University of BRISTOL

SWIFT HEP

SoftWare InFrastructure and Technology for High Energy Physics

SWIFT-HEP WP5 Analysis Systems

Sam Eriksen

22nd November 2023

Overview

- WP5 overview + roadmap
- Current progress in WP5
- Future for WP5

‘Monthly’ updates

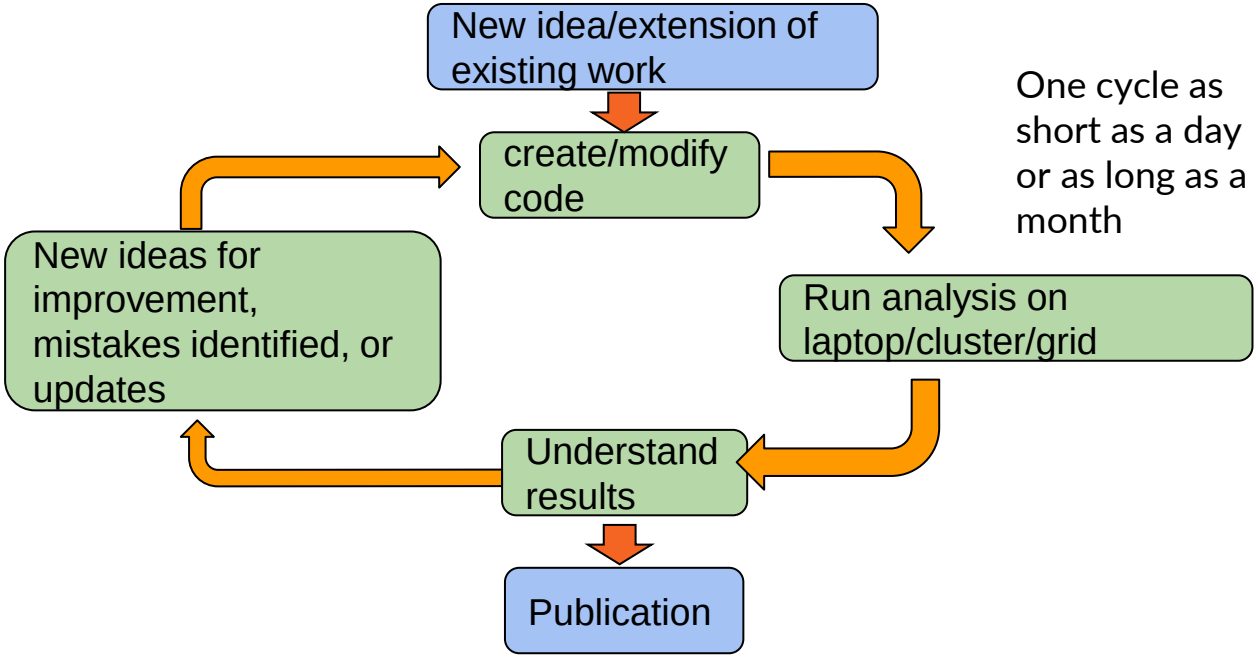
- September 2023
- May 2023
- March 2023
- February 2023

WP5: Analysis Systems

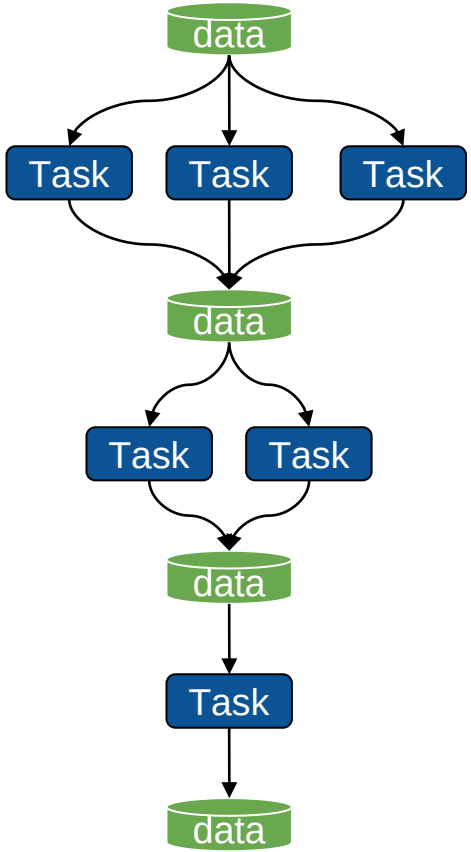
WP5: Analysis Systems

Run analysis workloads optimally on distributed resources

Analysis Anatomy

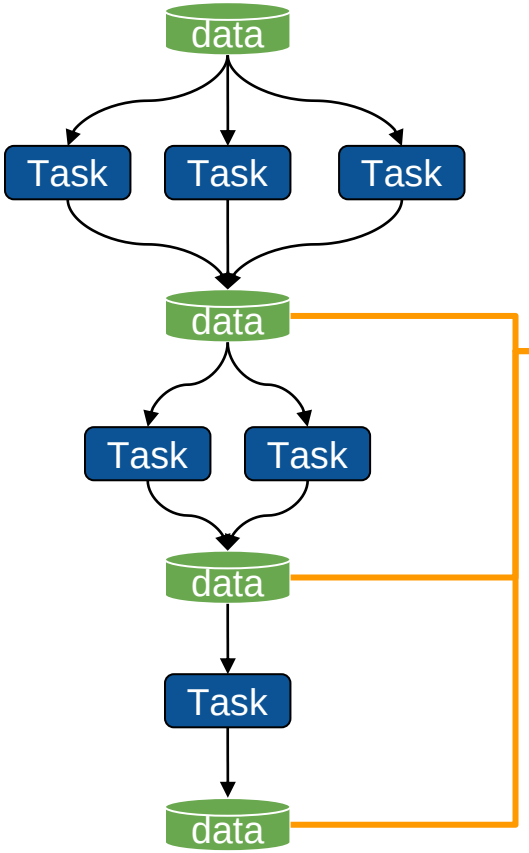


One cycle as short as a day or as long as a month



See talk by Luke Krezcko for some **BIG** Picture

Analysis Anatomy



WP5

WP1

Analysis **step** output

caching

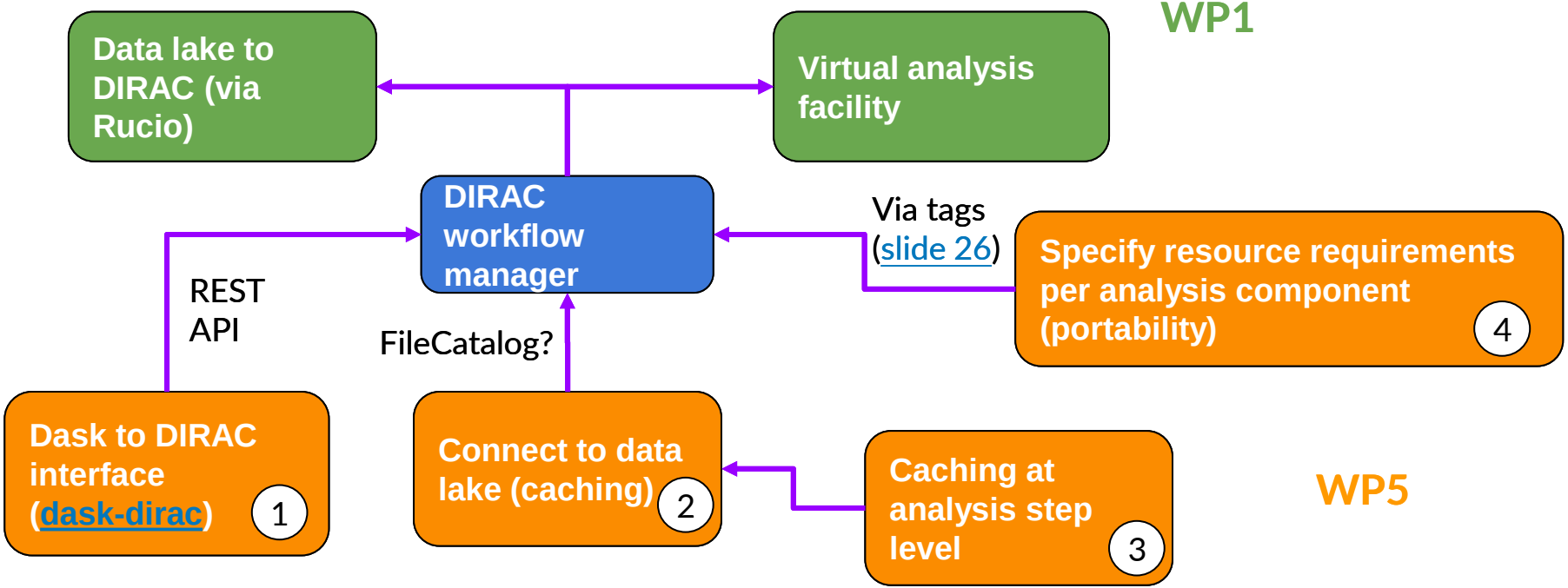
DIRAC

Data lake

WP5 in a nutshell:
Run analysis workloads optimally** on distributed (GridPP) resources

** balanced between user-experience and computing efficiency

WP5: Roadmap



More detailed slides

WP5: Roadmap

1. **Dask to DIRAC interface (dask-dirac)**
   - Add extension to dask
   - Dask is able to parallelize any python code
2. **Connect to data lake (caching)**
   - Add the ability to save output after dask instance has closed
3. **Caching at analysis step level**
   - Avoid having to re-run analysis steps
4. **Specify resource requirements per analysis component (portability)**
   - E.g. Let some stages run on GPUs

See talk by Luke Krezcko for some future planning

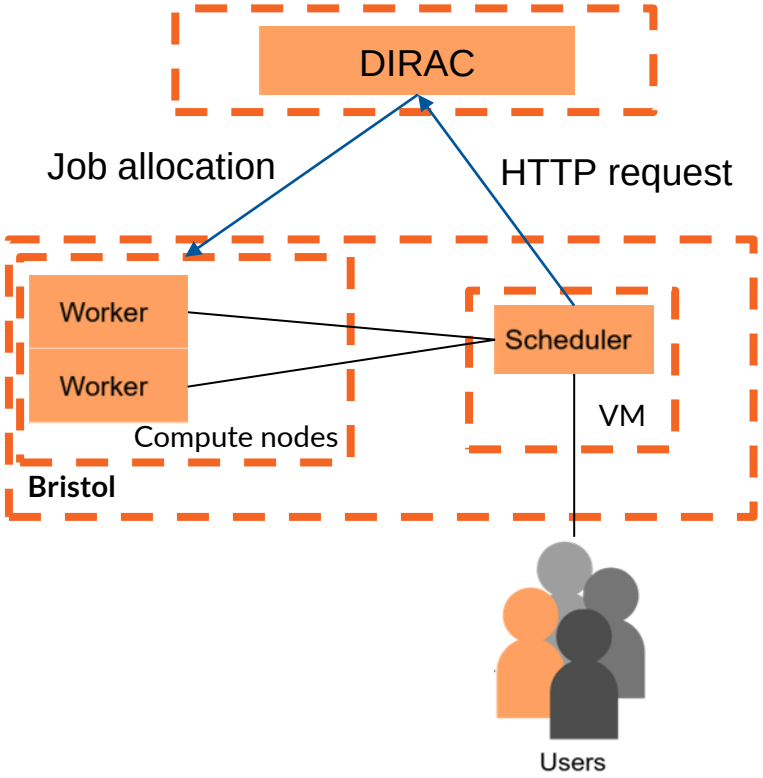
What's happened since the last workshop

Where did we leave things

Dask to DIRAC interface (dask-dirac)

- New VM at Bristol to host a dask scheduler
- Created an extension to dask to deploy workers via Dirac (via certificate server)
- Should now be able to run AGCs

```
elif af == "DIRAC":
    from dask_dirac import DiracCluster
    from dask.distributed import Client

    cluster = DiracCluster(cores=8,
                memory="24GB",
                scheduler_options={"port": 8786},
                #dirac_site="LCG.UKI-SOUTHGRID-RALPP.uk",
                dirac_site="LCG.UKI-SOUTHGRID-BRIS-HEP.uk",
                cert_path="/users/ak18773/SWIFT_HEP/dev_dirac/diracos/etc/grid-security/certificates",
                owner_group="gridpp_user",
                user_proxy="/tmp/x509up_u397871",
                submission_url="https://diracdev.grid.hep.ph.ic.ac.uk:8444",
            )

    cluster.scale(jobs=2)
```



Running AGCs

- In order to test what we've done, we planned on using IRIS-HEP AGCs (CMS ttbar)
- Already uses dask so is a simple edit to get to work with our 'cluster'
- Naturally things didn't go that smoothly
  - python version pickling
  - My pains are detailed in monthly update meetings

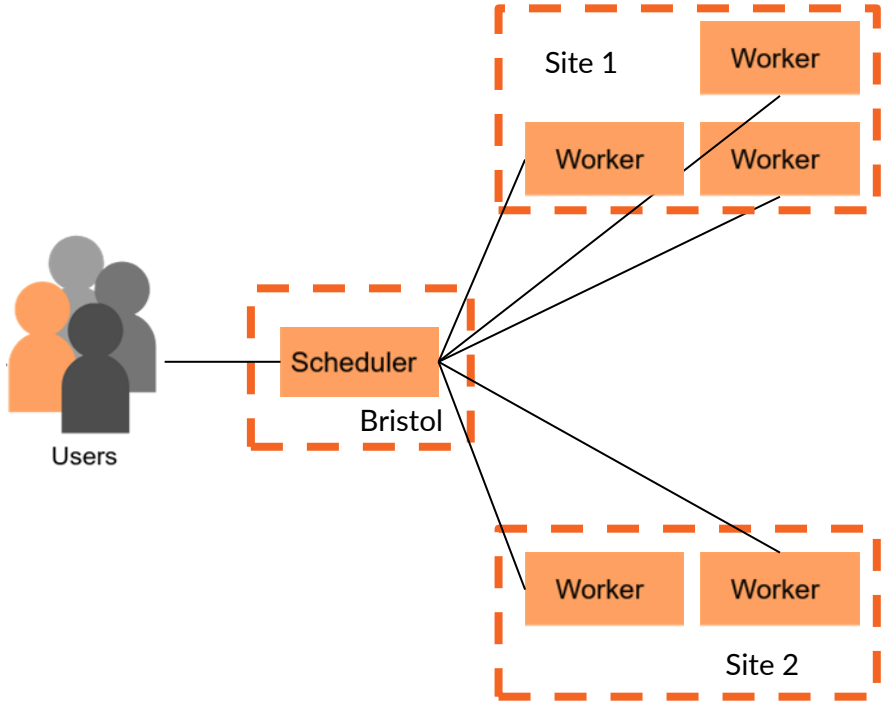
```
elif af == "DIRAC":
    from dask_dirac import DiracCluster
    from dask.distributed import Client

    cluster = DiracCluster(cores=8,
                memory="24GB",
                scheduler_options={"port": 8786},
                #dirac_site="LCG.UKI-SOUTHGRID-RALPP.uk",
                dirac_site="LCG.UKI-SOUTHGRID-BRIS-HEP.uk",
                cert_path="/users/ak18773/SWIFT_HEP/dev_dirac/diracos/etc/grid-security/certificates",
                owner_group="gridpp_user",
                user_proxy="/tmp/x509up_u397871",
                submission_url="https://diracdev.grid.hep.ph.ic.ac.uk:8444",
            )

    cluster.scale(jobs=2)
```

Going beyond a single site

Dask to DIRAC interface (dask-dirac)

- Workers and Scheduler needed to be able to talk to each other -> firewalls stopped this at other sites

- Essentially to get around it, we'd need to write a new connection protocol
  - Explored RabbitMQ, SSH forwarding, dask-gateway
  - See dask forum discussion on all of these



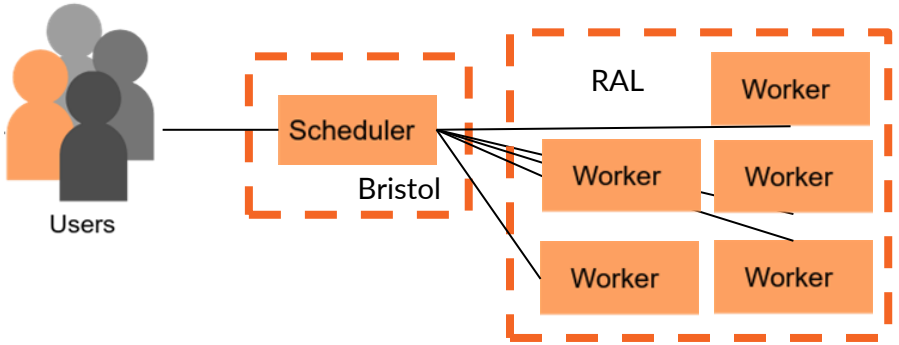
Going beyond a single site

Dask to DIRAC interface (dask-dirac)

- This hampers plans to use Brunel site for testing
- Luckily, RAL has nodes which have open ports

```
if site == "LCG.UKI-SOUTHGRID-RALPP.uk":
    return " --worker-port 50000:52000"
```

- Can look at some basic benchmarking



Going beyond a single site – basic benchmarking

Dask to DIRAC interface (dask-dirac)

- This hampers plans to use Brunel site for testing
- Luckily, RAL has nodes which have open ports
- Can now do basic benchmarking

setting	number of files	total size	number of events
1	9	22.9 GB	10,455,719
2	18	42.8 GB	19,497,435
5	43	105 GB	47,996,231
10	79	200 GB	90,546,458
20	140	359 GB	163,123,242
50	255	631 GB	297,247,463
100	395	960 GB	470,397,795
200	595	1.40 TB	705,273,291
-1	787	1.78 TB	940,160,174

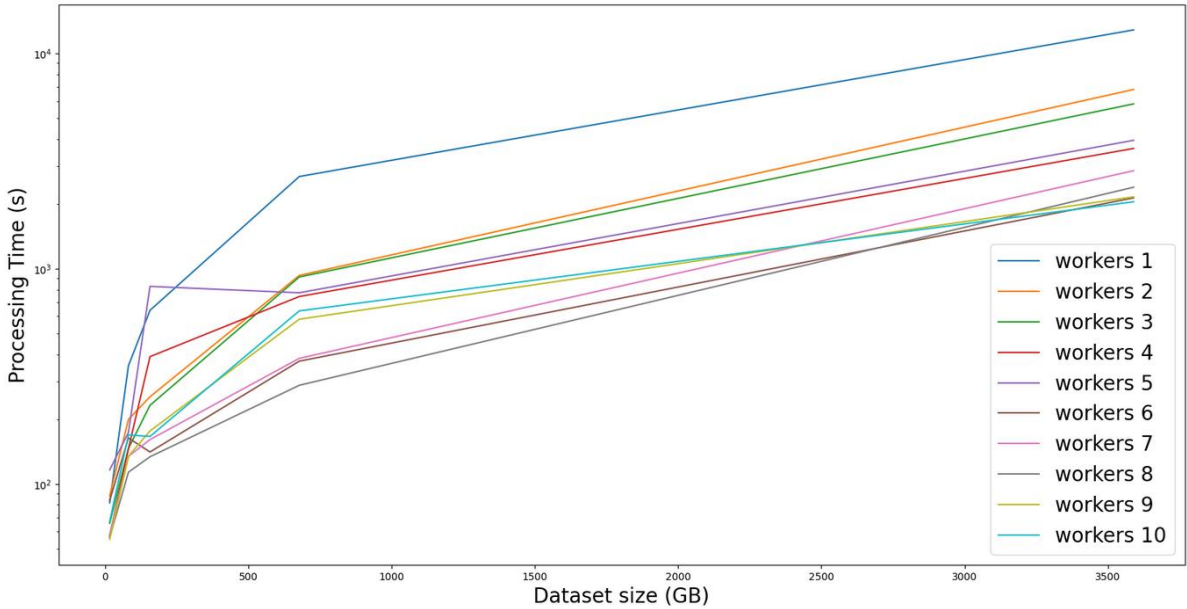
Going beyond a single site – basic benchmarking

Dask to DIRAC interface (dask-dirac)

In plot
- Each line is a different number of workers
- Lower = faster to process

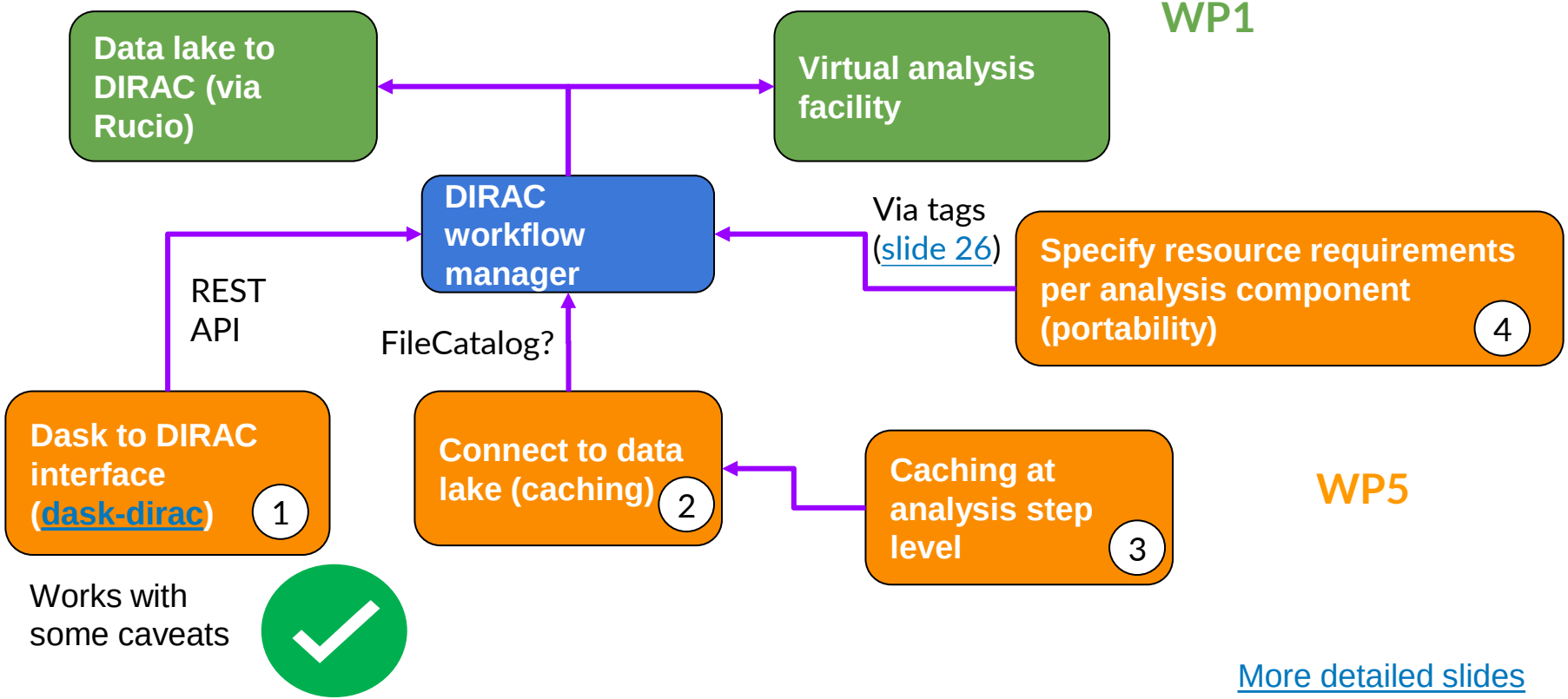
Generally, things behave as expected
- More workers = takes less time
- Some cases where more worker make things take longer

- All workers running at RAL T2

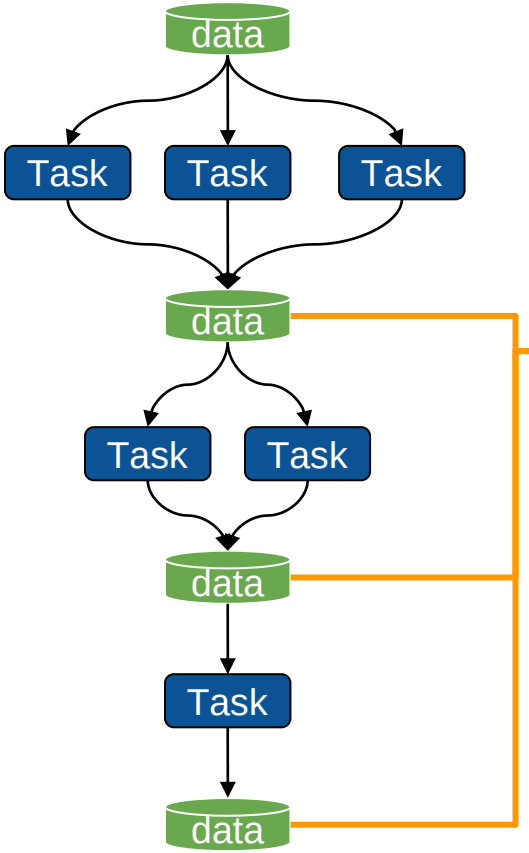


More slides on this: monthly update September 2023

WP5: Roadmap


WP1
Data lake to DIRAC (via Rucio)
Virtual analysis facility
DIRAC workflow manager
Via tags (slide 26)
Specify resource requirements per analysis component (portability)
4
REST API
FileCatalog?
Dask to DIRAC interface (dask-dirac)
1
Connect to data lake (caching)
2
Caching at analysis step level
3
WP5
Works with some caveats
More detailed slides

Interacting with the data lake



Connect to data lake (caching)

WP5

WP1

Analysis **step** output

caching

DIRAC

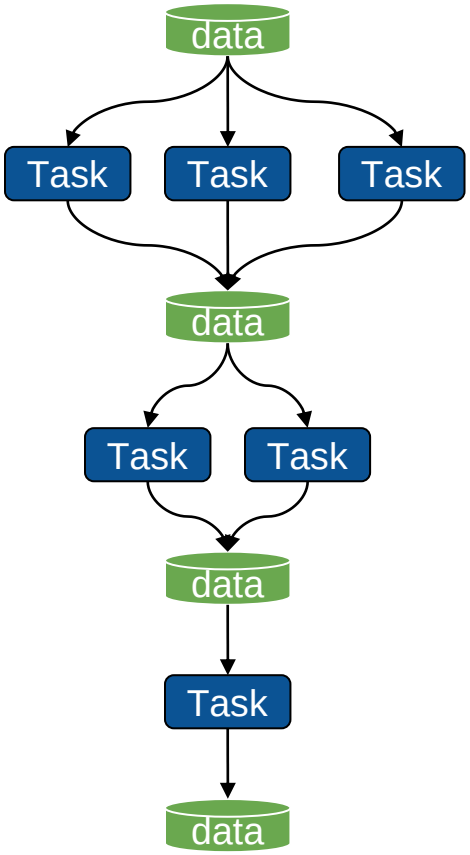
Data lake

Plan;

- Add HTTP filecatalog interaction functionality to dask_dirac

- Test moving data from workers to data lake without going to scheduler node first

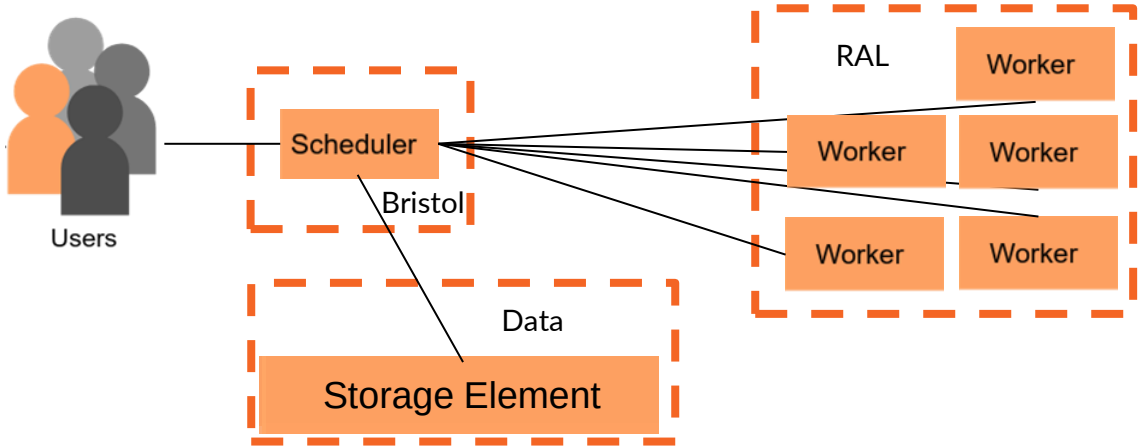
Interacting with the data lake



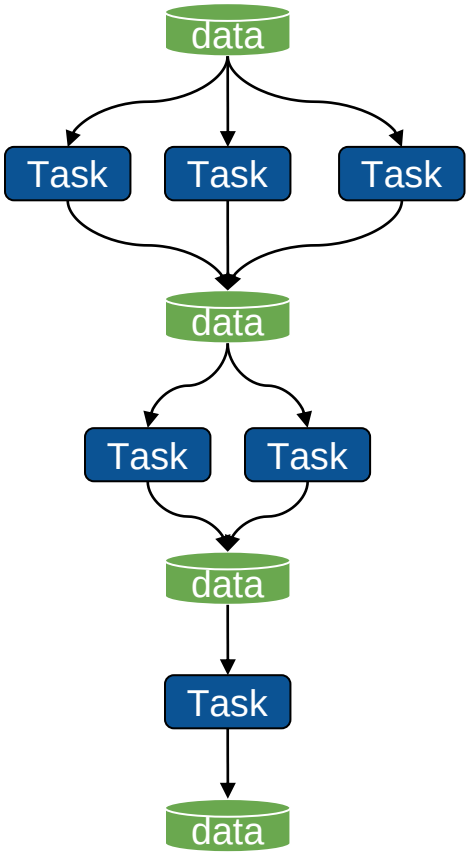
Connect to data lake (caching)

Status;

- Able to do all file manipulation easily except adding files
- Currently using gfal for file adding (add file with gfal, then register file via HTTP)



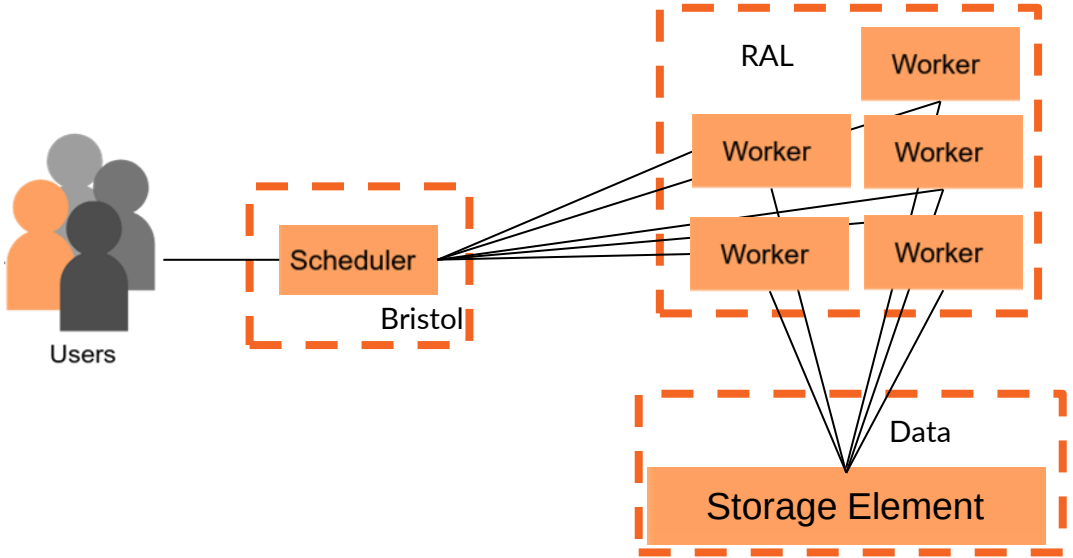
Interacting with the data lake



Connect to data lake (caching)

Going forward;

- Add dask worker plugin so data does not have to go via scheduler into storage
- Potential for RUCIO to handle this file manipulation (in talks with Tim Noble about this)



Other things we've been tackling in WP5 and the plan going forward

Making things reproducible

DIRAC certificate

- Get a conda env following: https://github.com/DIRACGrid/DIRACOS2
- Then run `dirac-proxy-init –g gridpp_user`

Get dask-dirac

- From pypi `pip install dask-dirac`
- (For development, use github)

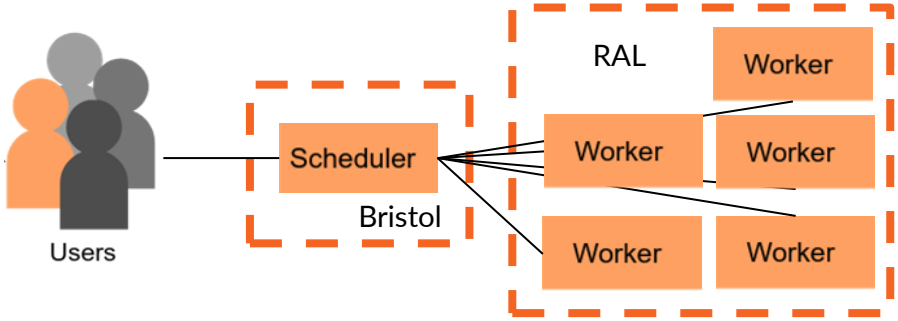
Get AGC branch with edits

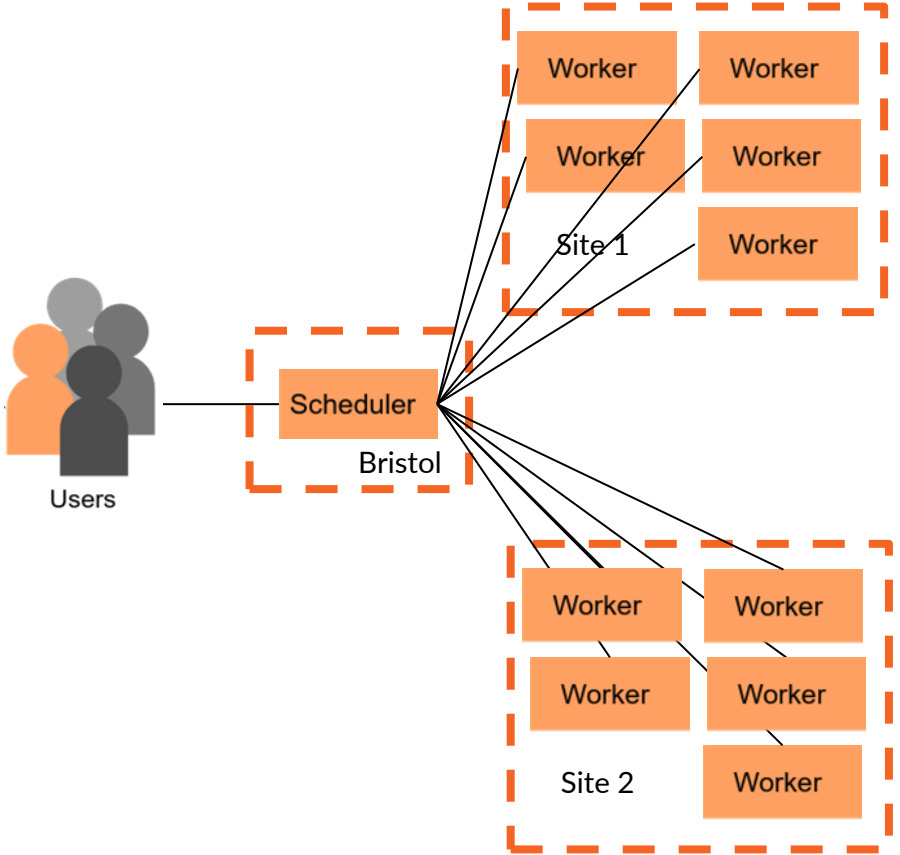
- Using AGC from SWIFT-HEP fork
- `git clone -b se_daskdirac git@github.com:SWIFT-HEP/analysis-grand-challenge.git`
- Edit config to use users certificate

Run CMS analysis

Documentations in dask-dirac is in progress

Multi-site workers





- Currently can submit to any site…
- But only ONE site for a given dask cluster
- Improving JDL templating to allow for multiple sites to be submitted to
- This would still be limited by worker node port openness

Trial with Rdataframe AGC

In addition to the ‘standard’ AGC, there is also a ROOT Rdataframe implementation
https://github.com/root-project/analysis-grand-challenge

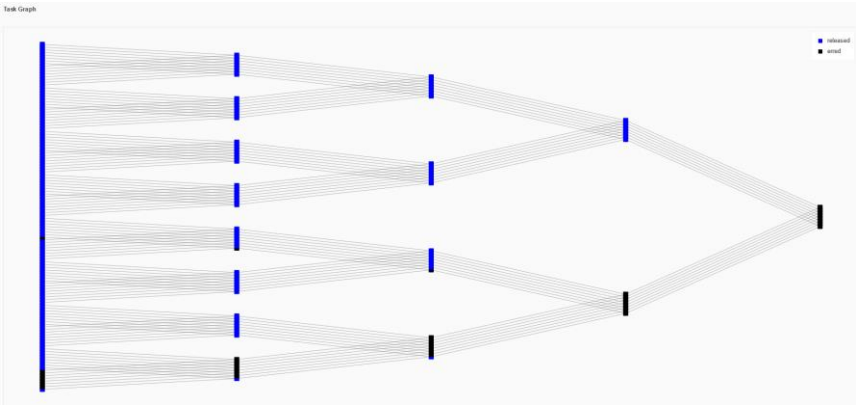
Key difference;
- Requires ROOT
- And therefore, a different worker container

Status;
- Doesn’t work at the moment

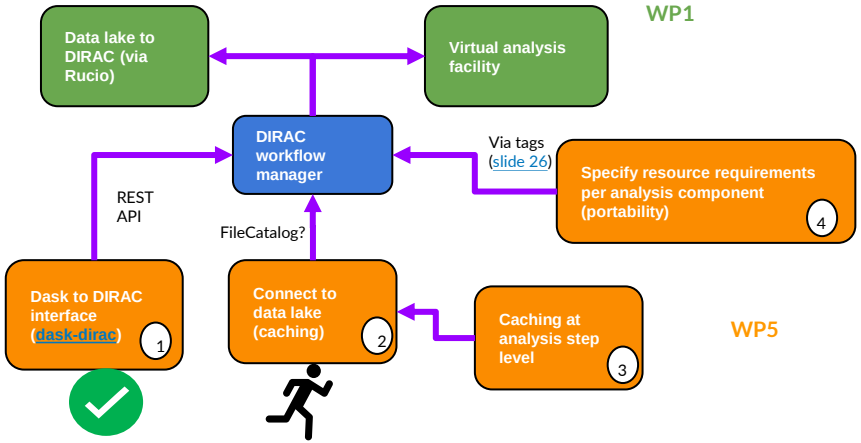
Exception information

Exception: `AttributeError("'RDatasetSpec' object has no attribute 'AddGroup'")`

▸ **Traceback**



Summary and plan going forward



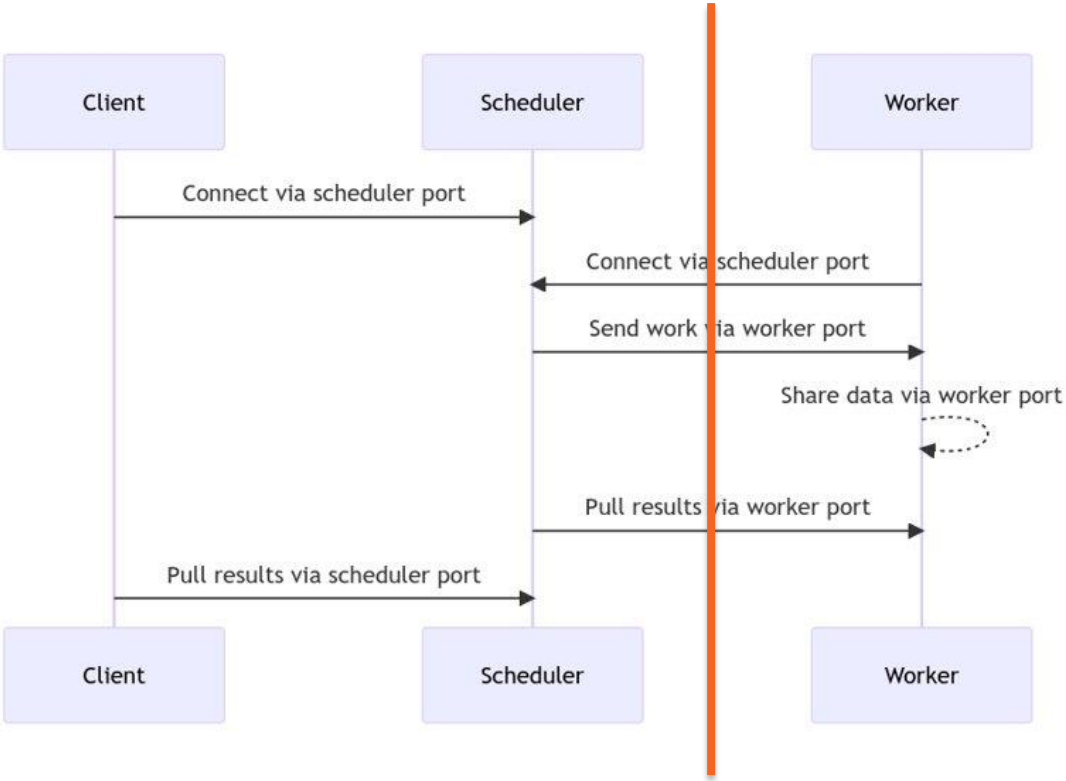
- We are getting closer achieving the goals of WP5 Phase 1
- We can reliably run CMS AGC
- There is functionality for file manipulation but is painful, possibly solution in RUCIO
- Implementation of multi-site worker deployment

University of BRISTOL

Questions?

BACKUP

The network explained



So where is the problem?

Imagine network boundary between scheduler and workers

Scheduler port is accessible from workers

Worker port is **ONLY** accessible to scheduler if connection is recycled (part of **ESTABLISHED** --> firewall OK)

Default Dask operation: this can happen at **RANDOM** (most likely for small # of workers)

The network solution

- We know connections can be recycled and bypass firewall if they are part of an ESTABLISHED connection
- We also know of a working solution in our field: The HTCondor Connection Broker
  - Workers, schedulers, etc connect to a SHARED_PORT
  - As long as SHARED_PORT is open in firewall on a node accessible to both scheduler and workers --> connection can be established
- Most simple solution: Can the Dask Connection proxy be rewritten to hold worker connections?
  - What are the downsides for 100-1000 worker nodes?