SWIFT HEP

SoftWare InFrastructure and Technology for High Energy Physics

SwiftHep workshop #3

Plans for a UK analysis challenge

Luke Kreczko for SWIFT-HEP WP5

University of BRISTOL

Outline

- Analysis workflows and the Analysis Work package
- Intersection with WP1: Data and workflow management
- The Analysis Grand Challenges
- Summary and outlook

Analysis Work Package

Analysis key points

Physics

Last mile of long chain of data recording and processing.

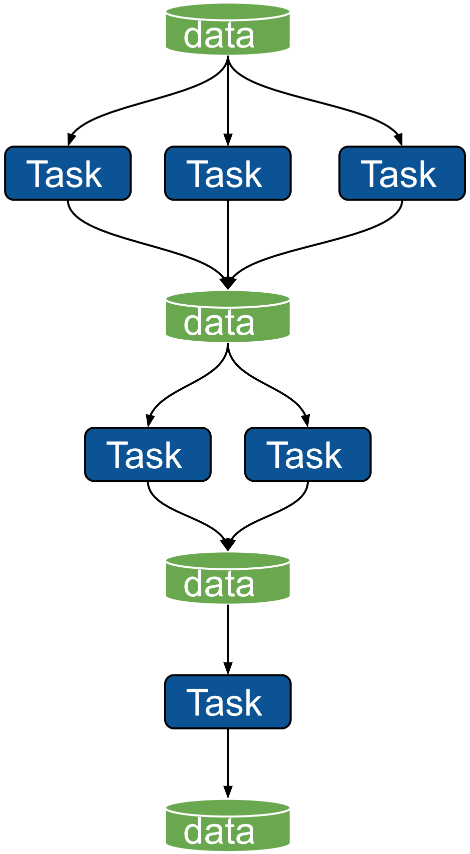
Goals: gain insight and create new knowledge

Computing

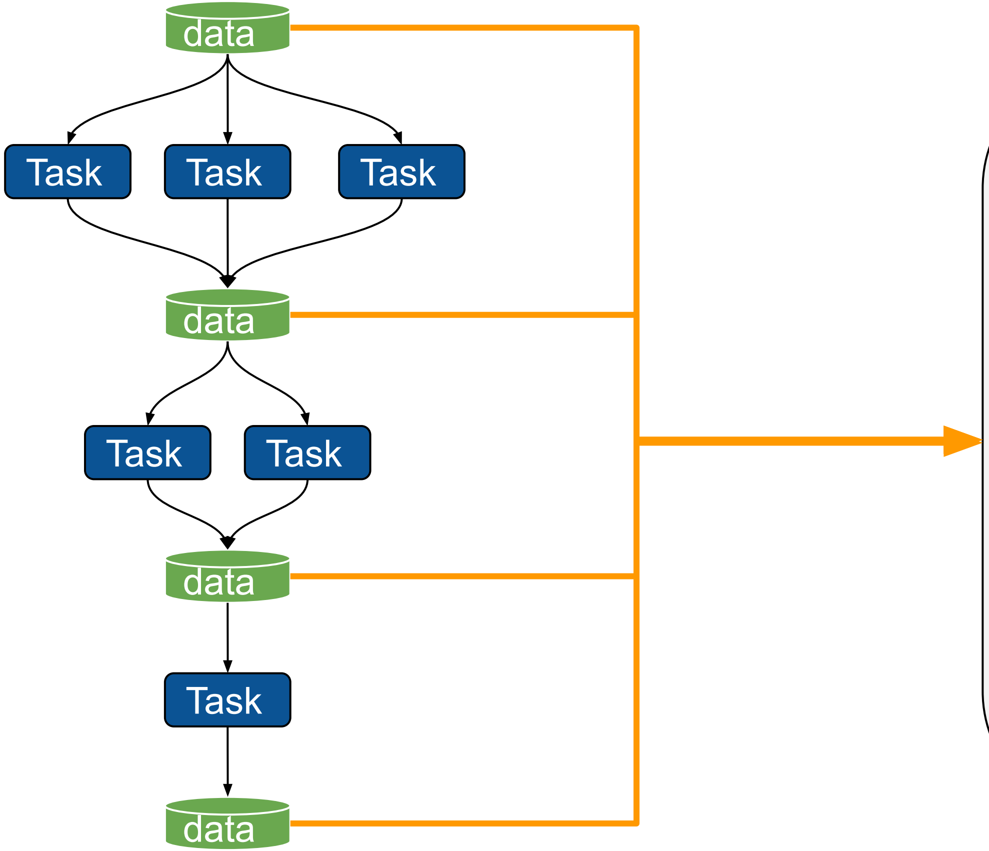
Analysis workflow (data + software) depends on experiment, analysis group, subset of data (signal + relevant backgrounds), analysis iteration.

Flexibility is paramount.

Anatomy of an analysis workflow



Anatomy of an analysis workflow

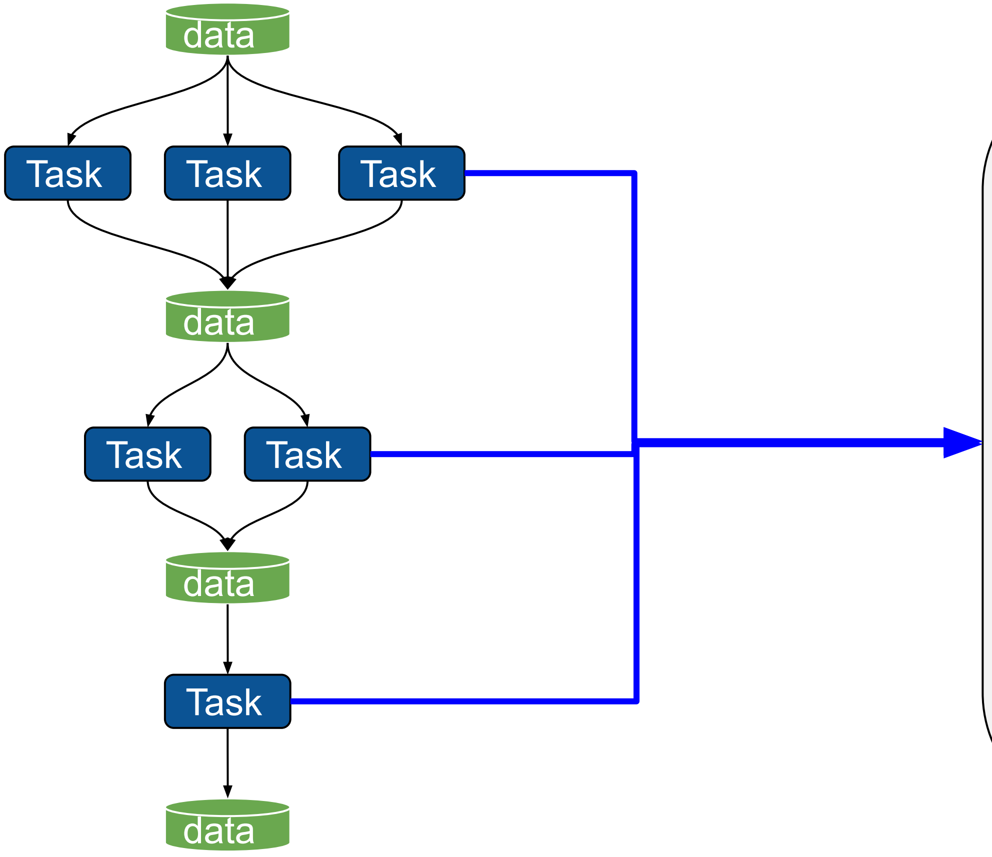


Data Formats

ROOT files: standard for input/output
Internally: Experiment/analysis specific

HDF5	npz
pandas	datacard
parquet	other?

Anatomy of an analysis workflow

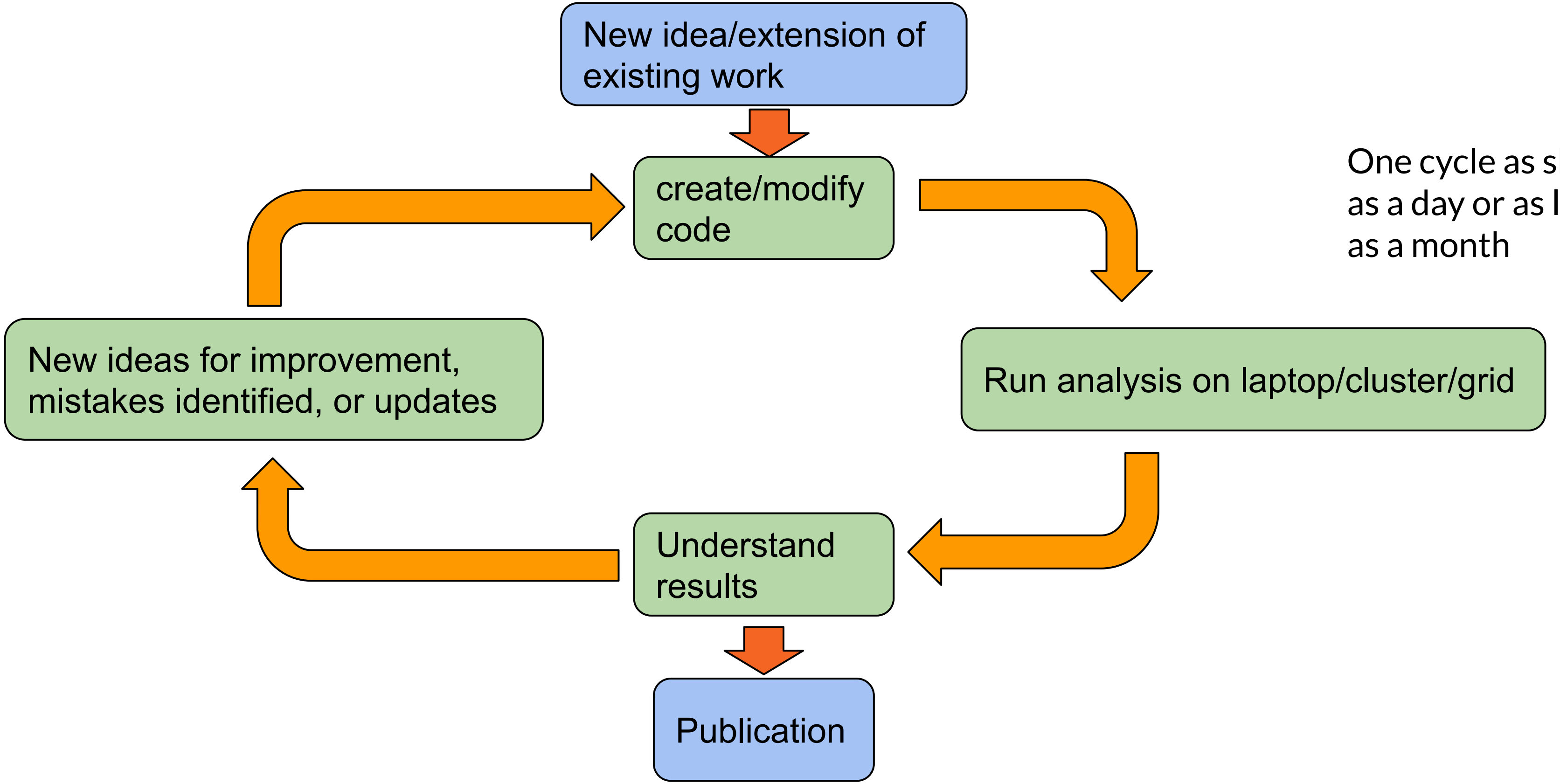


Processing

- Event loop vs vectorized processing
- Monoliths vs compute graphs
- GPU/FPGA capable vs strictly CPU
- Parallelizable vs strictly sequential
- Failure tolerance vs all or nothing
- Time sensitive vs “sometime next week”

- Varied resource requirements/efficiency

The cycle of analysis (an oversimplified view)

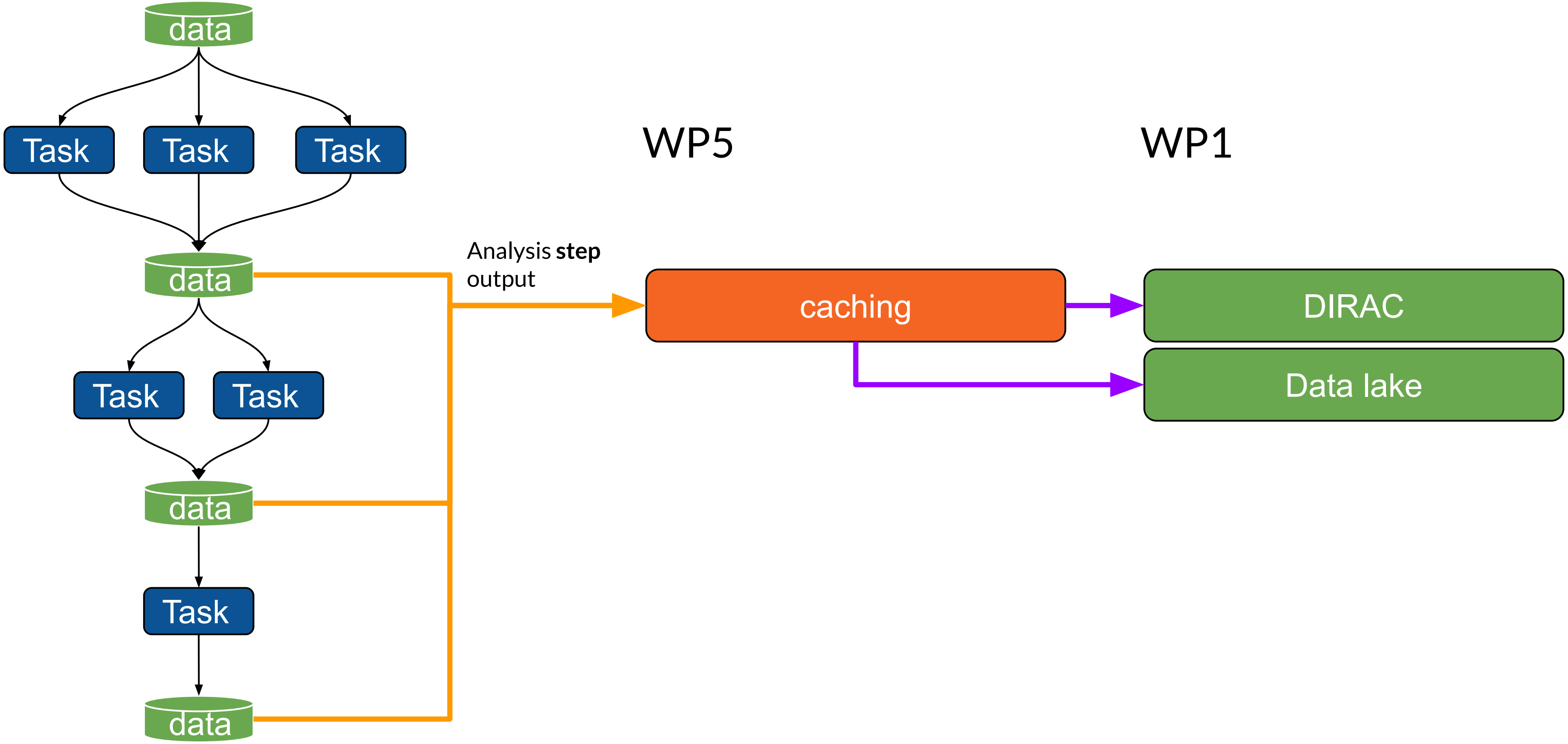


New idea/extension of existing work
create/modify code
Run analysis on laptop/cluster/grid
Understand results
New ideas for improvement, mistakes identified, or updates
Publication

One cycle as short as a day or as long as a month

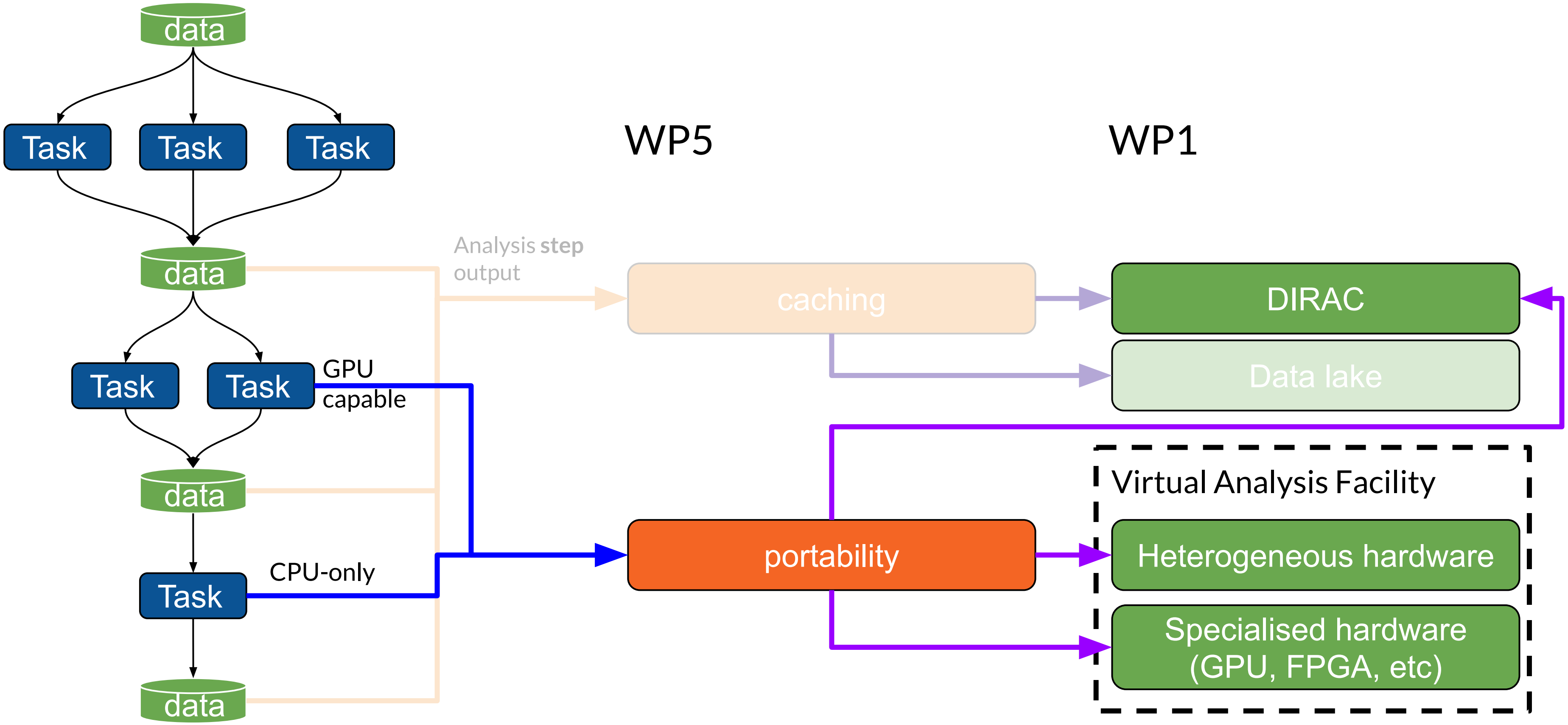
Work Package 5: Analysis

Analysis workflow



Work Package 5: Analysis

Analysis workflow

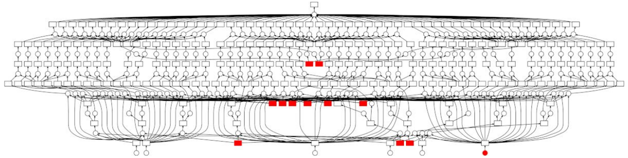


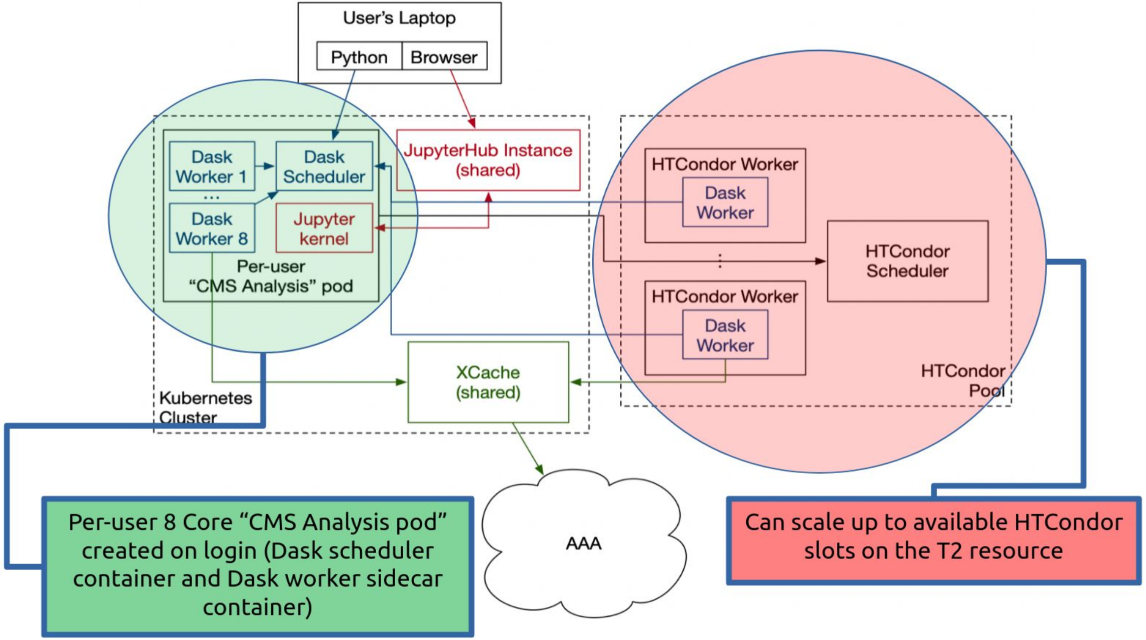
WP1 ⟷ WP5

(in practical terms)

Scheduling with coffea-casa

Uses Dask and dask-jobqueue





Per-user 8 Core "CMS Analysis pod" created on login (Dask scheduler container and Dask worker sidecar container)

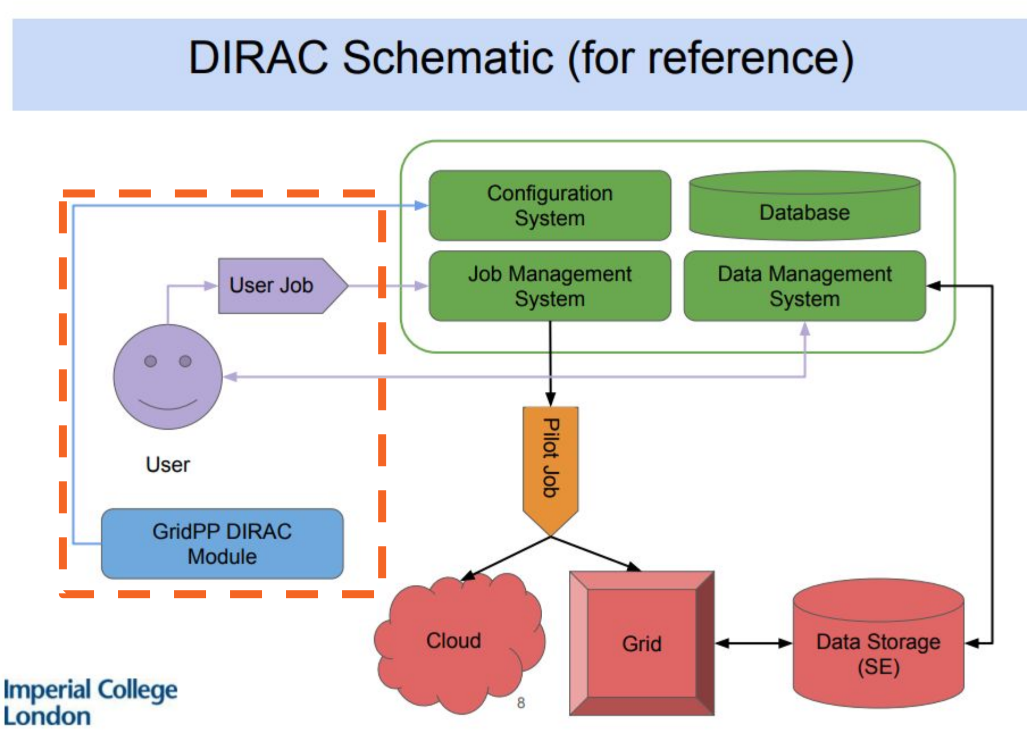
Can scale up to available HTCondor slots on the T2 resource

From coffea-casa docs

Scheduling with DIRAC

In a nutshell: scheduling across job management systems

Data management system for access to data lake (here caching)



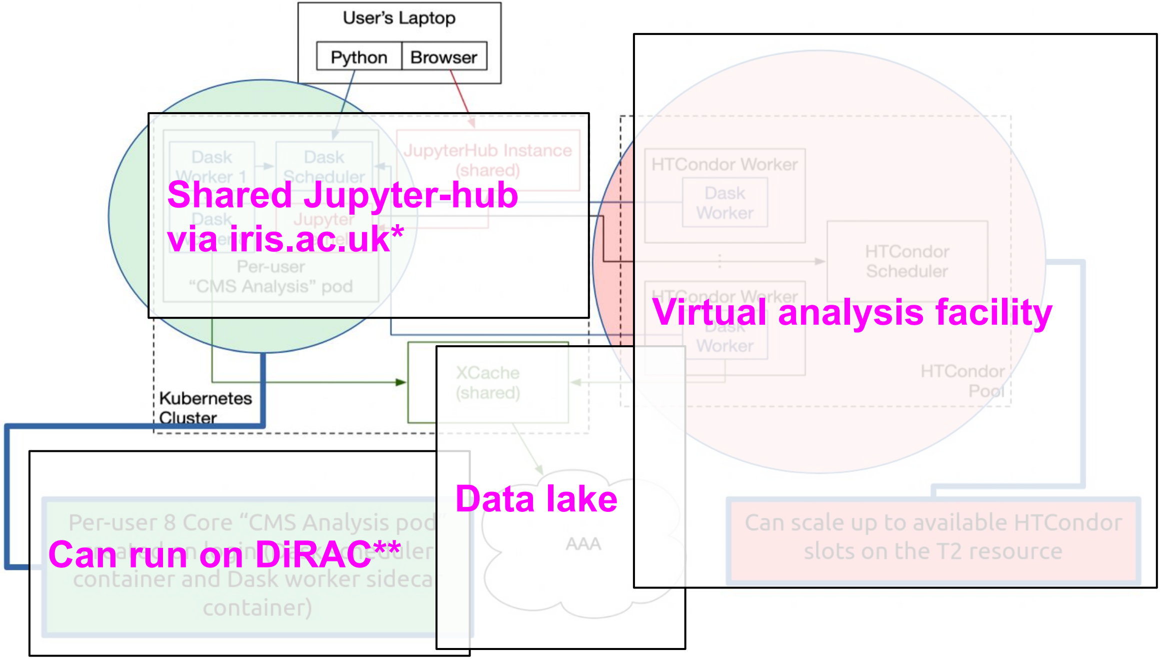
DIRAC Schematic (for reference)

Slide stolen from Janusz’s presentation at the [SWIFT-HEP May meeting]

SWIFT-HEP

“Adaptation”

As simple as adding DIRAC jobqueue to dask-jobqueue?


User’s Laptop
Python
Browser
Dask Worker 1
Dask Scheduler
JupyterHub Instance (shared)
Dask
Jupyter
Per-user “CMS Analysis” pod
Kubernetes Cluster
XCache (shared)
HTCondor Worker
Dask Worker
HTCondor Scheduler
HTCondor Worker
Dask Worker
HTCondor Pool
AAA
Per-user 8 Core “CMS Analysis pod”
container and Dask worker sidecar container)
Can scale up to available HTCondor slots on the T2 resource
Shared Jupyter-hub via iris.ac.uk*
Virtual analysis facility
Data lake
Can run on DiRAC**

*no relation to IRIS-HEP; **no relation to DIRAC

Concrete [starting] work items (1)

DiracJob and DiracJobQueueCluster in dask-jobqueue*

- Can use DIRAC command-line tools or python library
- In collaboration with DIRAC experts
  - Sensible defaults
  - Best way to communicate extra requirements (e.g. GPU, cached data)

Work here can then easily be migrated to Parsl and/or tested via joblib by volunteers

*we can start as an independent package and merge later

Concrete [starting] work items (2)

Storing (temporary) analysis cache on data lake

- Expiration dates: what is maximally reasonable? What makes sense on average?
- Permissions: users work in (dynamic) groups - What is the best approach for ACLs?
- Xrootd cache: Does it make sense to pre-fill input data based on scheduled DIRAC job?

Analysis Grand Challenges

Analysis Grand Challenges (IRIS-HEP)

IRIS-HEP are planning to verify work through several analysis grand challenges

Aiming for a realistic workflow, e.g.

- Existing analysis, their example: Higgs → tau tau
- Approx 200 TB of input data, their example: CMS NanoAOD
- Testing performance (speed, resource usage)
- Outputs: statistical inference, tables, control plots, HEP Data
- Other metrics: reproducibility of results (e.g. with REANA)

→ more info IRIS-HEP AGC Tools workshop, 25th of April 2022

Analysis Grand Challenges (SWIFT-HEP)

In SWIFT-HEP we can copy the main test with little extra effort*

But: can we involve analysis groups in the UK?

- Would need to provide documentation on the use of DiracJobQueue
- Need to allocate resources per group
- Need to make sure job wrappers and Analysis Facility monitoring capture all metrics (i.e. no additional work for users here)

*by swapping the dask-jobqueue configuration: HTCondor → DIRAC

Summary and Outlook

Analysis workflows can be quite challenging to optimize for

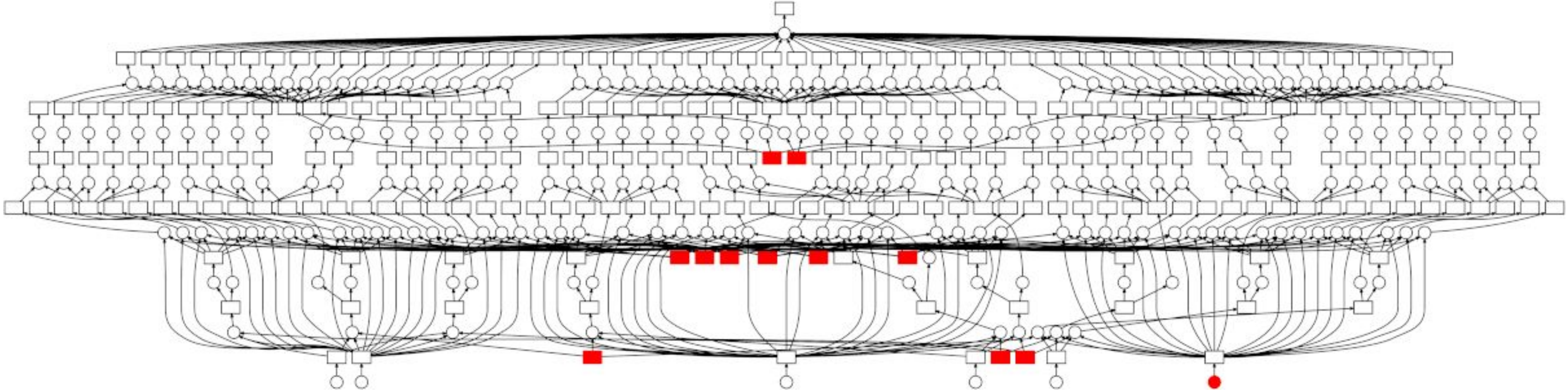
Collaboration between WP1 and WP5 and with IRIS-HEP for synergies

We could extend AGCs by including the UK community → extra bits driven by community

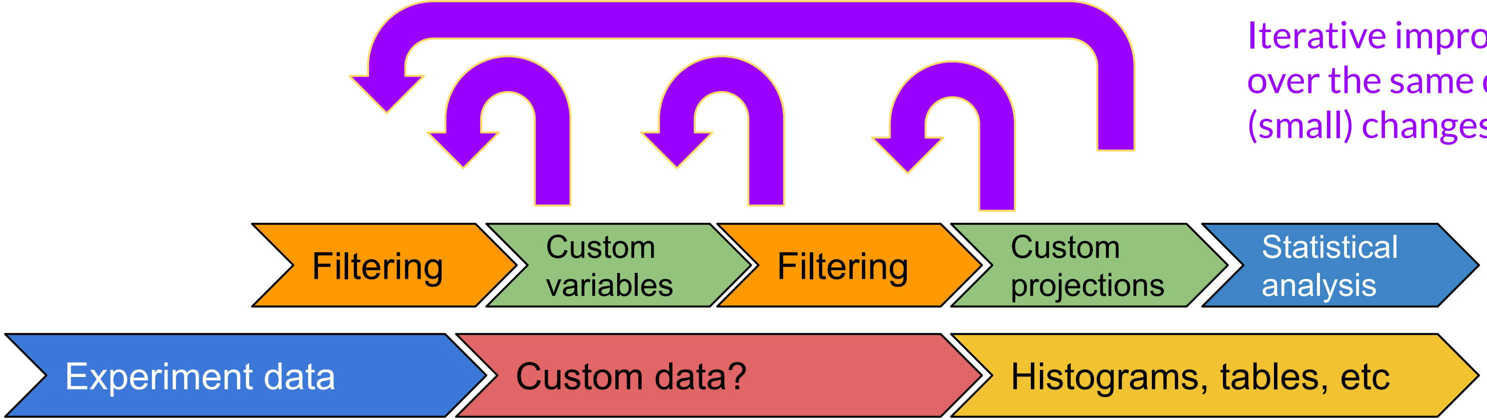
SWIFT-HEP WP5 to start next month

Backup slides

Analysis workflow example in Dask



Analysis pipeline example reality might differ

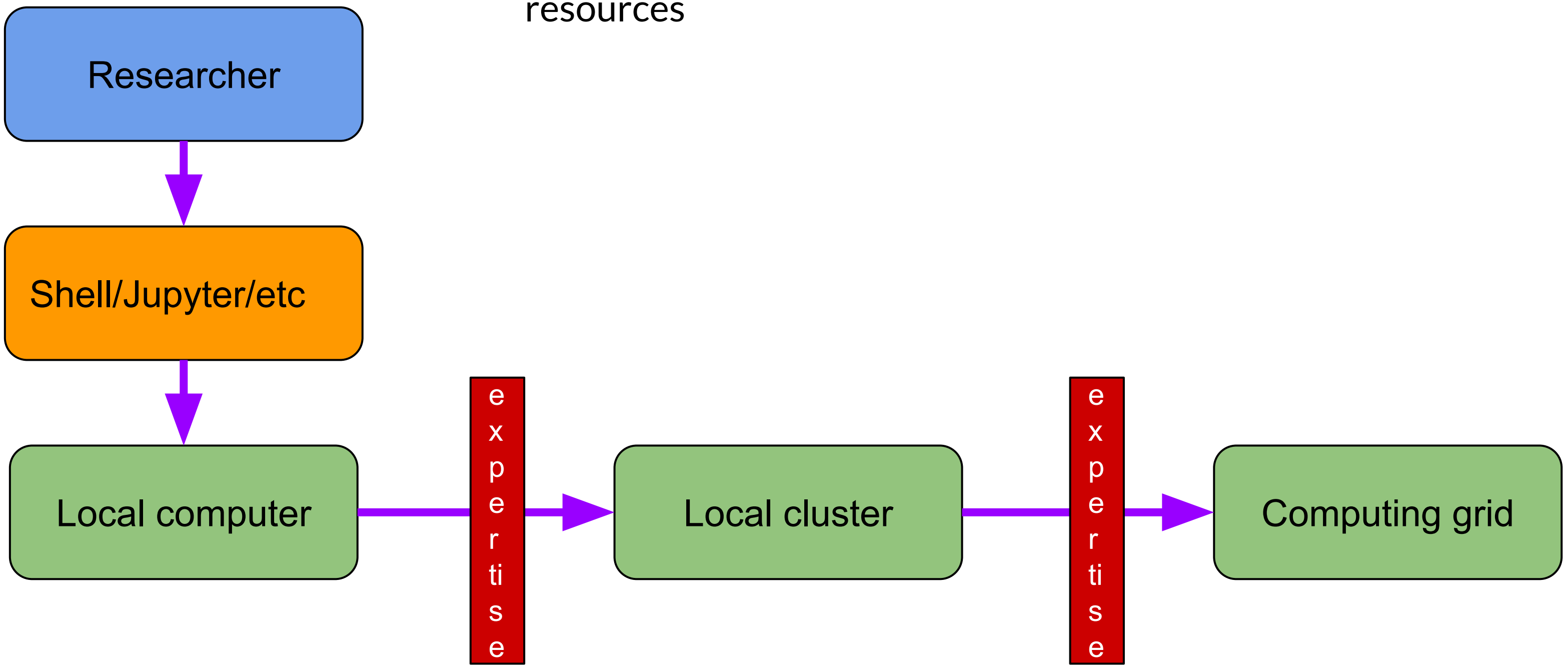


Iterative improvements - run over the same data with (small) changes

- Custom variables **might** include Machine Learning → training and inference on GPU
- **Depending** on underlying tools, statistical analysis can benefit from GPUs as well
- **Depending** on expertise, analysis code might be modular or one big block
- **Depending** on expertise each iteration will use resources efficiently, **or not**

Analysis Workflow: compute

As analysis needs increase, new expertise is needed to use more resources



Analysis Challenge

Large user-driven component → hard to optimize for every case

Inconsistent data use: new data sets, reprocessing of targeted data sets

Ideally, each iteration is as short as possible → “time to insight” low

iterative model == waste of computing resources?

Emerging trend: **interactive analysis**

Jupyter notebooks

Analysis “simplified”

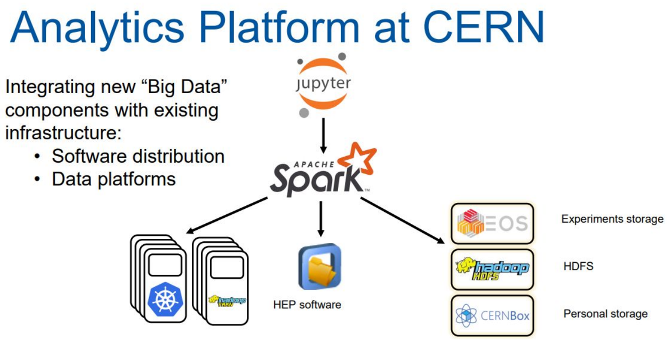
These kinds of workflows seem really desirable by the current generation of PhD students

Shifts a lot of “How to do distributed computing” to “What I want to get done” → declarative approaches are great for research

This disconnection allows experts to improve computing infrastructure “behind the scenes”

CERN's analytix cluster

Spark + Hadoop (link)

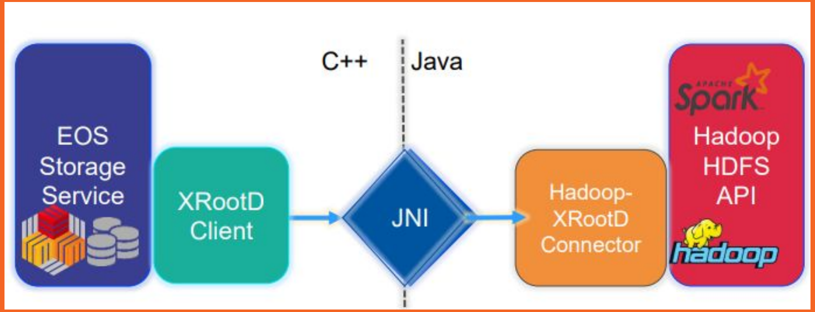


Analytics Platform at CERN

Integrating new "Big Data" components with existing infrastructure:

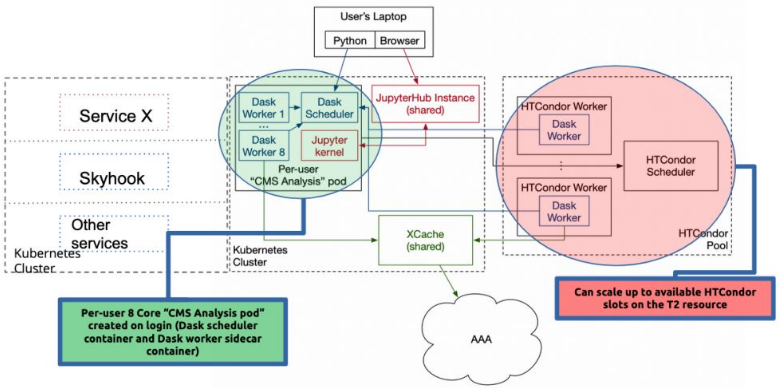
- Software distribution
- Data platforms

- Initially for log processing on Hadoop
- Can run ROOT analysis on Spark
- Accessible via **CERN's SWAN service (Jupyter)**
- Access to external storage via plugin



IRIS-HEP Coffea-casa

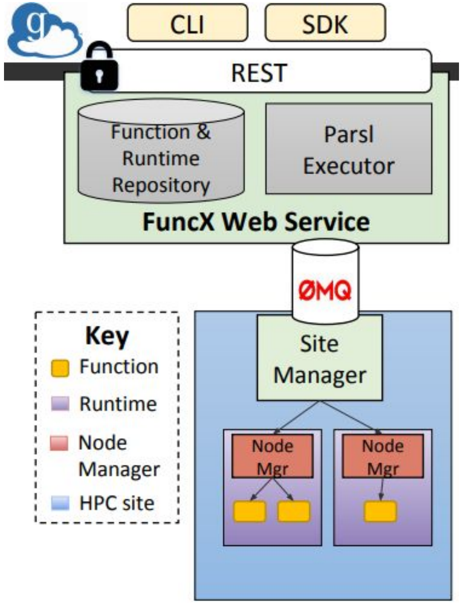
Analysis facility on top of an HTCondor cluster (link)



- Dask as a key component
- Uses TLS proxy (Traefik) to route requests from outside to the Dask cluster
- Dask-jobqueue for submitting to batch system (e.g HTCondor)
- More details in next talk

funcX

Federated function as a service (link)



- "Serverless" approach to compute (similar to FnProject)
- Reduces barriers to access distributed resources
- Low-latency, on-demand
- Can be used to build a catalogue of functions
- Functions can be deployed on special resources → "binding algorithms to hardware"

Hyper (Lux-Zeplin)

non-LHC analysis via Dask on HPC and HTC (see talk)

“Hyper is an uproot wrapper that lets you execute any Python code easily in parallel”

- Dask as a key component
- Dask-jobqueue for submitting to batch system
- Uses boost_histogram, uproot, numexpr & more
- Tested on a UK cluster and at NERSC
- Example for interactive distributed analysis without a dedicated analysis facility

IRIS-HEP

Analysis Grand Challenges
[AGCs]
(incl. ATLAS, CMS and WLCG)

Related IRIS-HEP workshop

Multiple challenges in the years 2022, 2023, 2025, 2027

Analysis: Demonstrate analysis system can cope with increased data volume while delivering enhanced functionality**

Data volume: realistically sized HL-LHC end-user analysis dataset (~ 200 TB)

Reproducibility and Reinterpretation

Interested in getting more experiments involved to broaden usability