Swift-HEP
logo placeholder

Swift-HEP/ExcaliburHEP workshop
15th January 2021

# Analysis in ECHEP and towards Swift-HEP

Eduardo Rodrigues & Luke Kreczko
UNIVERSITY OF LIVERPOOL
University of BRISTOL

# Swift-HEP got funded - what's next?

- Quick recap of ECHEP → Swift-HEP
- Analysis Work Package timeline
- Collaboration with other WPs
- Summary

# A quick recap

# A quick recap: ECHEP - the challenges

Extensive summary available in the ECHEP Report. Highlights below

- Data analysis in HEP experiments is a **diverse and large-scale workflow landscape**
- Analysis code performance can differ by orders of magnitude depending on **algorithm complexity and implementation details***
- Failure rates at the experiments around 20% (**wasting O(MGBP)** of cloud credit equivalent per experiment)
- Monitoring does not fully capture **analysis repetition**: resubmitting jobs, research group rotas, etc - potential for improvement

# UK expertise - data analysis focus

- The UK has **expertise in data analysis and large computing infrastructure**
- Impact on the international scene through **involvements with the HSF**, **Scikit-HEP and FAST-HEP** projects
    - The UK currently hosts two conveners of HSF’s PyHEP working group
- Contributions to many international HEP packages by UK members
- Building up new generation of experts with CDTs for data intensive science

# Future challenges

- **Complexity and scale** of data analysis workflows are **expected to increase**
  - More data, new frontiers, more complex experiments
- Computing infrastructure is expected **NOT to increase** proportionally
  - Iterative nature of data analysis and repetition will not bode well
- Distributed computing resources are **highly heterogeneous**
  - Reproducibility of research can be difficult
- Specialized **hardware accelerators** (GPU, FPGA) are sparse - but so is the expertise to use them for data analysis

# Swift-HEP analysis work package

# Possible solutions to future challenges

A lot of effort on the global stage:

- Software solutions targeting specific areas
    - from PyHEP: uproot, awkward-array, coffea - all in a Python big data ecosystem
- New analysis interfaces (e.g. CERN's SWAN)
- Dedicated analysis centres (e.g. CMS Analysis Facility)
- Data analysis platforms for reproducibility and speed (reana, ServiceX)

All aimed to improve performance, usability and reproducibility

There is no magic bullet - but there is a methodology to move forward

# Divide and conquer: Declarative approach

Declarative approach: researchers specify the **algorithms as mathematical formulas** instead of their implementation (e.g, ROOT's RDataFrame)

- **Reduces amount of code** (thus bugs) to be written by researchers
- Enables experts to **improve implementation over time**
- Underlying **software** solution (i.e. the engine) can even be **completely exchanged** without any changes to analysis description

Caveat: requires a **culture change** within the HEP community

Benefit: any improvement has **impact on ALL analyses** going this route

# Swift-HEP analysis work package (WP5)

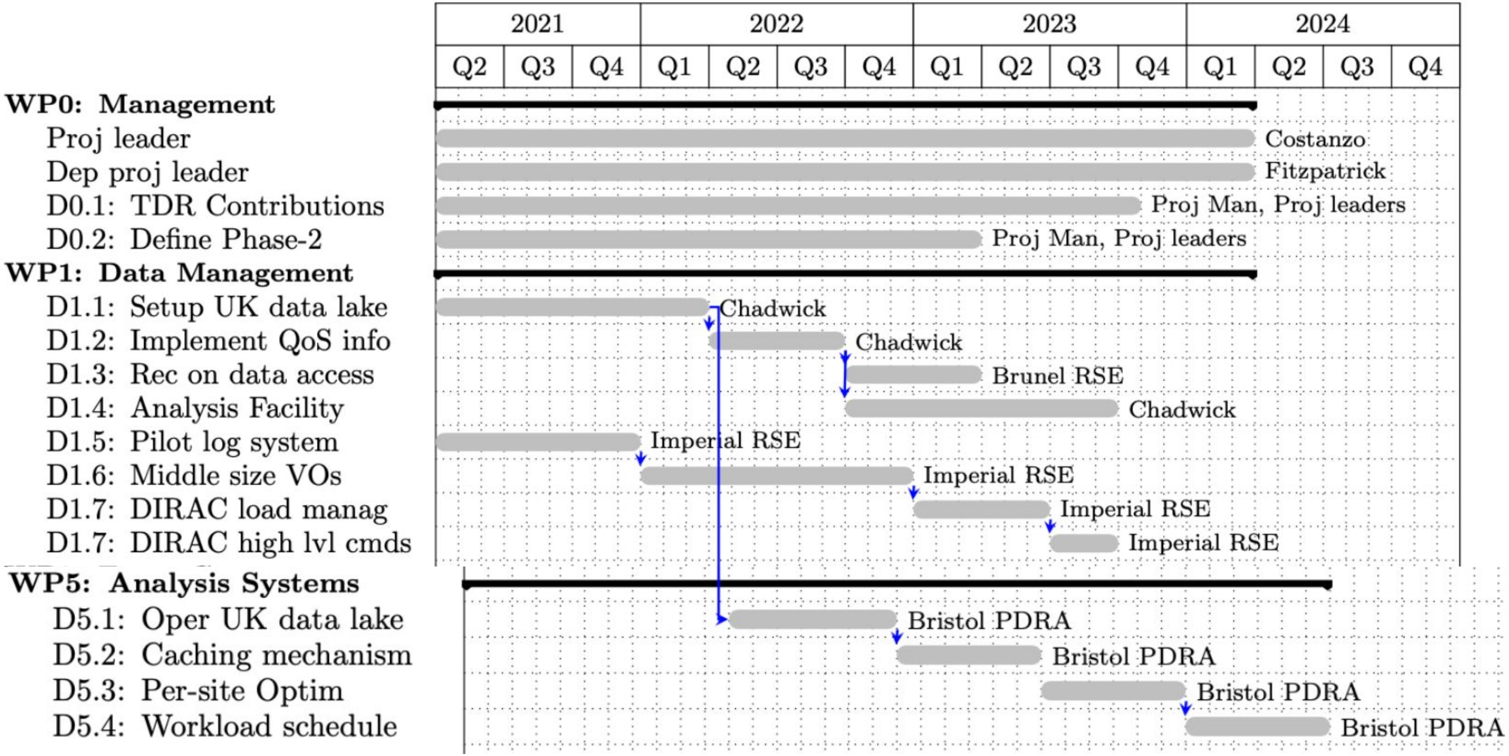
Declarative analysis is a **good starting point** for future improvements, but all deliverables will be usable without it.

WP5 is **tightly coupled to WP1**: “Intelligent data and workflow management”

Proposed work focuses on two aspects of data analysis: **repetition and performance**.

For repetition we will work on **caching analysis steps** and for performance we will look at **per-site optimization of workloads**

# Swift-HEP Gantt Chart

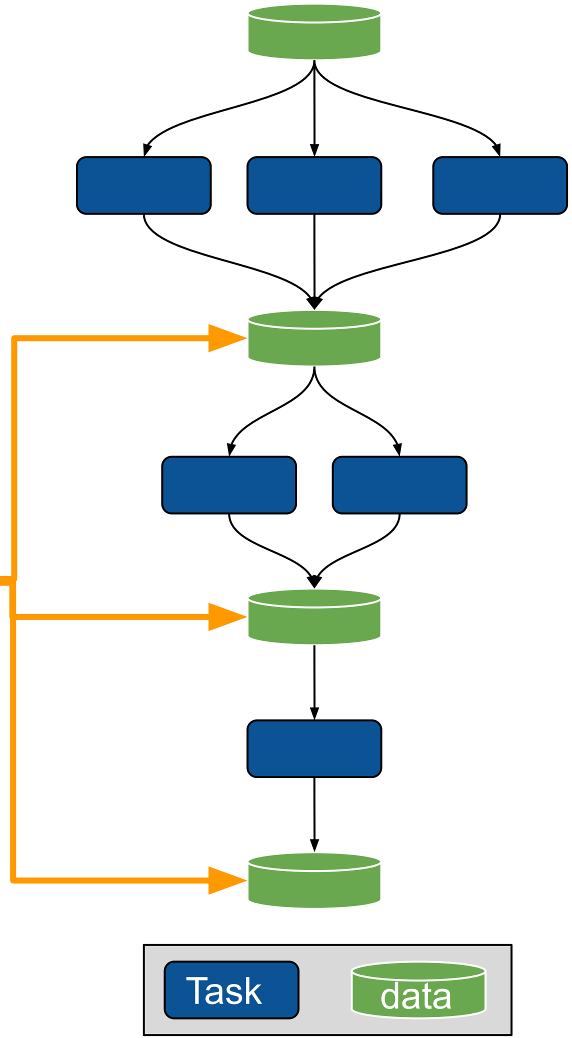



Bristol PRDA (50% FTE) starting in **April 2022** - once data lake is available

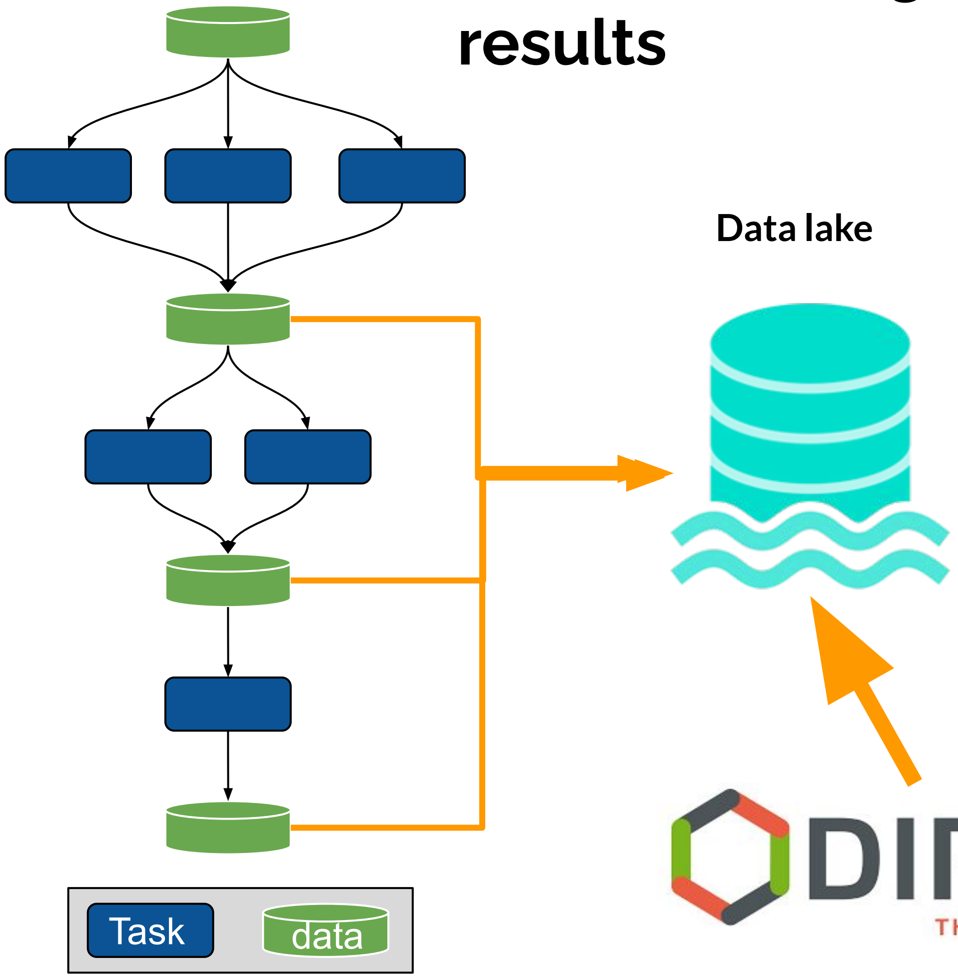
# WP5.1: Caching of intermediate analysis results

D5.1: Interoperability with UK data lake (Month 13-18)

- Develop a local caching mechanism
- **Starting point**: FAST-HEP and IRIS-HEP tools with Parsl - **access point for caching**
- Store intermediate results in the **UK data lake** (D1.1)
- Simple **lifetime and access permissions** to start with - can be extended later



# WP5.1: Caching of intermediate analysis results

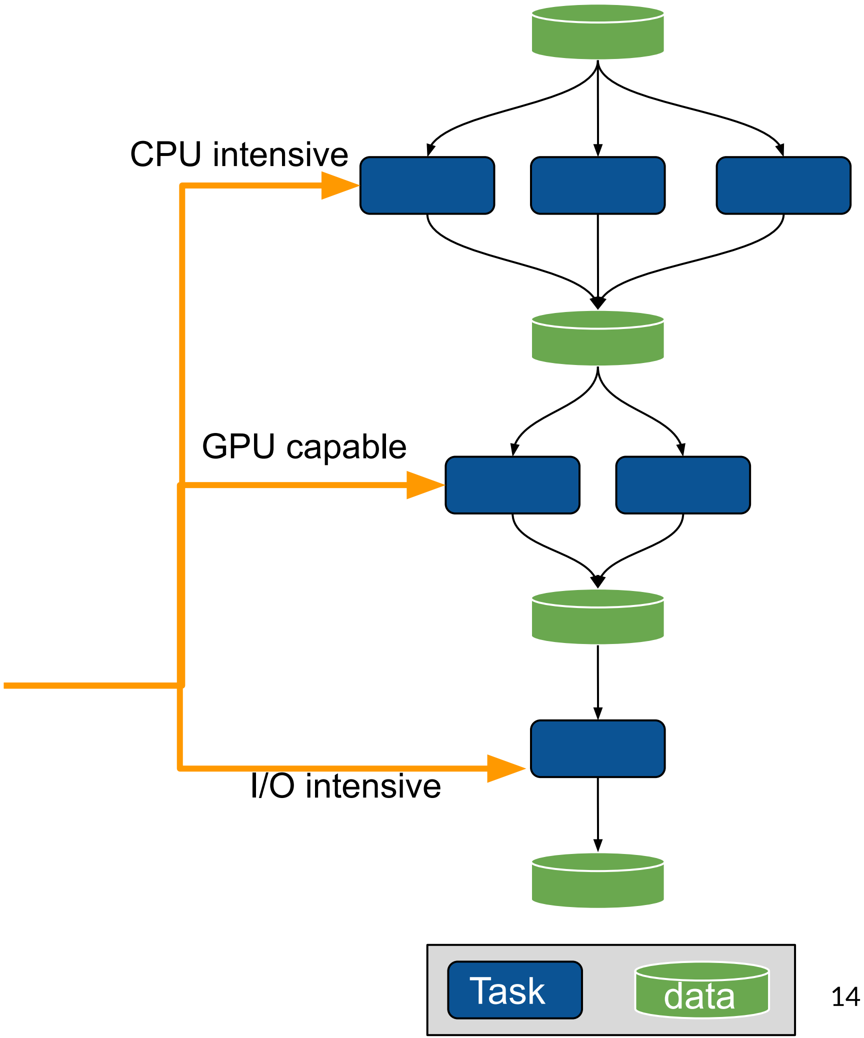



## D5.2 Integration of caching mechanism with workflow management (Month 19-24)

- Integration of caching with **workflow management** (D1.7)
- Skip  workflow steps if a cache entry exists

DIRAC
THE INTERWARE

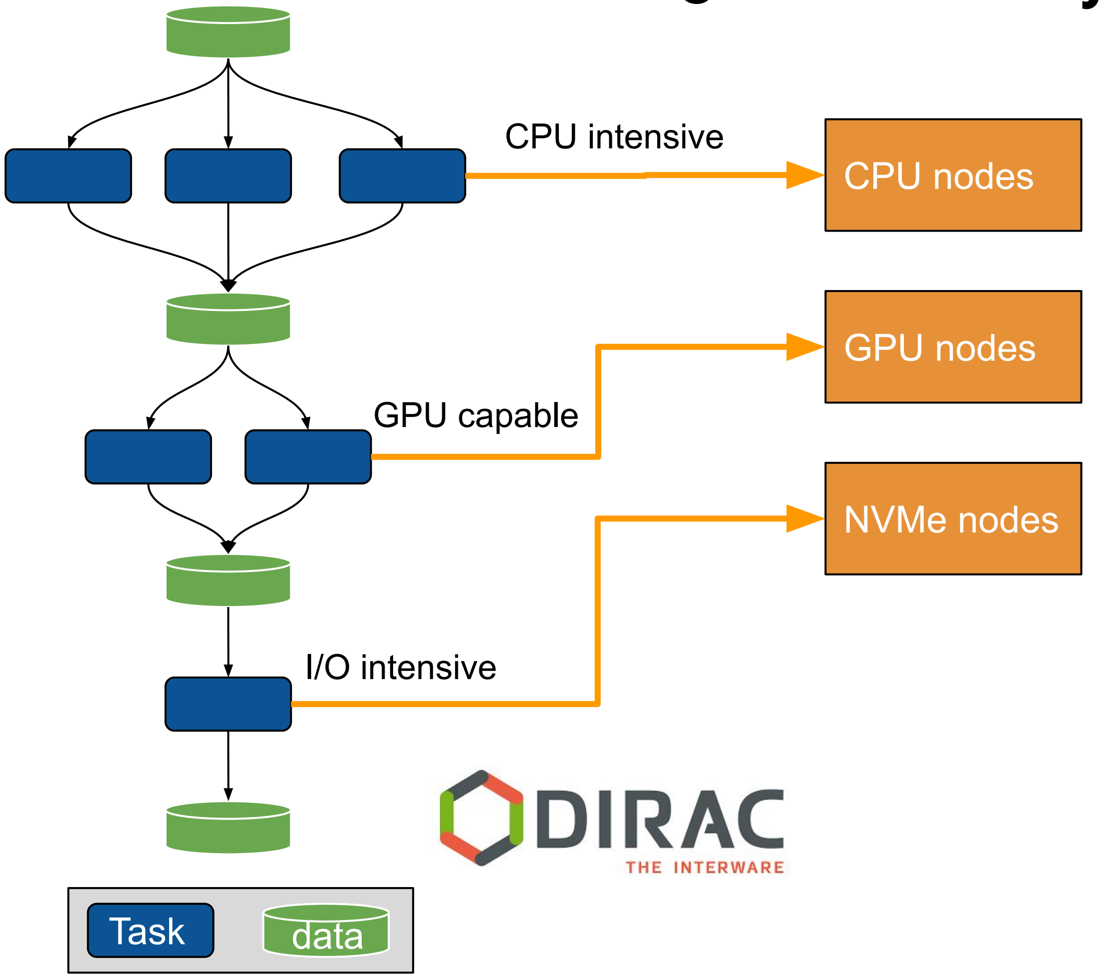
# WP5.2. Portability of analysis code




## D5.3 Per-site optimization of workloads (Month 25-30)

- Algorithms have **preferred hardware capabilities** (GPU, NVMe storage etc)
- Requires "self-aware" cluster - exchange mechanism for execute nodes and software
  - An extension of Machine/Job Features
- Collaboration with **virtual analysis facility (D1.4)**
- Requires **mechanism in software to pick implementation at runtime**

# WP5.2. Portability of analysis code



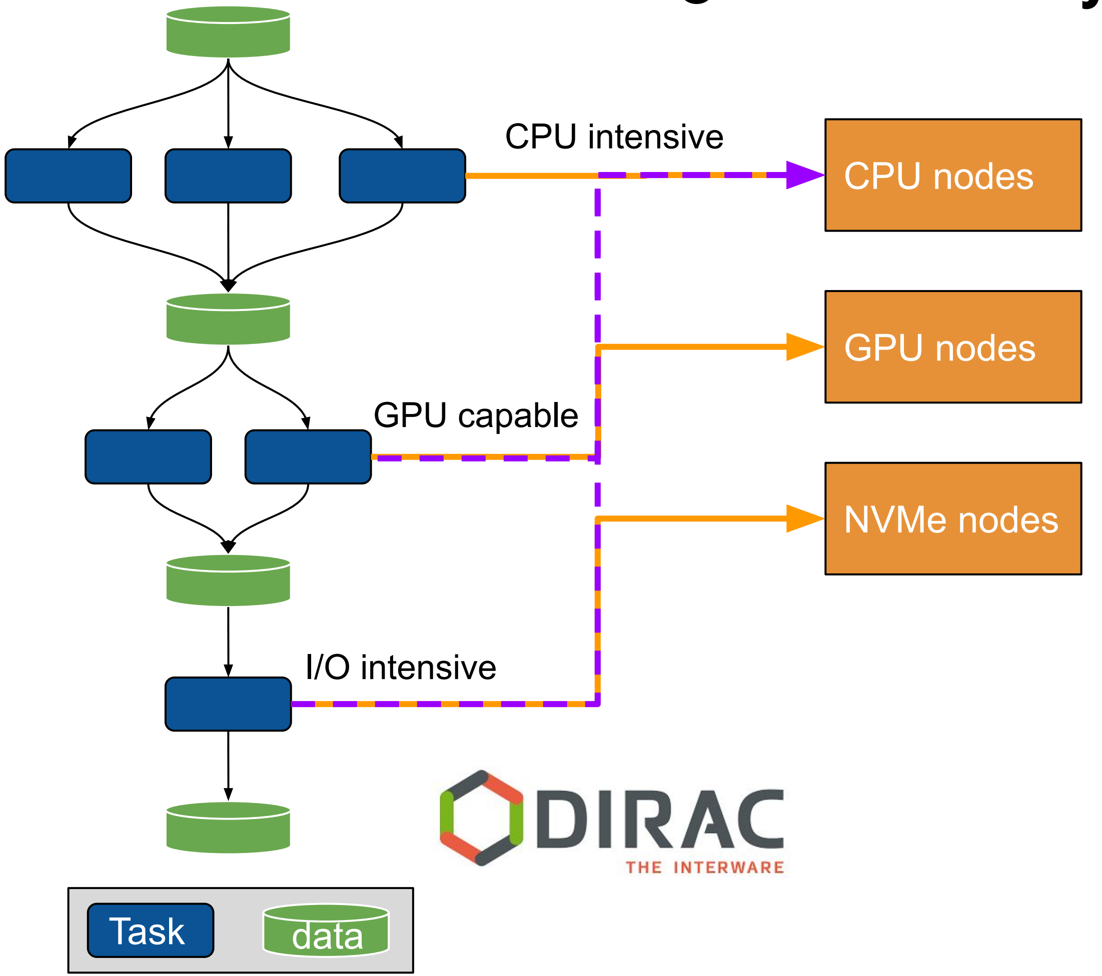

Current approach schedules **entire workflow on one type of node**

Breaking up into analysis steps enables fine-grained scheduling

What if waiting on specialized nodes >> just running on generic hardware?

→ Work with “Intelligent data and workflow management” (WP1)

# WP5.2. Portability of analysis code




## D5.4 Dedicated workload scheduling for the Workflow management (Month 31-36)

- Exchange mechanism between software and workflow management to **extract ranking of preferred hardware** (e.g. can use GPU to speed up 10x, but can also use CPU)
- Dynamic workload management (D1.7) can then **match according to ranking**
- Also useful to monitor computing resource requirements

# Collaboration with other Work Packages

# Collaboration areas

ECHEP and Swift-HEP efforts made it clear that we need to improve and extend the current software landscape

- **Optimization and platforms**: a common theme across WP2.1, WP3.1, WP4 and WP 5.2
- need to **improve software related teaching and training**
- Require a **sustainable funding model** for software engineering efforts
- **International collaboration** with HSF data analysis WG and IRIS-HEP analysis systems

Software quality and performance reliant on investments in training and experienced person power - Swift-HEP is a first step

# Creating sustainable careers for RSEs

Local HEP Research Software Engineer (RSE) teams working on local, national and international projects

- IRIS-HEP and FIRST-HEP in the USA are successful due to **targeted and sustained investment**
- To operate large-scale science experiments in the UK in the future we need something similar

RSEs funded through GridPP, Swift-HEP and research group projects could be a way forward - but each institute might need a different mix

# Summary

# Summary

Exciting times - Swift-HEP investment is very welcome

Swift-HEP will focus on analysis improvements in conjunction with Swift-HEP's "Intelligent data and workflow management" (WP1)

Data analysis field is challenging, but many international efforts exist and collaboration will be vital

Need to work on a long-term sustainable solution for software effort in UK HEP

# Thank you for listening

# Any questions or comments?

# Backup slides

# User control vs resource usage

Every experiment experiences failures in user jobs with both computing infrastructure (e.g. file access) and user code issues

| experiment | Fraction of failed analysis jobs | CPU efficiency | Main failure reason | Estimated lost CPU hours per year |
|---|---|---|---|---|
| ATLAS | ~20% of wall time | ~80% | Crashes in the user code | --- |
| Belle II | ~12 % (failed & rescheduled) | --- | Simple mistakes (e.g. syntax errors) | --- |
| CMS | ~20% | ~70% | File access failures | ~ 76Mh (~1.3 MGBP) |
| LHCb | ~20% | ~85% | Stalled jobs ( i.e. job killed by batch system before successful completion — the user set a wrong time limit for their jobs) | ~ 8.76 Mh (~ 150 kGBP) |

Some experiments have a job scouting system to try and predict user job failures

From B.Krikler’s talk (1st ECHEP workshop)

# Bugs and reproducibility


BBC | Sign in | News | Sport | Weather | Shop | Earth | Travel | Mo

NEWS

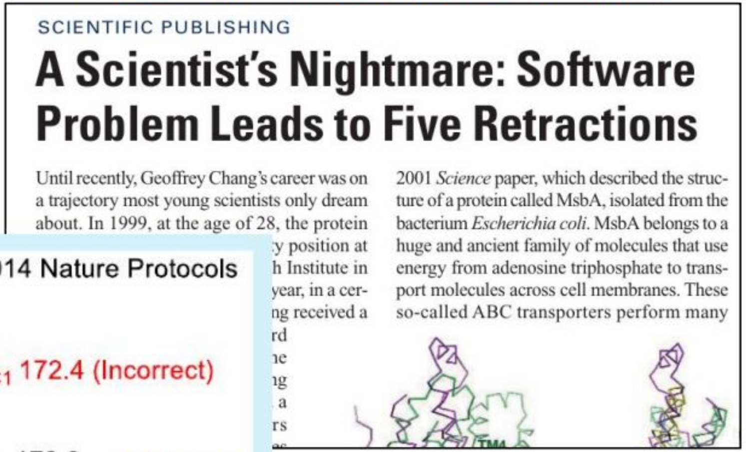
Home | Video | World | US & Canada | UK | Business | Tech | Science | Magazine | Ente

**Most scientists 'can't replicate studies by their peers'**

By Tom Feilden
Science correspondent, Today programme

22 February 2017 | Science & Environment

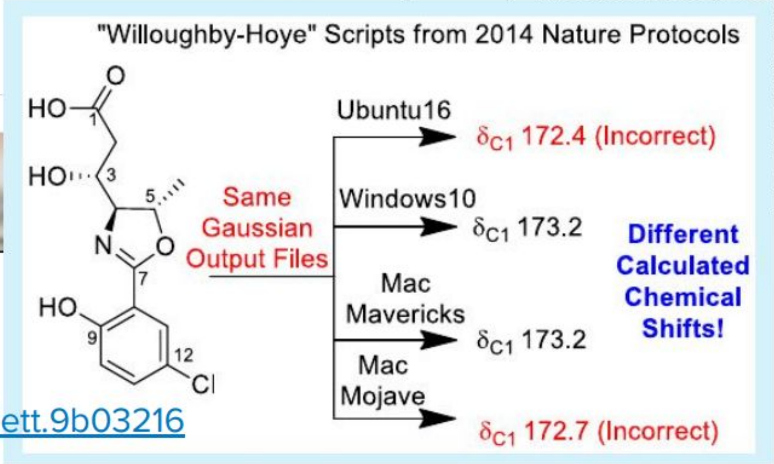
Dec. 2006 DOI: 10.1126/science.314.5807.1856


SCIENTIFIC PUBLISHING

**A Scientist’s Nightmare: Software Problem Leads to Five Retractions**

Until recently, Geoffrey Chang’s career was on a trajectory most young scientists only dream about. In 1999, at the age of 28, the protein ...y position at ...h Institute in ...year, in a cer- ...ng received a ...rd ...ne ...ng ...a ...rs

2001 *Science* paper, which described the structure of a protein called MsbA, isolated from the bacterium *Escherichia coli*. MsbA belongs to a huge and ancient family of molecules that use energy from adenosine triphosphate to transport molecules across cell membranes. These so-called ABC transporters perform many




Oct. 2019
DOI:10.1021/acs.orglett.9b03216

# CMS Analysis Facility/real-time data query system

Steps 3 & 4 have certainly overlap

- [Analysis with Apache Spark]
  - Caching capabilities
- [Real-time data query system]
  - Explores fast data query and caching
  - Shows big difference depending on how the data are accessed
  - code transformation performance should be similar to [TTreeReaderArray]
- Now all under [ServiceX]
  - Use Kafka for streaming
  - Cache results for instant replay

https://arxiv.org/abs/1711.01229

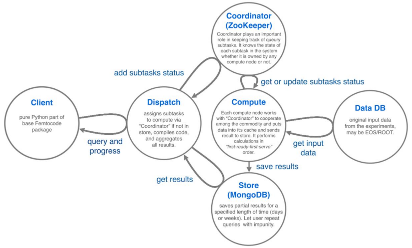
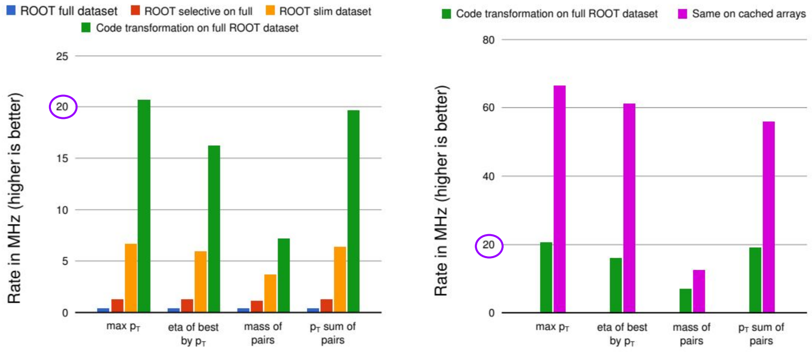



**Figure 2.** Schematic for distributed query processing to minimize cache misses (see text).

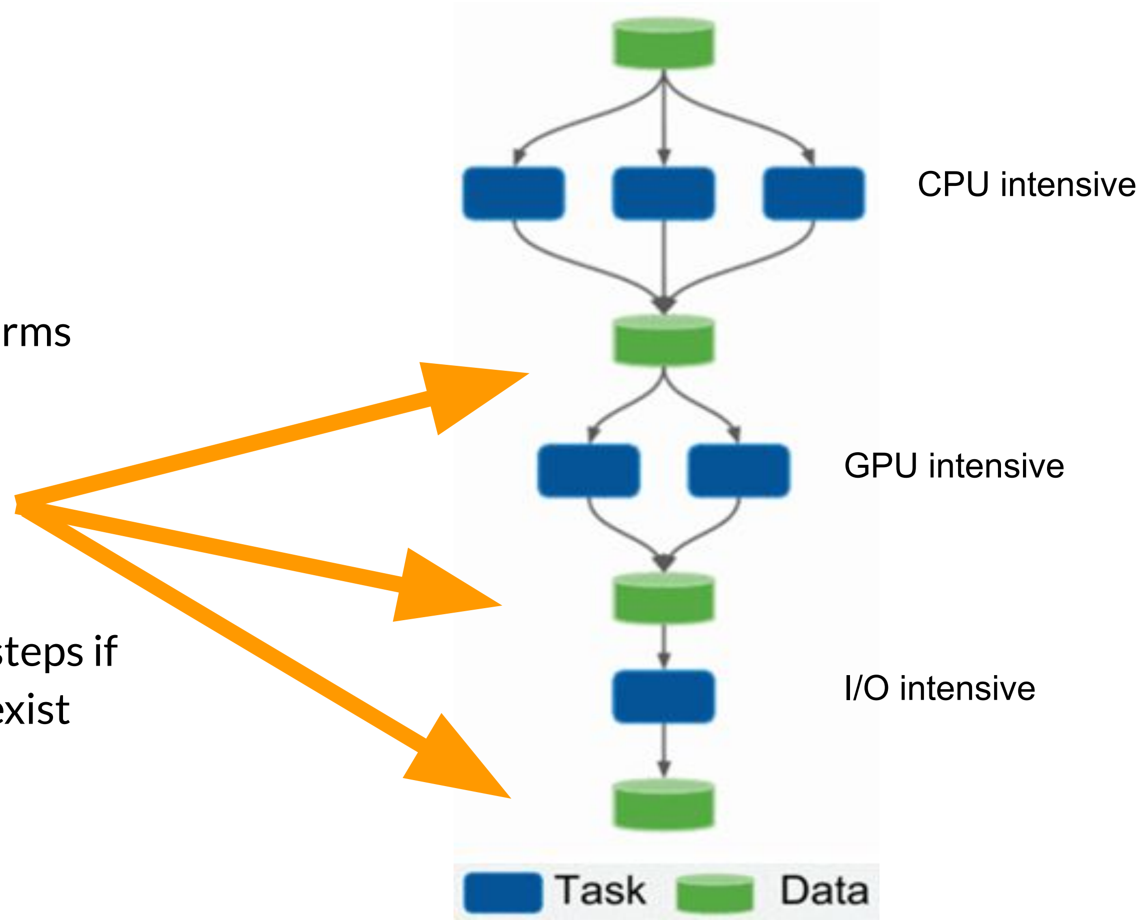


# Caching in a nutshell

Each analysis job performs several steps

Each step can have an intermediate result

Store results and skip steps if intermediate outputs exist



—

# Portability

Various python packages to optimize code for different architectures:

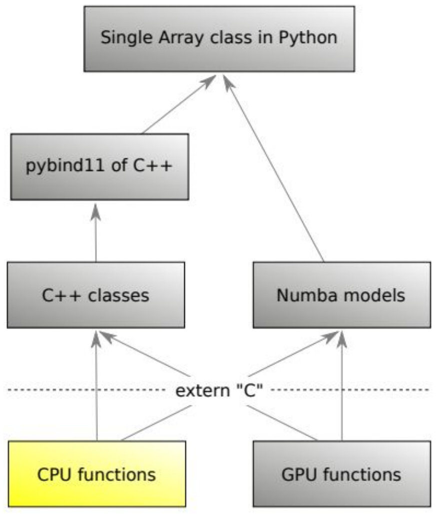
- Numba: vectorization, parallelism
- CudaPy: GPU
- PyNQ: [FPGA]()
  - Needs specific bitstream

```
@jit(nopython=True)
def haversine(s_lat,s_lng,e_lat,e_lng):
    # approximate radius of earth in km
    R = 6373.0

    s_lat = np.deg2rad(s_lat)
    s_lng = np.deg2rad(s_lng)
    e_lat = np.deg2rad(e_lat)
    e_lng = np.deg2rad(e_lng)
```

```
@jit(nopython=True, parallel=True)
def some_func(*args):
    out = np.zeros(length_of_output)
    for i in prange(n):
        # independent and parallel loop
        out[i] = ...some stuff...
    return out
```

```
@cuda.jit(device=True)
def
haversine_cuda(s_lat,s_lng,e_lat,e_ln
g):
    '''
    This is now a non-vectorized
    All inputs are expected to be
scalars.
    '''
    # approximate radius of earth in km
    R = 6373.0
    s_lat = s_lat * math.pi / 180
    s_lng = s_lng * math.pi / 180
    e_lat = e_lat * math.pi / 180
    e_lng = e_lng * math.pi / 180
```

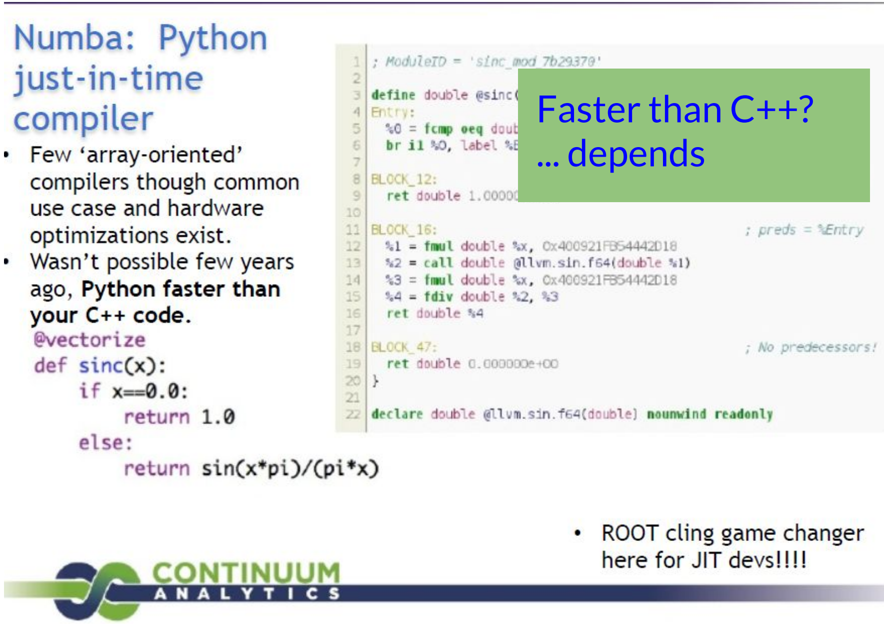



[Awkward 1.0, PyHEP 2019, Jim Pivarski]()

# Python for accelerated development

(the near future is Python)

- High level programming → fast to try something
- Compact language → fewer lines of code, fewer bugs
- Quick refactoring
- Access to Big data tools → ML + distributed processing
- Full-Stack prototyping (even FPGA)
- Can be easy to change architectures (e.g. CPU → GPU with numba/tensorflow)




https://zenodo.org/record/1418513#.XniKnoj7TAQ

# results for high complexity anaylisi (1,441,999 events)

| method | HDD (CERN CI) | SSD | **comment** |
| --- | --- | --- | --- |
| numpy | 37.1 s | 12.5 s | Advanced python |
| Loop depth 3 | 1436.8 s | 822.4 s | Beginner python |
| C++ loop depth 3 | 8.9 s | 4.3 s (18.5 s) | Advanced ROOT |
| C++ GetEntry | 308.9 s | 148.3 s | Beginner ROOT |
| C++ GetEntry + disabling unused branches | --- | 15.7 s (51.2s) | Beginner ROOT |

Note 1: Arrays stored in NanoAOD do not work well with SetBranch method (output is wrong for some events))
Note 2: Using namespaces in the ROOT macro, increases processing time (???)

# WP5.1: Caching of intermediate analysis results

- D5.3 Per-site optimization of workloads (Month 25-30)
  - Choose implementation based on available hardware (GPU, NVMe storage etc)
  - Requires “self-aware” cluster - exchange mechanism for execute nodes and software
  - Collaboration with virtual analysis facility
  - Requires mechanism in software to pick implementation at runtime
- D5.2 Integration of caching mechanism with workflow management (Month 19-24)
  - Integration of caching with workflow management (D1.7)
  - Skip  workflow steps if a cache entry exists

# WP5.2. Portability of analysis code

- D5.1: Interoperability with UK data lake (Month 13-18)
  - Develop a local caching mechanism
  - Starting point: FAST-HEP and IRIS-HEP tools with Parsl - access point for caching
  - Store intermediate results in the UK data lake (D1.1)
  - Simple lifetime and access permissions to start with - can be extended later
- D5.4 Dedicated workload scheduling for the Workflow management (Month 31-36)
  - Exchange mechanism between software and workflow management to extract ranking of preferred hardware (e.g. can use GPU to speed up 10x, but can also use CPU)
  - Dynamic workload management (D1.7) can then match according to ranking
  - Also useful to monitor computing resource requirements

# WP5.2. Portability of analysis code

## D5.4 Dedicated workload scheduling for the Workflow management (Month 31-36)

- Exchange mechanism between software and workflow management to **extract ranking of preferred hardware** (e.g. can use GPU to speed up 10x, but can also use CPU)
- **Dynamic workload management** (D1.7) can then match according to ranking
- Also useful to **monitor computing resource requirements**