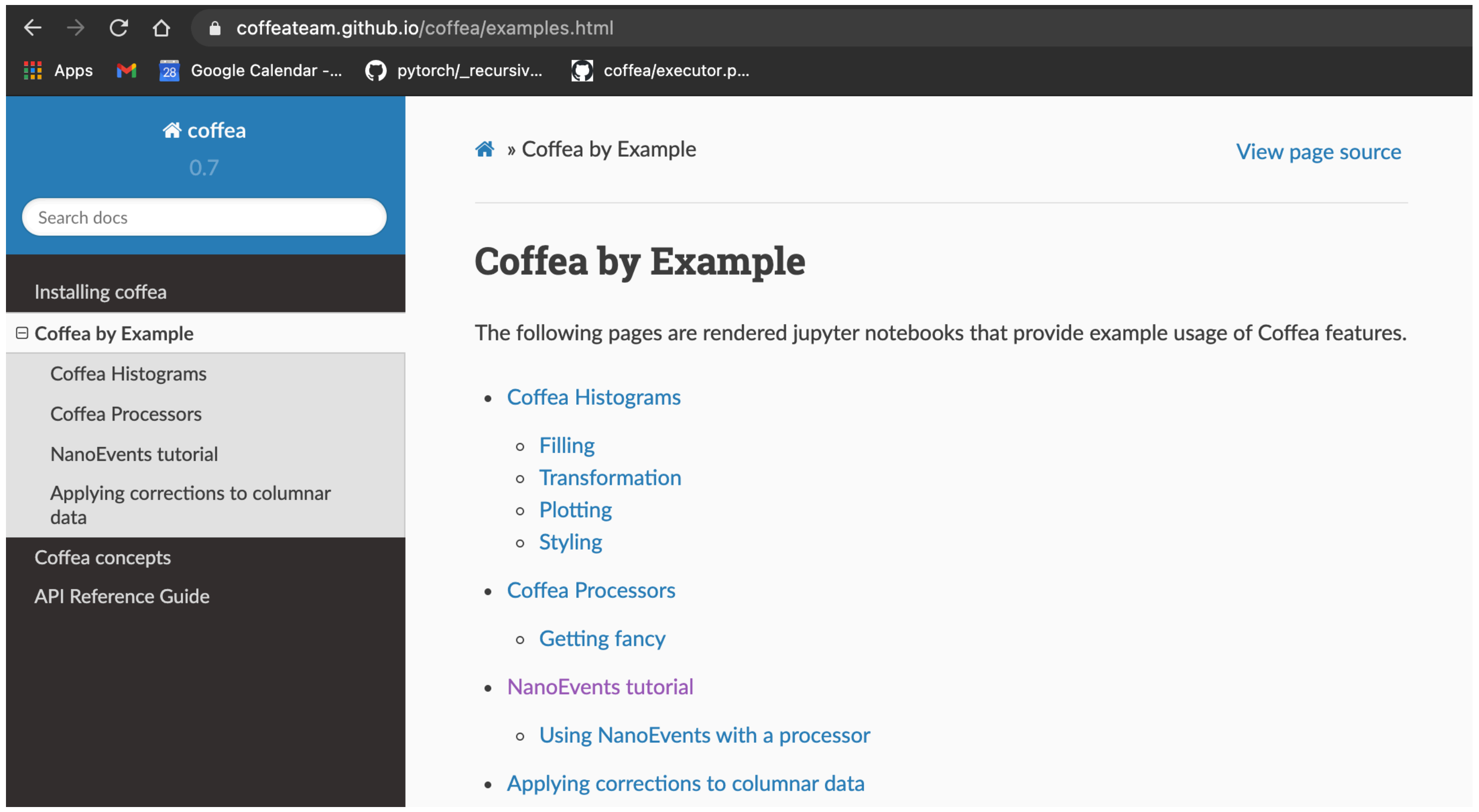
```
cut = (events.MET.pt < 100.) & (events.Electron.pt > 30.)
hist.fill(eta=events.Electron.eta[cut].flatten())
```

Columnar

# Example of Physics Code

```python
class Q8Processor(processor.ProcessorABC):
    """For events with at least three light leptons and a same-flavor
    opposite-charge light lepton pair, find such a pair that has the
    invariant mass closest to 91.2 GeV in each event and plot the transverse
    mass of the system consisting of the missing transverse momentum and
    the highest-p_T light lepton not in this pair.
    """

    def process(self, events):
        events["Electron", "pdgId"] = -11 * events.Electron.charge
        events["Muon", "pdgId"] = -13 * events.Muon.charge
        events["leptons"] = ak.concatenate([events.Electron, events.Muon], axis=1,)
        events = events[ak.num(events.leptons) >= 3]

        pair = ak.argcombinations(events.leptons, 2, fields=["l1", "l2"])
        pair = pair[(events.leptons[pair.l1].pdgId == -events.leptons[pair.l2].pdgId)]
        with np.errstate(invalid="ignore"):
            pair = pair[
                ak.singletons(
                    ak.argmin(
                        abs(
                            (events.leptons[pair.l1] + events.leptons[pair.l2]).mass
                            - 91.2
                        ),
                        axis=1,
                    )
                )
            ]
        events = events[ak.num(pair) > 0]
        pair = pair[ak.num(pair) > 0][:, 0]

        l3 = ak.local_index(events.leptons)
        l3 = l3[(l3 != pair.l1) & (l3 != pair.l2)]
        l3 = l3[ak.argmax(events.leptons[l3].pt, axis=1, keepdims=True)]
        l3 = events.leptons[l3][:, 0]

        mt = np.sqrt(2 * l3.pt * events.MET.pt * (1 - np.cos(events.MET.delta_phi(l3))))
        return (
            hist.Hist.new.Reg(
                100, 0, 200, name="mt", label=r"$\ell$-MET transverse mass [GeV]"
            )
            .Double()
            .fill(mt)
        )

    def postprocess(self, accumulator):
        return accumulator
```

# Coffea documentation and support



coffeateam.github.io/coffea/examples.html

coffea 0.7

Search docs

Installing coffea

Coffea by Example

Coffea Histograms

Coffea Processors

NanoEvents tutorial

Applying corrections to columnar data

Coffea concepts

API Reference Guide

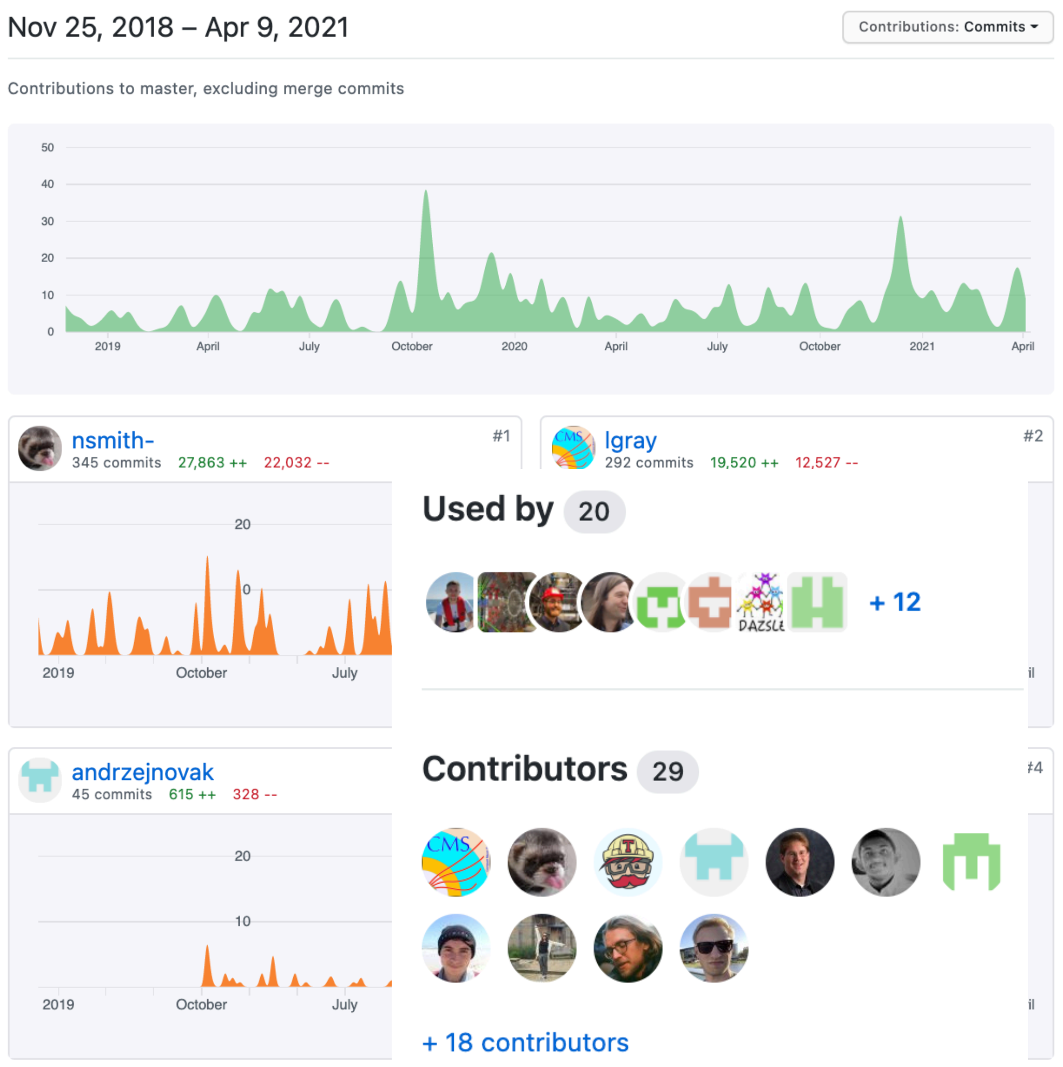
» Coffea by Example

View page source

**Coffea by Example**

The following pages are rendered jupyter notebooks that provide example usage of Coffea features.

- Coffea Histograms
  - Filling
  - Transformation
  - Plotting
  - Styling
- Coffea Processors
  - Getting fancy
- NanoEvents tutorial
  - Using NanoEvents with a processor
- Applying corrections to columnar data



Nov 25, 2018 – Apr 9, 2021

Contributions: Commits

Contributions to master, excluding merge commits

nsmith- #1 345 commits 27,863 ++ 22,032 --

lgray #2 292 commits 19,520 ++ 12,527 --

andrzejnovak 45 commits 615 ++ 328 --

Used by 20 + 12

Contributors 29

+ 18 contributors

- Extensive documentation of code base in multiple forms
  - Basic documentation website
  - Jupyter Notebooks
  - YouTube videos
- Significant use by other projects, large contributor base
  - Intend to keep this project going for a long time
  - 79 direct forks of the coffea repository
  - Standard open-source core + community supported model

# Writing Physics Analysis Code in Coffea (I)

```python
import awkward as ak
import hist
import matplotlib.pyplot as plt
from coffea import nanoevents, processor

processor.NanoAODSchema.warn_missing_crossrefs = False


class Q3Processor(processor.ProcessorABC):
    """Plot the p_T of jets with |η| < 1."""

    def process(self, events):
        return (
            hist.Hist.new.Reg(100, 0, 200, name="ptj", label="Jet $p_{T}$ [GeV]")
            .Double()
            .fill(ak.flatten(events.Jet[abs(events.Jet.eta) < 1].pt))
        )

    def postprocess(self, accumulator):
        return accumulator


if __name__ == "__main__":
    runner = processor.Runner(
        executor=processor.FuturesExecutor(workers=4),
        schema=nanoevents.NanoAODSchema,
        chunksize=2 ** 19,
    )

    output = runner(
        fileset={"SingleMu": ["Run2012B_SingleMu.root"]},
        treename="Events",
        processor_instance=Q3Processor(),
    )
    output.plot()
    plt.gcf().savefig("pt.pdf")
```

# Physics Code (II)

```python
class Q8Processor(processor.ProcessorABC):
    """For events with at least three light leptons and a same-flavor
    opposite-charge light lepton pair, find such a pair that has the
    invariant mass closest to 91.2 GeV in each event and plot the transverse
    mass of the system consisting of the missing transverse momentum and
    the highest-p_T light lepton not in this pair.
    """

    def process(self, events):
        events["Electron", "pdgId"] = -11 * events.Electron.charge
        events["Muon", "pdgId"] = -13 * events.Muon.charge
        events["leptons"] = ak.concatenate([events.Electron, events.Muon], axis=1,)
        events = events[ak.num(events.leptons) >= 3]

        pair = ak.argcombinations(events.leptons, 2, fields=["l1", "l2"])
        pair = pair[(events.leptons[pair.l1].pdgId == -events.leptons[pair.l2].pdgId)]
        with np.errstate(invalid="ignore"):
            pair = pair[
                ak.singletons(
                    ak.argmin(
                        abs(
                            (events.leptons[pair.l1] + events.leptons[pair.l2]).mass
                            - 91.2
                        ),
                        axis=1,
                    )
                )
            ]
        events = events[ak.num(pair) > 0]
        pair = pair[ak.num(pair) > 0][:, 0]

        l3 = ak.local_index(events.leptons)
        l3 = l3[(l3 != pair.l1) & (l3 != pair.l2)]
        l3 = l3[ak.argmax(events.leptons[l3].pt, axis=1, keepdims=True)]
        l3 = events.leptons[l3][:, 0]

        mt = np.sqrt(2 * l3.pt * events.MET.pt * (1 - np.cos(events.MET.delta_phi(l3))))
        return (
            hist.Hist.new.Reg(
                100, 0, 200, name="mt", label=r"$\ell$-MET transverse mass [GeV]"
            )
            .Double()
            .fill(mt)
        )

    def postprocess(self, accumulator):
        return accumulator
```

## Coffea Corrections

```
from coffea.btag_tools import BTagScaleFactor

btag_sf = BTagScaleFactor("data/DeepCSV_102XSF_V1.btag.csv.gz", "medium")

print("SF:", btag_sf.eval("central", events.Jet.hadronFlavour, abs(events.Jet.eta), events.Jet.pt))
print("systematic +:", btag_sf.eval("up", events.Jet.hadronFlavour, abs(events.Jet.eta), events.Jet.pt))
```

```
rochester_data = coffea.lookup_tools.txt_converters.convert_rochester_file(
    "tests/samples/RoccoR2018.txt.gz", loaduncs=True
)
rochester = coffea.lookup_tools.rochester_lookup.rochester_lookup(rochester_data)
data_k = rochester.kScaleDT(
    events.Muon.charge, events.Muon.pt, events.Muon.eta, events.Muon.phi
)
```

- Via awkward arrays coffea can process most if not all tabular/columnar data
  - Can ingest parquet, root by default and extensible to any reasonable file format
- Often we want to apply corrections to our data or make variations to estimate the effect of systematics
  - Coffea has tools that make bookkeeping easy for this task for all corrections used by CMS
  - We’ve done our best to make sure these tools are well validated and usable in analysis
  - Coffee supports correctionlib out of the box in latest versions

## Coffea + ML

```
counts = awkward.num(diphotons, axis=-1)
bdt_inputs = numpy.column_stack(
    [awkward.to_numpy(awkward.flatten(bdt_vars[name])) for name in var_order]
)
tempmatrix = xgboost.DMatrix(bdt_inputs, feature_names=var_order)
scores = diphoton_mva.predict(tempmatrix)
diphotons["bdt_score"] = awkward.unflatten(scores, counts)
```

```
import torch
import awkward as ak

model = torch.load('/some/model.pt')
x = ak.Array([[1., 2., 3.], [4., 5.], [6.]])
# find the x with largest probability in a given event, assuming 'model' is trained to do that
probs = ak.softmax(ak.unflatten(model(torch.tensor(ak.flatten(x))).numpy(), ak.num(x)), axis=1)
```

- All ML toolkits natively use flat columnar data
  - Awkward arrays are compatible with all ML tools by default, and has auto-diff
  - No show-stoppers for interface compatibility or using *any* ML framework in analysis
- Analyses on CMS and ATLAS are using the following ML tools with coffea
  - xgboost, TFLite, nVidia Triton
    - The list of available frameworks is mostly driven by package size (all of these are ~ 20MB each)
    - There is no "light" installation of PyTorch, and nearly every ML framework is covered by Triton
    - Preserving BDT-based analysis workflows, upgrading from TMVA to xgboost is seamless thanks to tmva-to-xgboost (already demonstrated to give same answers within rounding error)

## Coffea + Combine

```
outputFile = uproot.recreate(os.path.join(outdir, "M3_Output.root"))

outputFile["dataObs"] = hData.to_hist()

datasets = h.axis("dataset").identifiers()
systematics = h.axis("systematic").identifiers()
for _dataset in datasets:
    for _systematic in systematics:
        outputFile[f"{_dataset}_{_systematic}"] = h.integrate("dataset", _dataset).integrate("systematic", _systematic).to_hist()

outputFile.close()
```

- There is no well defined api for combine so the primary way to interact with it is by feeding it ROOT files full of histograms and preparing workspaces
- The uproot package now allows for writing histograms and TTrees to disk in the root file format
  - For histograms must project down to 3 dimensions or fewer
- Logic for creating models and workspaces can still be achieved via RooFit + PyRoot
  - If using tfcombine, SmooFit, jaxfit prototypes then alternative ways of building a fitting model are available

**Code example walkthrough:**

**CMS DAS TTGamma Long Exercise**

**Distributed computation on condor demo (LPC specific):**

**lpcjobqueue/simple_example.py**

**(lxplus demo available on demand)**