ECHEP Analysis Area Update

2020.04.20

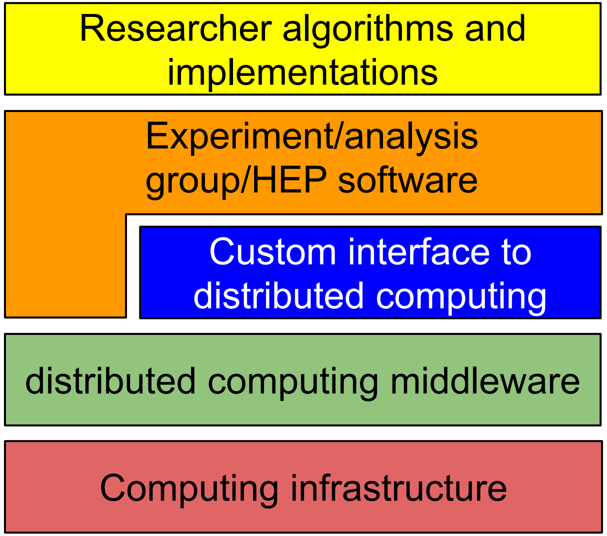
Eduardo Rodrigues, Luke Kreczko

The HEP analysis stack

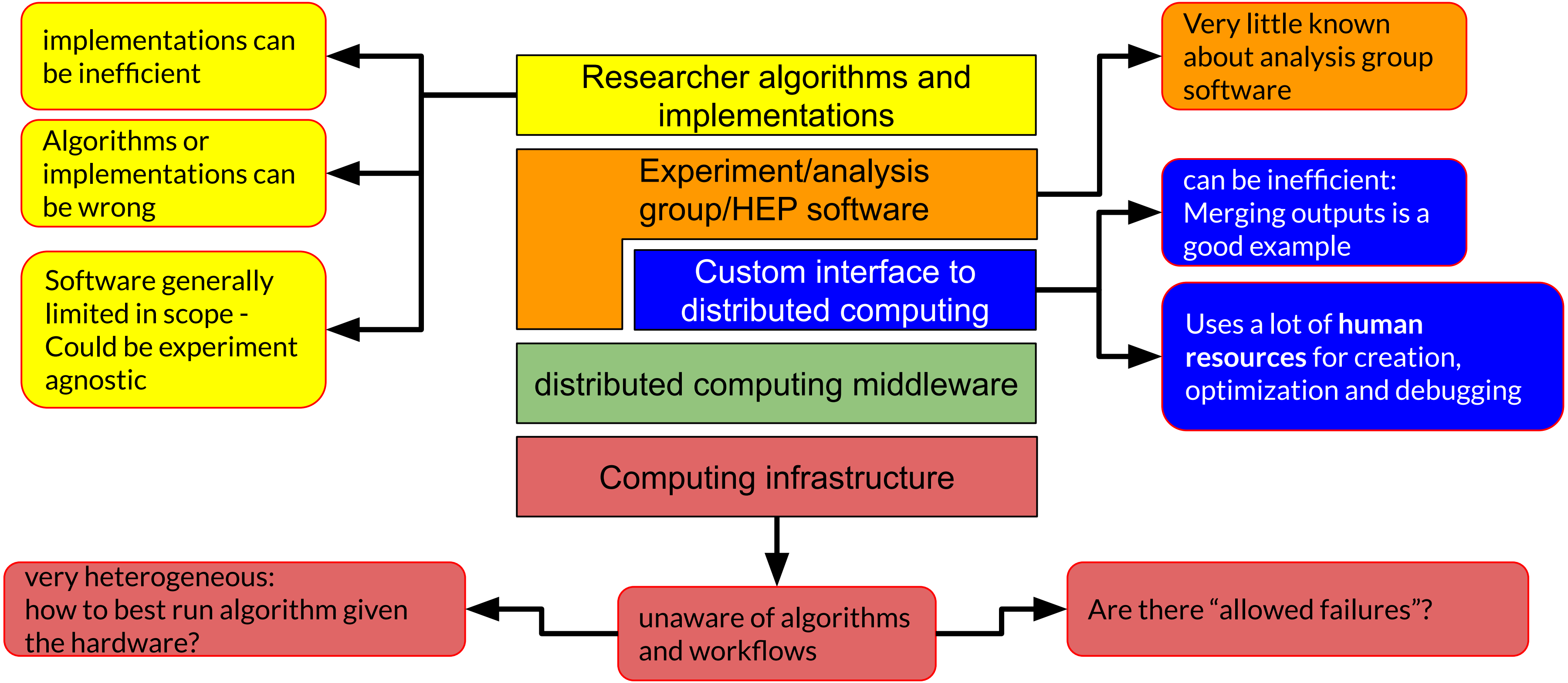
HEP Analysis Stack

As any other HEP software area, analysis software and related issues cannot be viewed in isolation

- Researchers develop/reuse algorithms and implement them to their best programming ability
- Implementations might be based on experiment frameworks, analysis group specific software or general HEP tools
- Access to distributed computing might involve custom interfaces (e.g. researcher written bash or python scripts)
- Access to computing controlled via middleware
- Computing infrastructure is distributed and very heterogeneous


Researcher algorithms and implementations
Experiment/analysis group/HEP software
Custom interface to distributed computing
distributed computing middleware
Computing infrastructure

HĒP Analysis Stack - Many possibilities for inefficiencies or failures



Researcher algorithms and implementations

implementations can be inefficient

Algorithms or implementations can be wrong

Software generally limited in scope - Could be experiment agnostic

Experiment/analysis group/HEP software

Very little known about analysis group software

Custom interface to distributed computing

can be inefficient: Merging outputs is a good example

Uses a lot of **human resources** for creation, optimization and debugging

distributed computing middleware

Computing infrastructure

unaware of algorithms and workflows

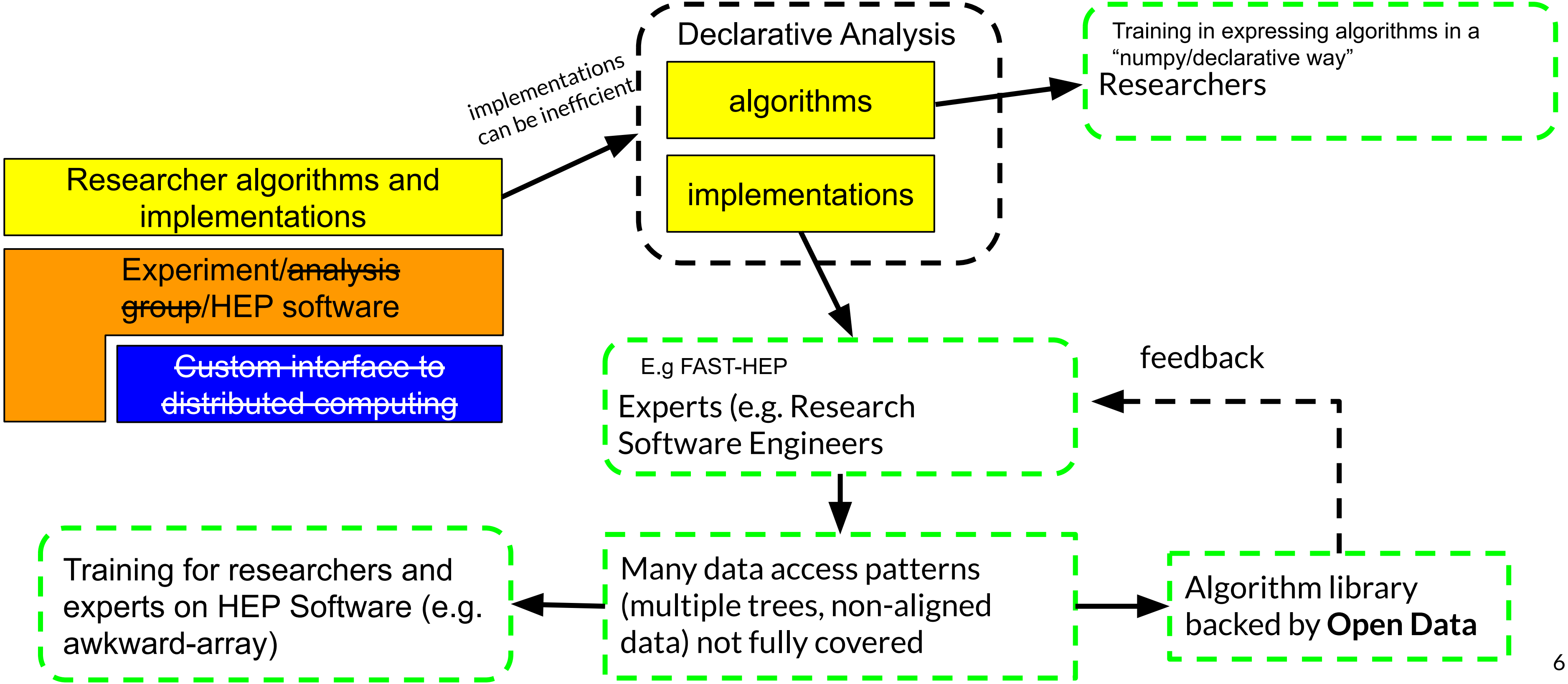
very heterogeneous: how to best run algorithm given the hardware?

Are there “allowed failures”?

How can we address the issues?

(and where does ECHEP fit it)

Possible ECHEP working items (in green)



Declarative Analysis

algorithms

implementations

implementations can be inefficient

Training in expressing algorithms in a “numpy/declarative way”

Researchers

Researcher algorithms and implementations

Experiment/~~analysis group~~/HEP software

~~Custom interface to distributed computing~~

E.g FAST-HEP

Experts (e.g. Research Software Engineers

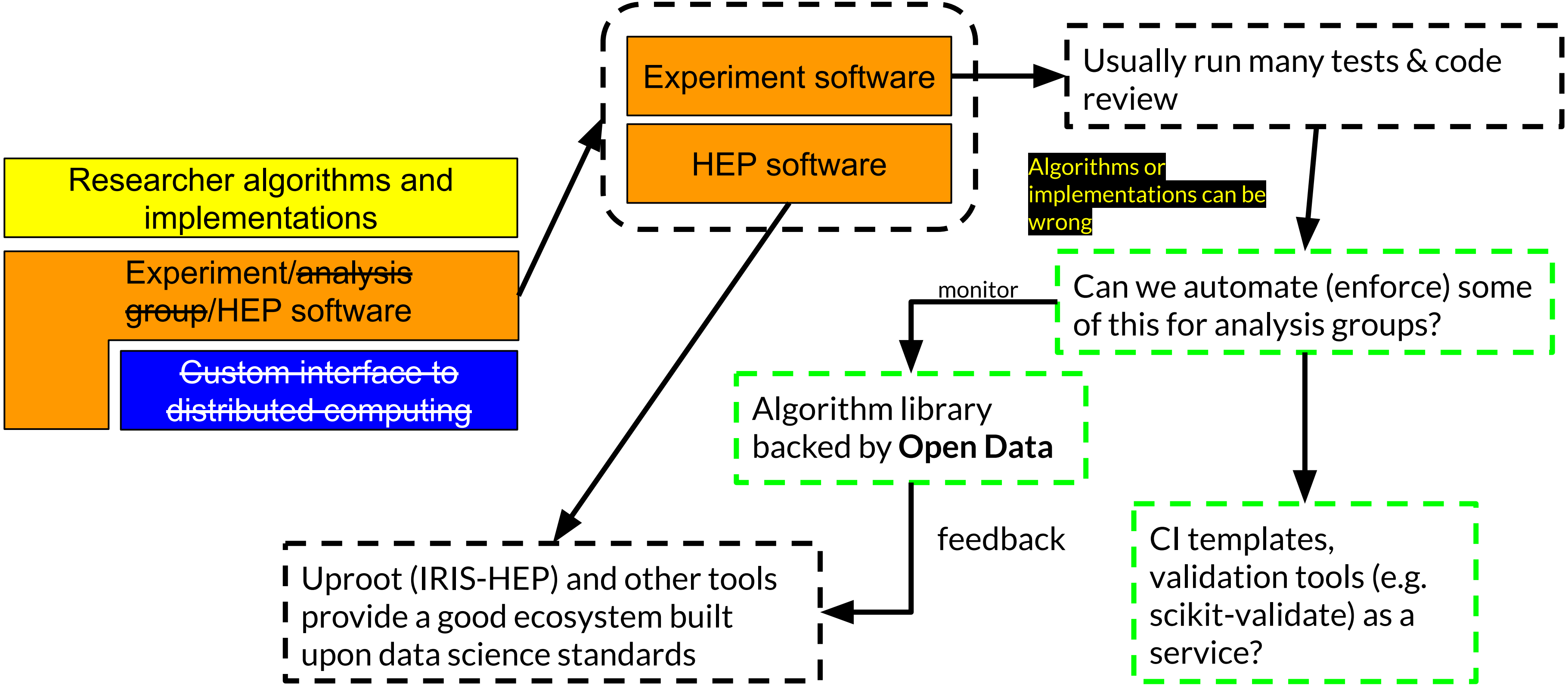
feedback

Training for researchers and experts on HEP Software (e.g. awkward-array)

Many data access patterns (multiple trees, non-aligned data) not fully covered

Algorithm library backed by **Open Data**

Possible ECHEP working items (in green)



Researcher algorithms and implementations

Experiment/~~analysis group~~/HEP software

~~Custom interface to distributed computing~~

Experiment software

HEP software

Usually run many tests & code review

Algorithms or implementations can be wrong

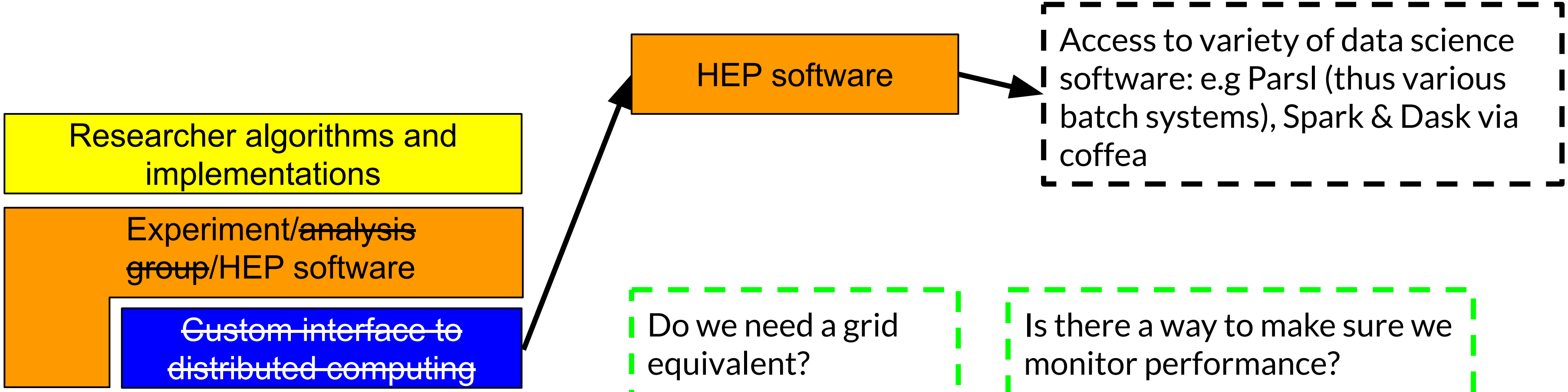
Can we automate (enforce) some of this for analysis groups?

Algorithm library backed by **Open Data**

CI templates, validation tools (e.g. scikit-validate) as a service?

Uproot (IRIS-HEP) and other tools provide a good ecosystem built upon data science standards

Possible ECHEP working items (in green)



Researcher algorithms and implementations

Experiment/~~analysis group~~/HEP software

~~Custom interface to distributed computing~~

HEP software

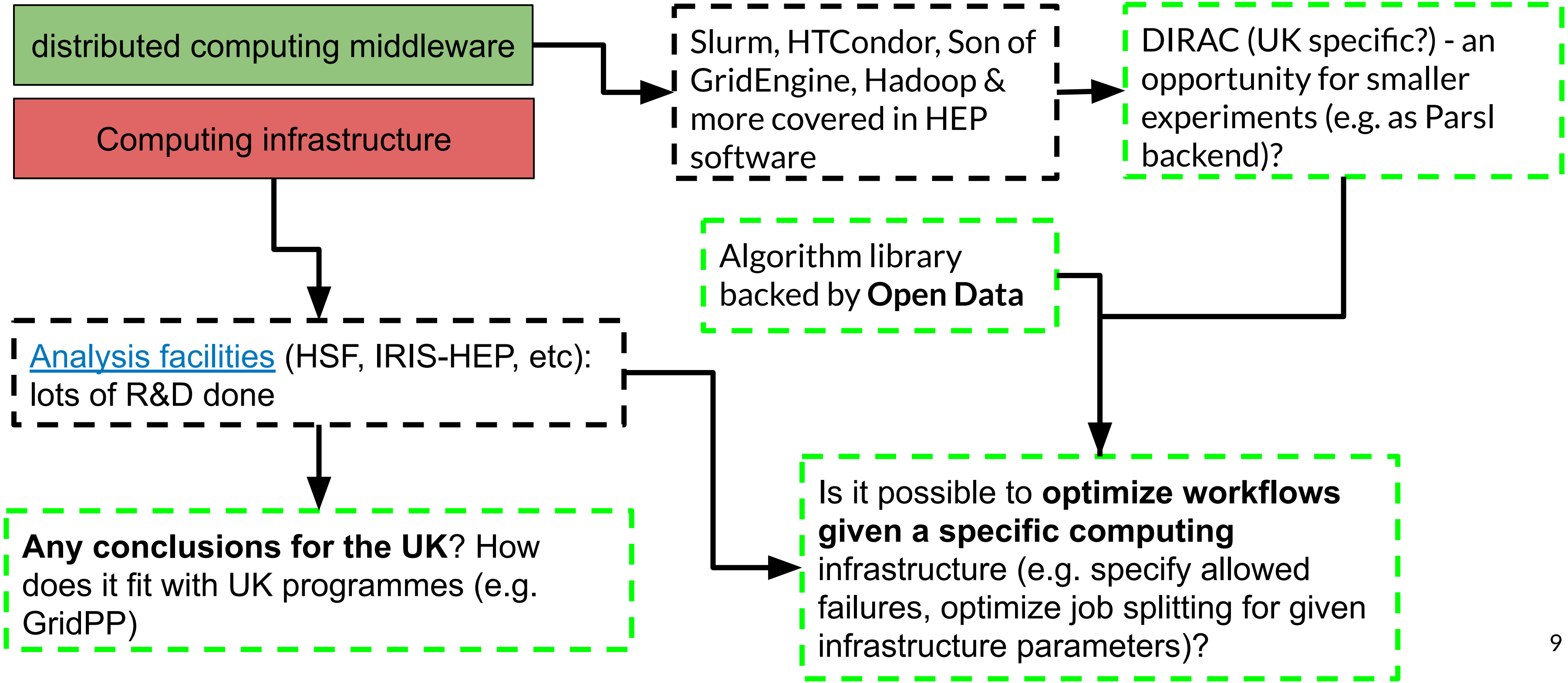
Access to variety of data science software: e.g Parsl (thus various batch systems), Spark & Dask via coffea

Do we need a grid equivalent?

Is there a way to make sure we monitor performance?

Can we construct a compute & workflow graph to optimize for architecture/infrastructure?

Possible ECHEP working items (in green)



distributed computing middleware

Computing infrastructure

Slurm, HTCondor, Son of GridEngine, Hadoop & more covered in HEP software

DIRAC (UK specific?) - an opportunity for smaller experiments (e.g. as Parsl backend)?

Algorithm library backed by **Open Data**

Analysis facilities (HSF, IRIS-HEP, etc): lots of R&D done

Any conclusions for the UK? How does it fit with UK programmes (e.g. GridPP)

Is it possible to **optimize workflows given a specific computing** infrastructure (e.g. specify allowed failures, optimize job splitting for given infrastructure parameters)?

Training needs

Trends

- Python at least as popular as C++, if not more already now in 2020 (cf. CMS and LHCb surveys)
- Declarative approaches as a means to improve compute efficiency (optimisations can be done behind the scenes, professionally)
- More query-style and interactive analysis (largely via notebooks)
- Machine learning and AI permeate everything

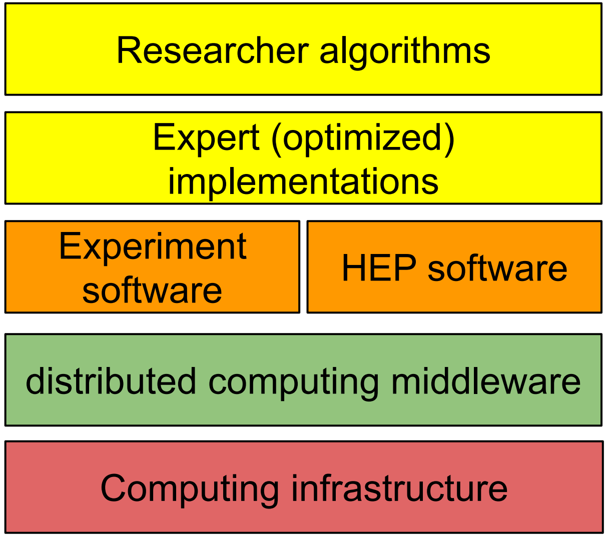
Ongoing efforts

- HSF PyHEP workshops
- HSF/IRIS-HEP training activities such as the HEP Software Carpentry (SC) workshops (1st event)

What can the UK do?

- Certainly tag along and contribute and/or drive some of these efforts
  - The case already in certain cases - PyHEP organisation, SC organisation and tutors
- Organise UK versions in the future - the community is large enough for that
  - Need to be cross-experiments, clearly
- Nobody seems to be organising beginners-type ML training events (IML more for experts, exception: MLHEP). UK community could organise some sort of AI/ML SC type of workshops and/or engage strongly with MLHEP

Analysis stack summary



Researcher algorithms

Expert (optimized) implementations

Experiment software

HEP software

distributed computing middleware

Computing infrastructure

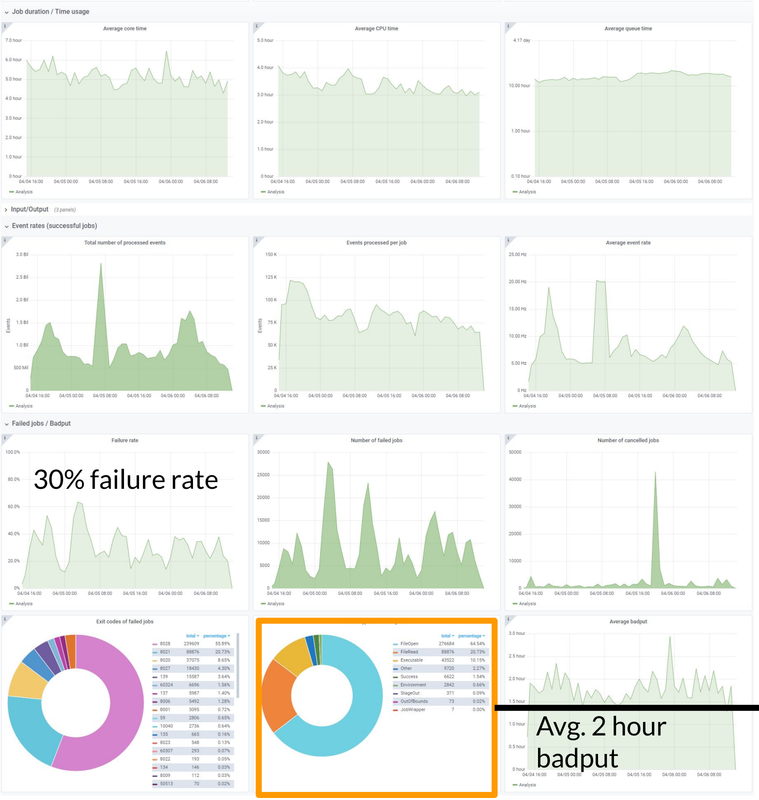
If the aim is to maximize resource savings AND physics outputs, we need to look at the whole stack

- Benchmarks (Algorithm library backed by **Open Data**) is a crucial step to synchronize requirements throughout the stack*
- Opportunity to improve data analysis methodology while building upon existing efforts (HSF, IRIS-HEP)
- Communication pathways might be needed (e.g. researchers/experts <-> computing infrastructure)
- Training at both beginner and export levels is necessary to make any kind of transition

*Can also help to reduce unintended side effects across HEP software (e.g. ROOT 6.20 nested-namespace slowdown)

Backup slides

CMS analysis job failures (30% failure rate)

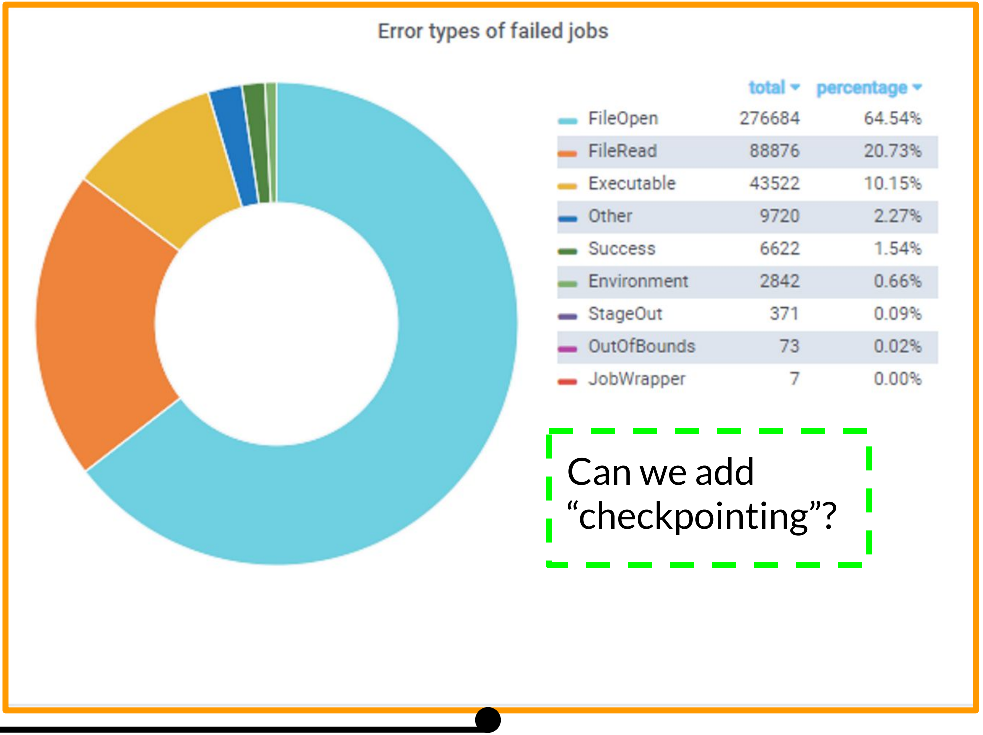


30% failure rate

Exit codes of failed jobs

	total	percentage
8028	239609	55.89%
8021	88876	20.73%
8020	37075	8.65%
8027	18430	4.30%
139	15587	3.64%
60324	6696	1.56%
137	5987	1.40%
8006	5492	1.28%
8001	3095	0.72%
59	2806	0.65%
10040	2736	0.64%
135	665	0.16%
8023	548	0.13%
60307	293	0.07%
8022	193	0.05%
134	146	0.03%
8009	112	0.03%
50513	70	0.02%

Avg. 2 hour badput



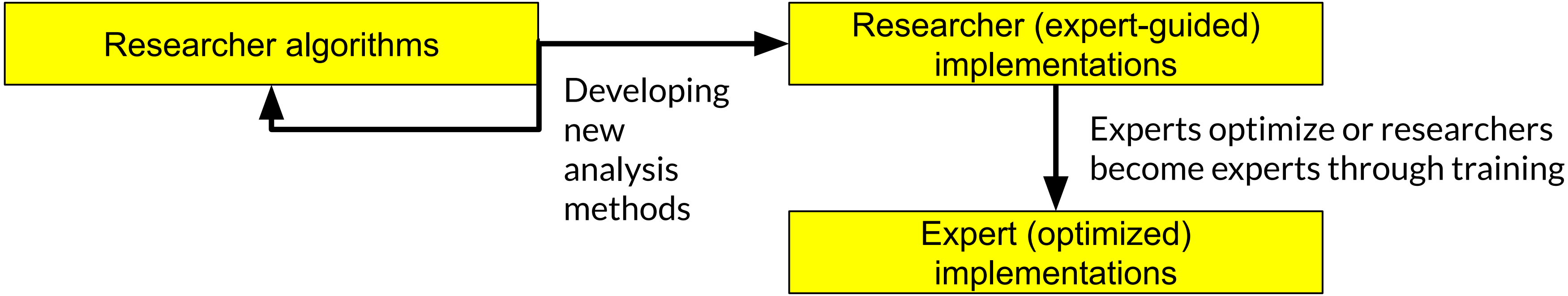
Error types of failed jobs

	total	percentage
FileOpen	276684	64.54%
FileRead	88876	20.73%
Executable	43522	10.15%
Other	9720	2.27%
Success	6622	1.54%
Environment	2842	0.66%
StageOut	371	0.09%
OutOfBounds	73	0.02%
JobWrapper	7	0.00%

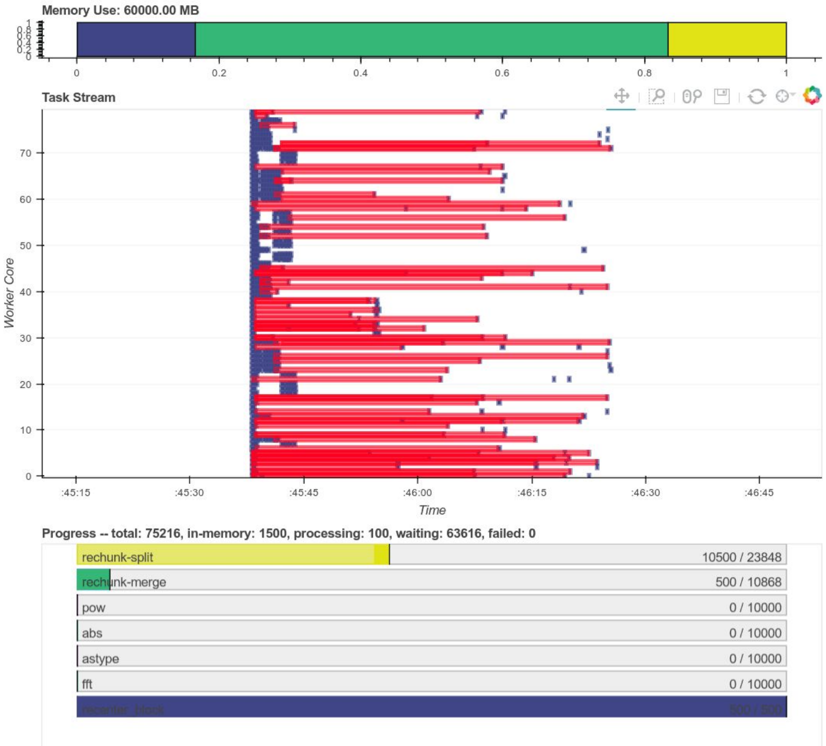
Can we add “checkpointing”?

If we do not teach researchers how to code, won’t we lack experts?

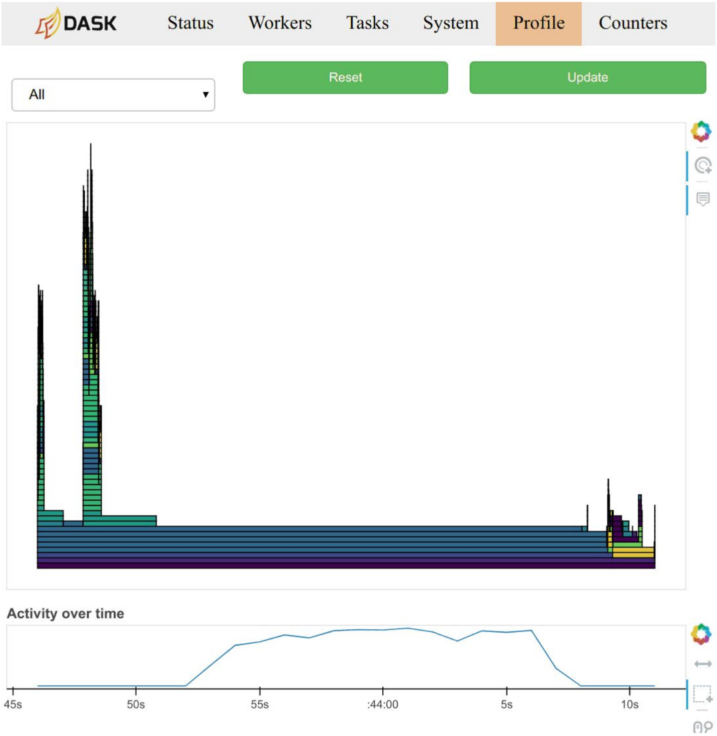
Easy start, natural progression - training at undergrad, postgrad, research associate level is always crucial


Researcher algorithms
Developing new analysis methods
Researcher (expert-guided) implementations
Experts optimize or researchers become experts through training
Expert (optimized) implementations

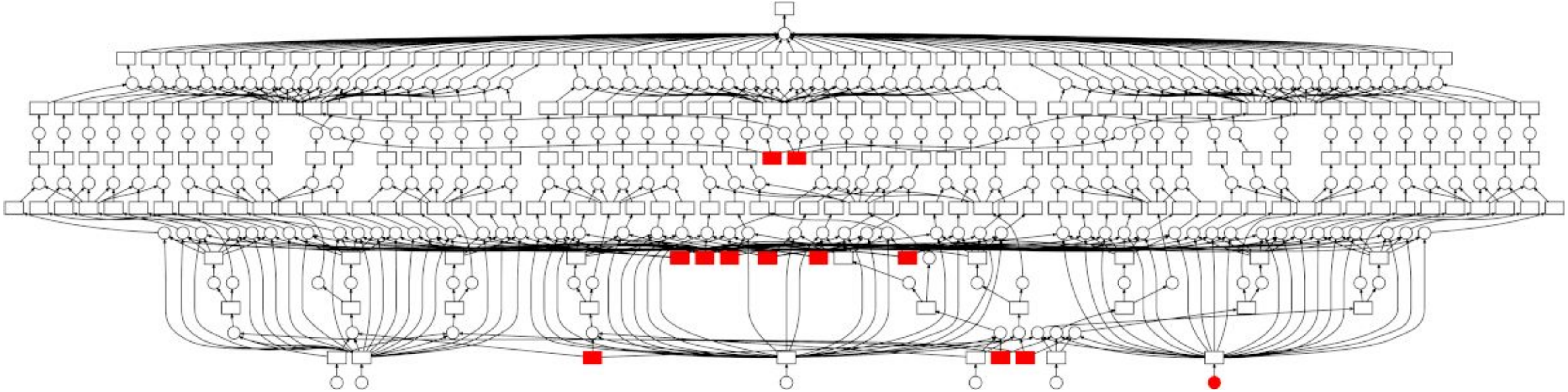
Why switching to Data science tools can be good



```
Progress -- total: 75216, in-memory: 1500, processing: 100, waiting: 63616, failed: 0
rechunk-split                                10500 / 23848
rechunk-merge                                  500 / 10868
pow                                              0 / 10000
abs                                              0 / 10000
astype                                           0 / 10000
fft                                              0 / 10000
recenter_block                                 500 / 500
```



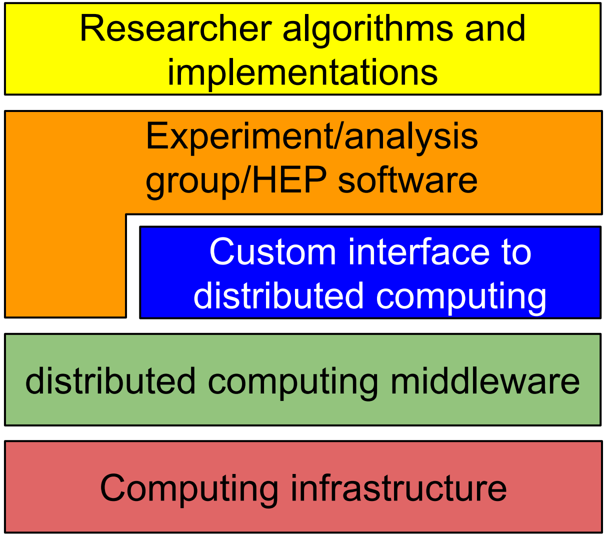
Computations/workflow graph example



HEP Analysis Stack

Many possibilities for inefficiencies or failures

- Algorithms or implementations can be wrong
- Implementations and custom interfaces to distributed computing can be inefficient
  - Merging outputs is a good example
  - Uses a lot of **human resources** for creation, optimization and debugging
- Software generally limited in scope
  - Could be experiment agnostic
- (very heterogeneous) Computing infrastructure is unaware of algorithms and workflows


Researcher algorithms and implementations
Experiment/analysis group/HEP software
Custom interface to distributed computing
distributed computing middleware
Computing infrastructure