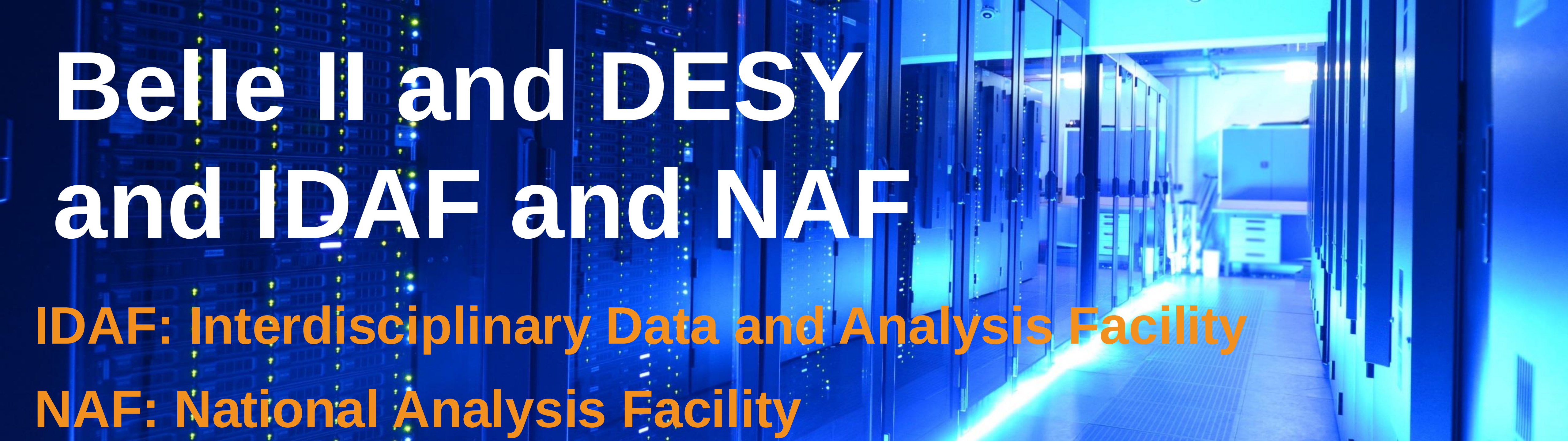


# Belle II and DESY and IDAF and NAF

## IDAF: Interdisciplinary Data and Analysis Facility

## NAF: National Analysis Facility

Yves Kemp, Christian Voß, Thomas Hartmann, Christoph Beyer, et al., DESY IT

Belle II Germany Meeting
Hamburg, 1.10.2024

# DESY research divisons ... In a nutshell  (those in Hamburg)

Accelerators »

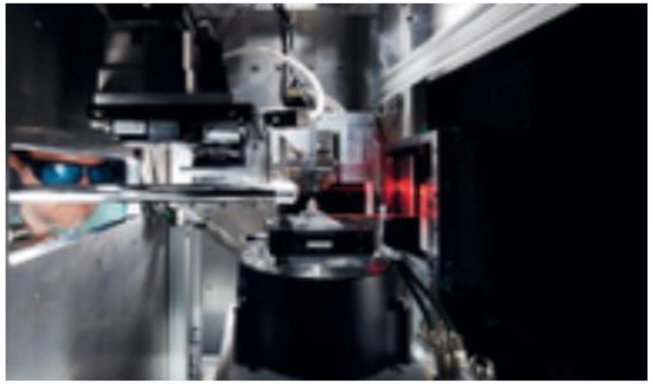
Running / Operating:

- Petra III, FLASH, XFEL, ...

Planning:

- Petra IV

General Accelerator R&D



Photon science »

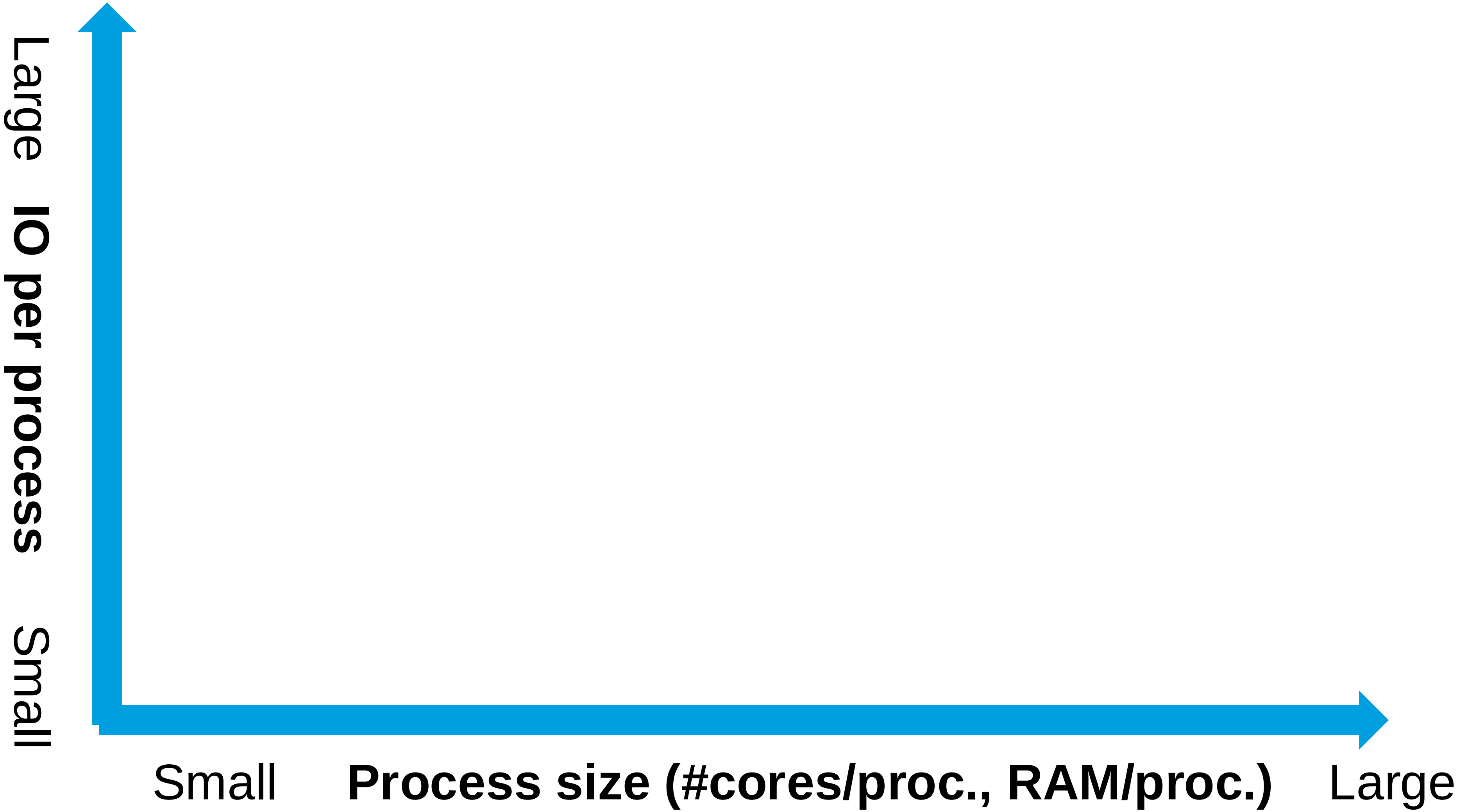
Petra III, FLASH, EXFEL, CFEL, CSSB, EMBL, HZG

Particle physics »

- LHC, HL-LHC
- Belle II
- ILC, ALPS, ....
- Theory division

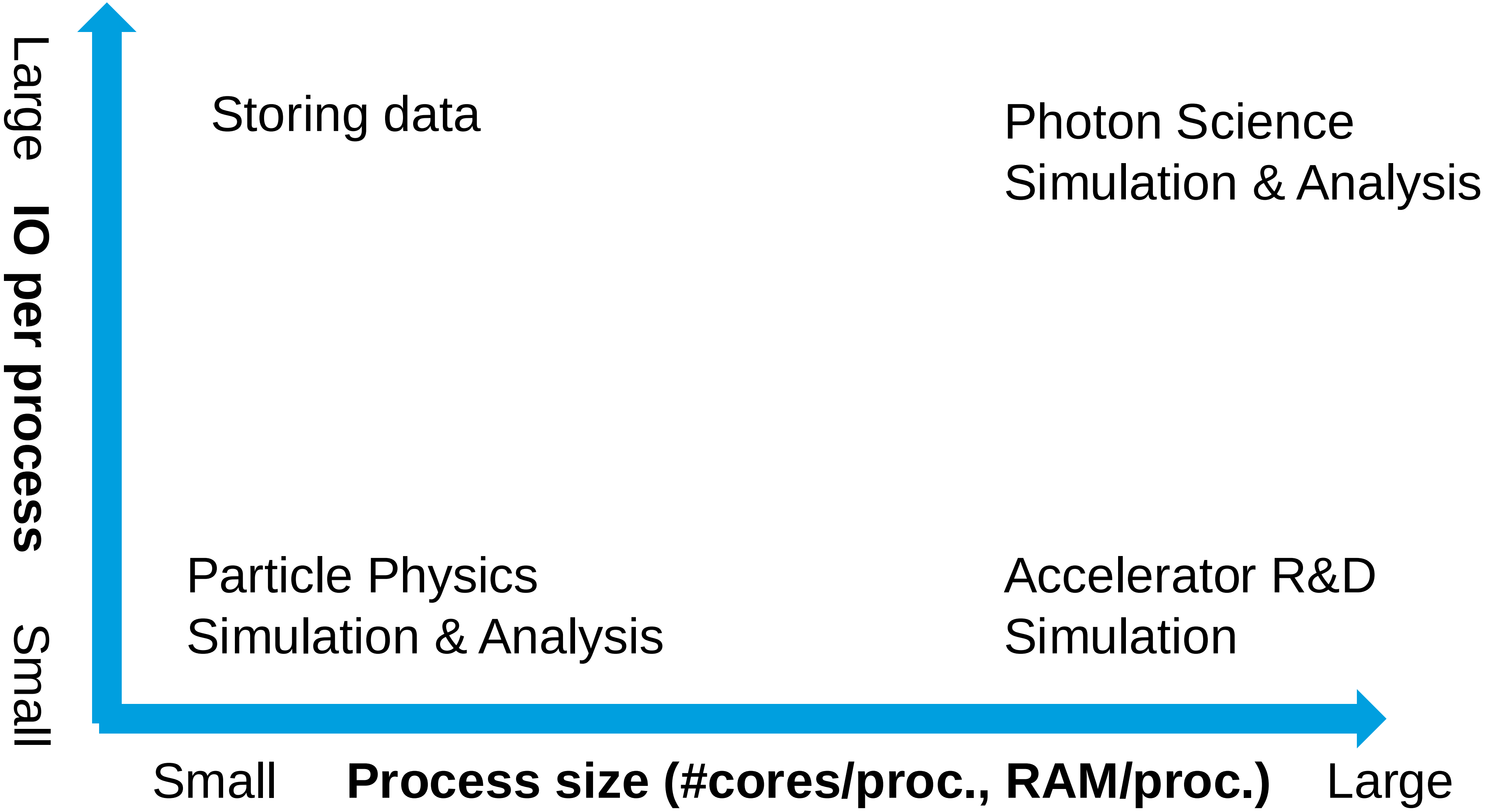
# Computational requirements: Job size vs IO needs

## Very very coarse



# Computational requirements: Job size vs IO needs

## Very very coarse



Large **IO per process** Small

Storing data

Photon Science
Simulation & Analysis

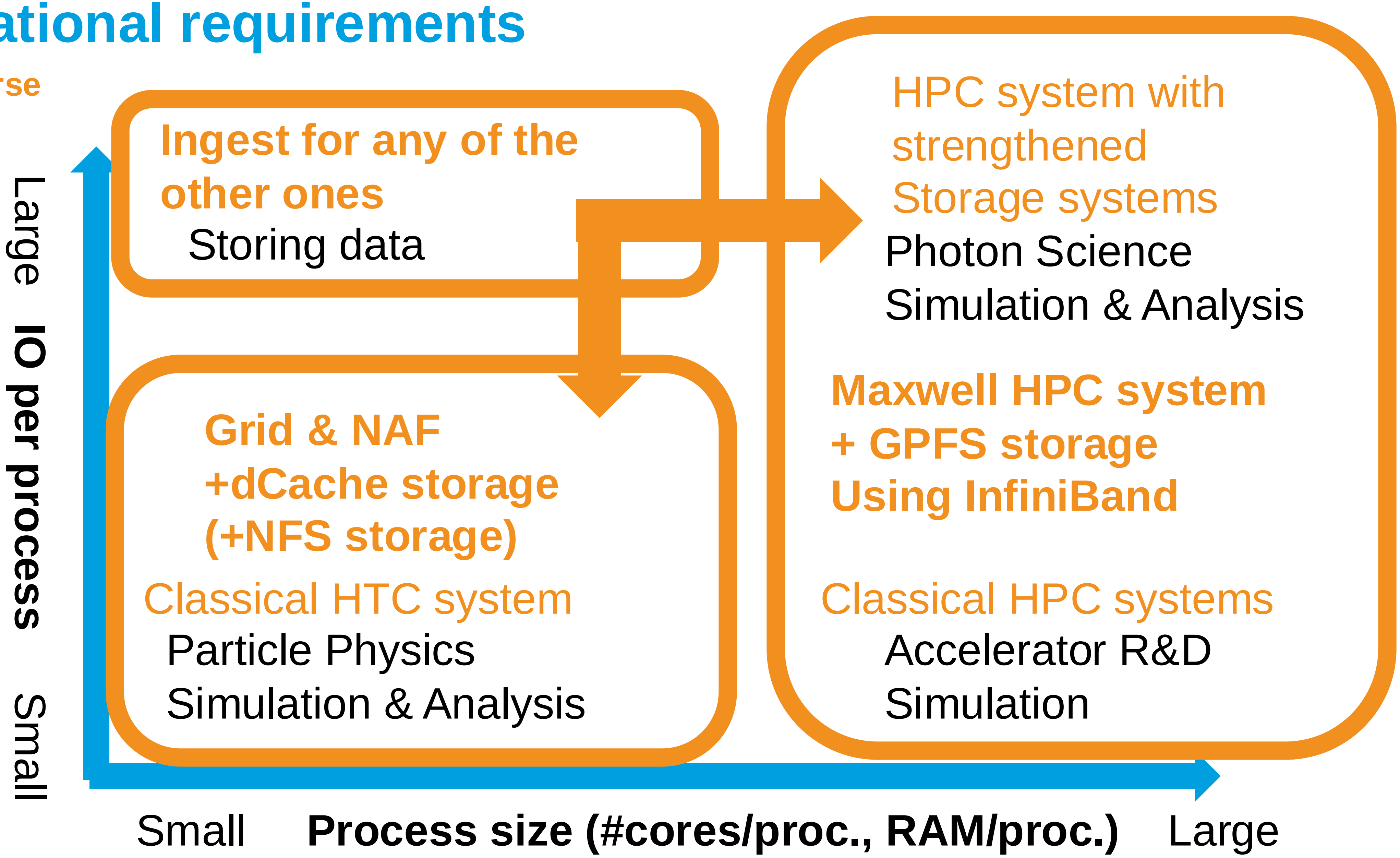
Particle Physics
Simulation & Analysis

Accelerator R&D
Simulation

Small **Process size (#cores/proc., RAM/proc.)** Large

# Computational requirements

Very very coarse



**Ingest for any of the other ones**

Storing data

HPC system with strengthened Storage systems

Photon Science

Simulation & Analysis

**Grid & NAF +dCache storage (+NFS storage)**

Classical HTC system

Particle Physics

Simulation & Analysis

**Maxwell HPC system + GPFS storage Using InfiniBand**

Classical HPC systems

Accelerator R&D

Simulation

Large **IO per process** Small

Small **Process size (#cores/proc., RAM/proc.)** Large

# The IDAF is about *people and experiments*

- Accelerator Data

FLASH. Free-Electron Laser FLASH

European XFEL

FF▸▸

- Accelerator Development Data

SINBAD

- HPC simulations
- Test-beam data

Detector and Accelerator R&D

- Facility User Data

European XFEL

PETRA III

FLASH. Free-Electron Laser FLASH

- Data of external Partners

CFEL SCIENCE

CSSB Centre for Structural Systems Biology

EMBL

Helmholtz-Zentrum hereon

Research with Photons

- Particle Physics Data

CMS

ATLAS EXPERIMENT

LHCb

Belle II

ALPS II

ilc international linear collider

- Astro-Particle Data

IceCube

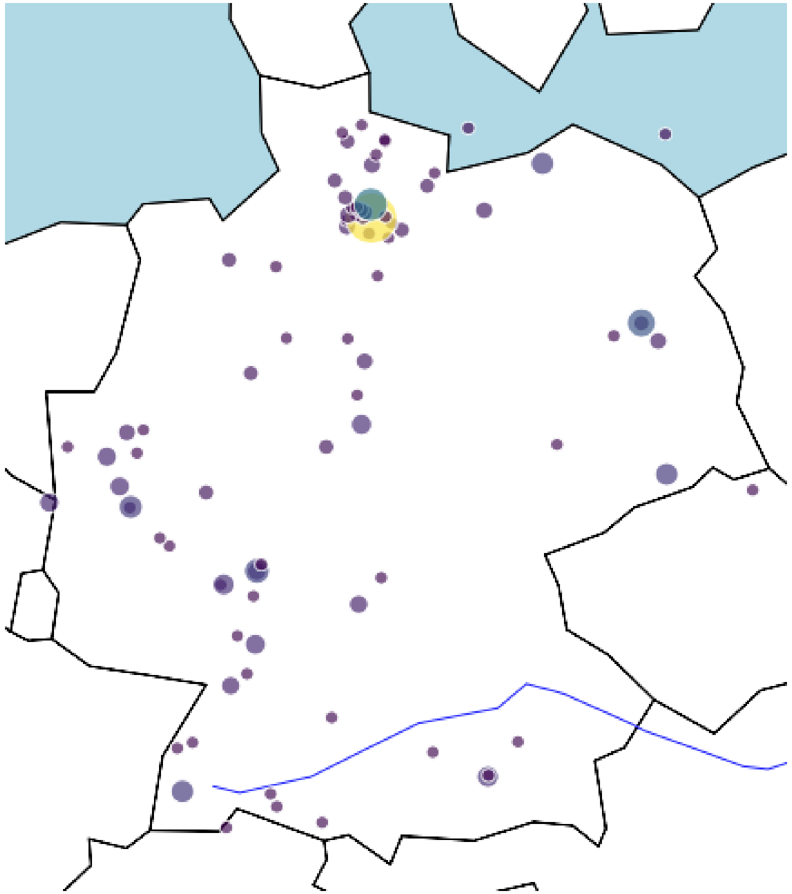
H.E.S.S.

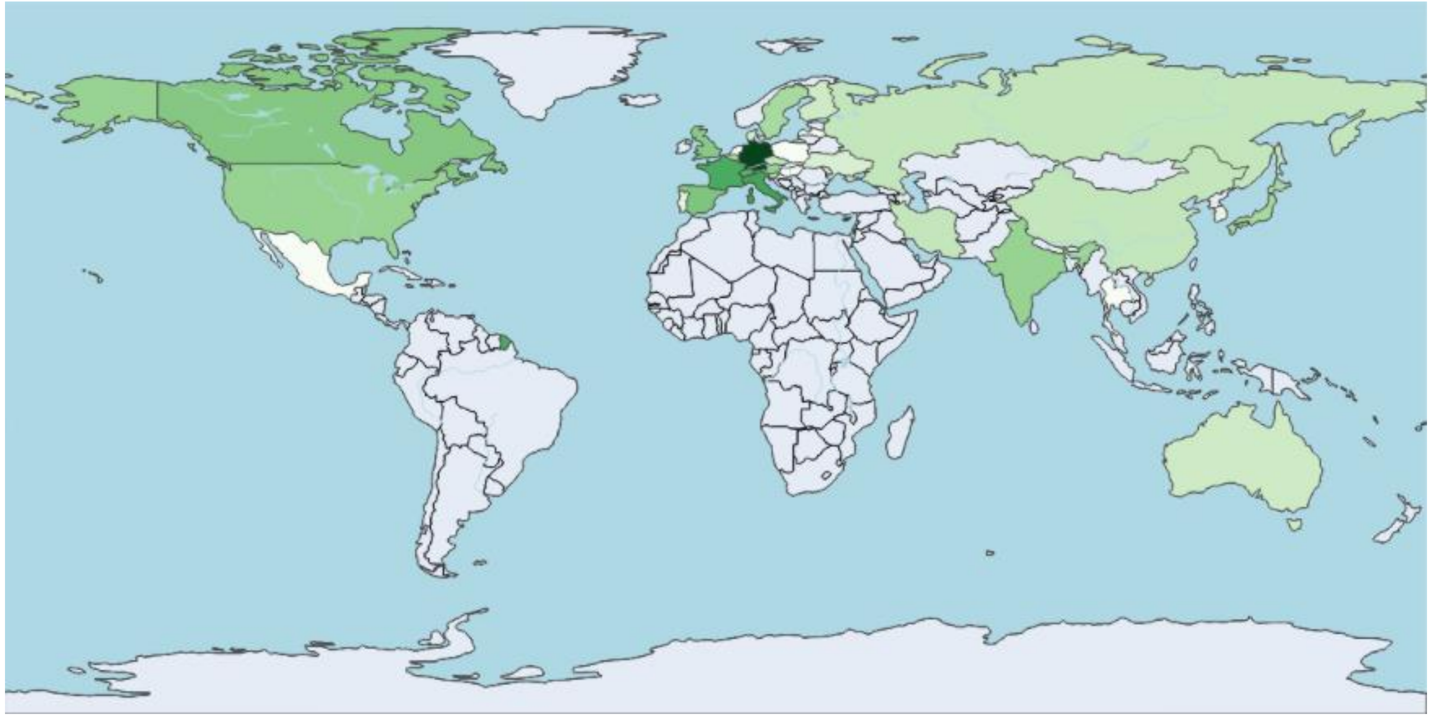
cta cherenkov telescope array

Astro- Particle Physics

# … and where they come from

## logins during two weeks in October 2023





Only NAF & Maxwell logins are accounted for (no Grid submission)

… mostly from academia (universities and institues)

… some commercial users

# The IDAF is about *services*

Interactive access and compute

Federated access to compute & data

Large-scale compute and workflows

Integration with DAQ

Data storage & management

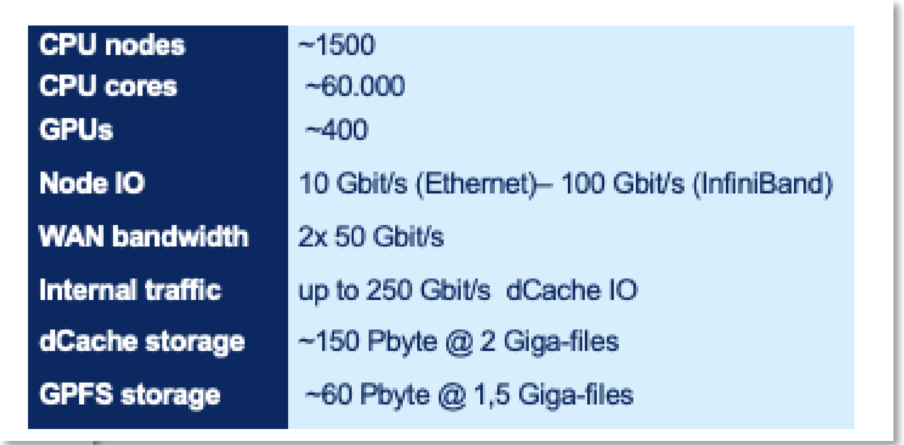
Metadata management

Software and containers

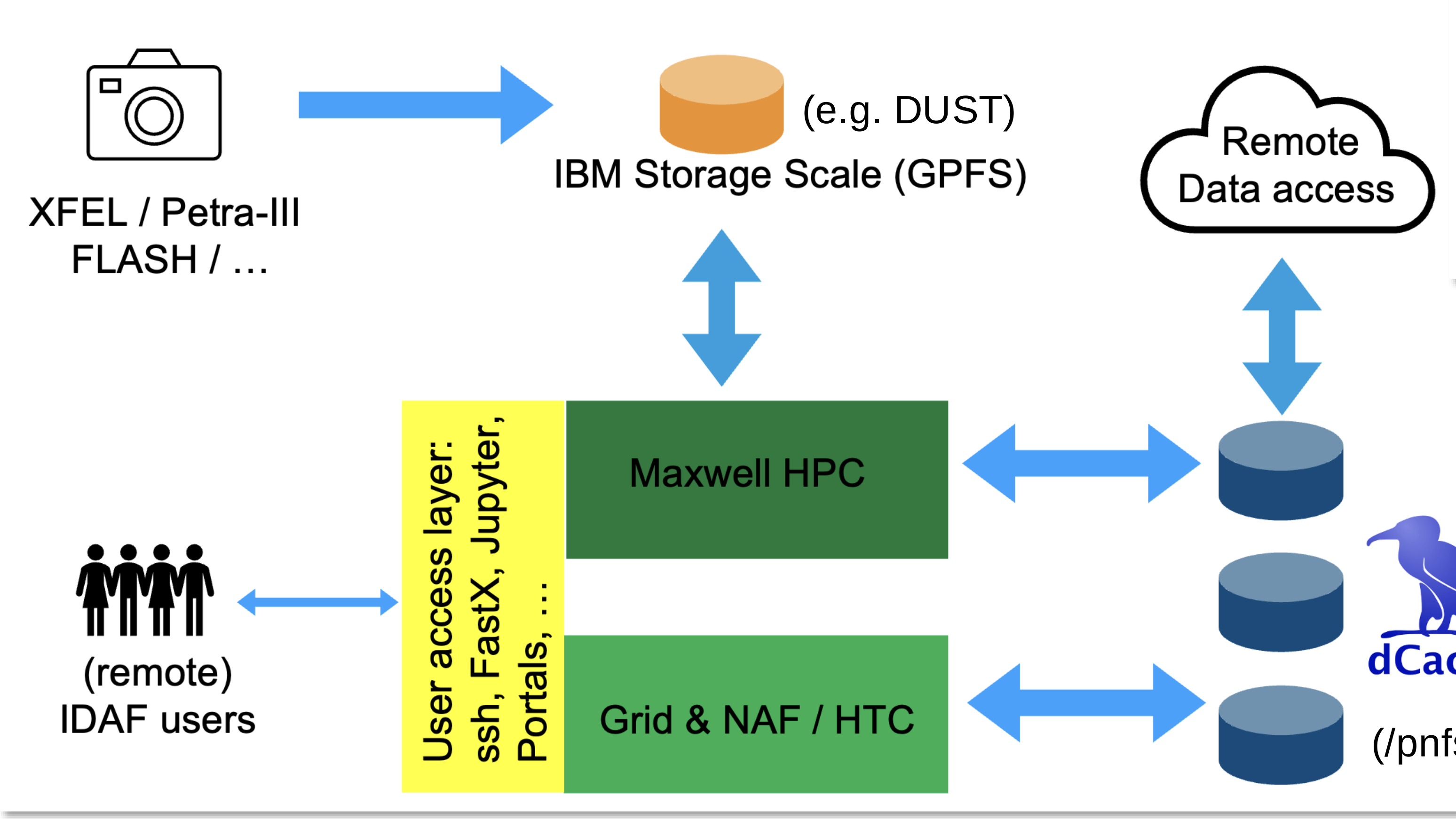
Documentation, support, training

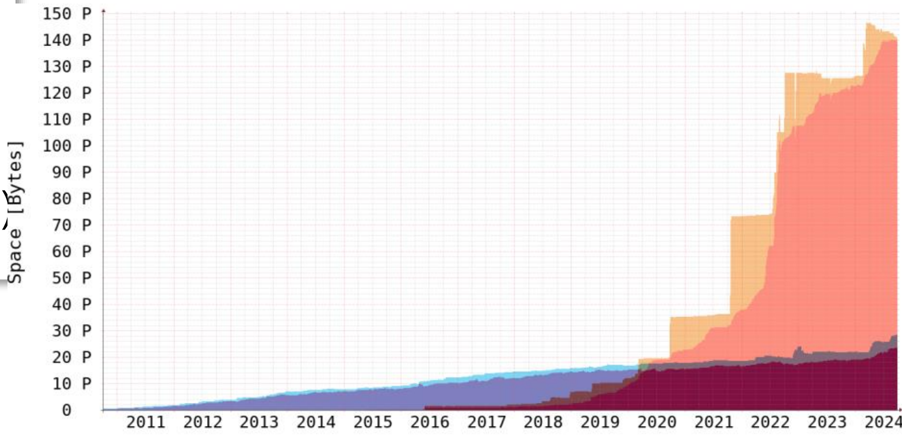
Code management and CI/CD

# … and the IDAF is also about infrastructure

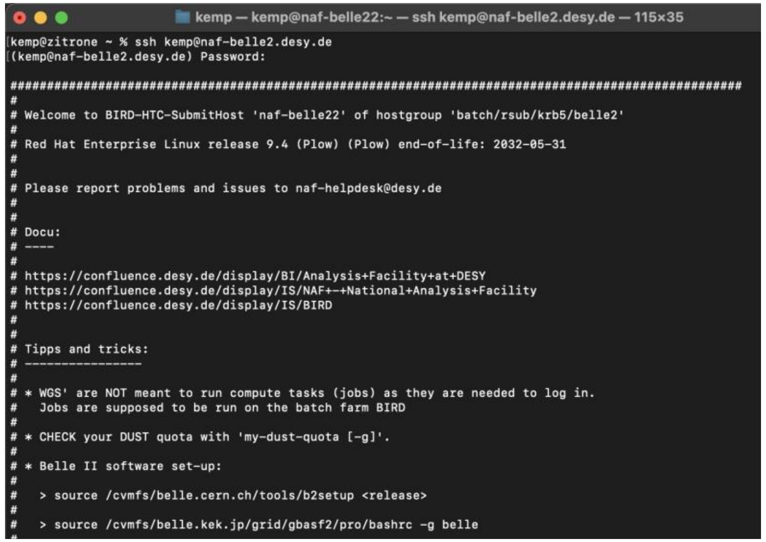


| | |
|---|---|
| CPU nodes | ~1500 |
| CPU cores | ~60.000 |
| GPUs | ~400 |
| Node IO | 10 Gbit/s (Ethernet)– 100 Gbit/s (InfiniBand) |
| WAN bandwidth | 2x 50 Gbit/s |
| Internal traffic | up to 250 Gbit/s dCache IO |
| dCache storage | ~150 Pbyte @ 2 Giga-files |
| GPFS storage | ~60 Pbyte @ 1,5 Giga-files |





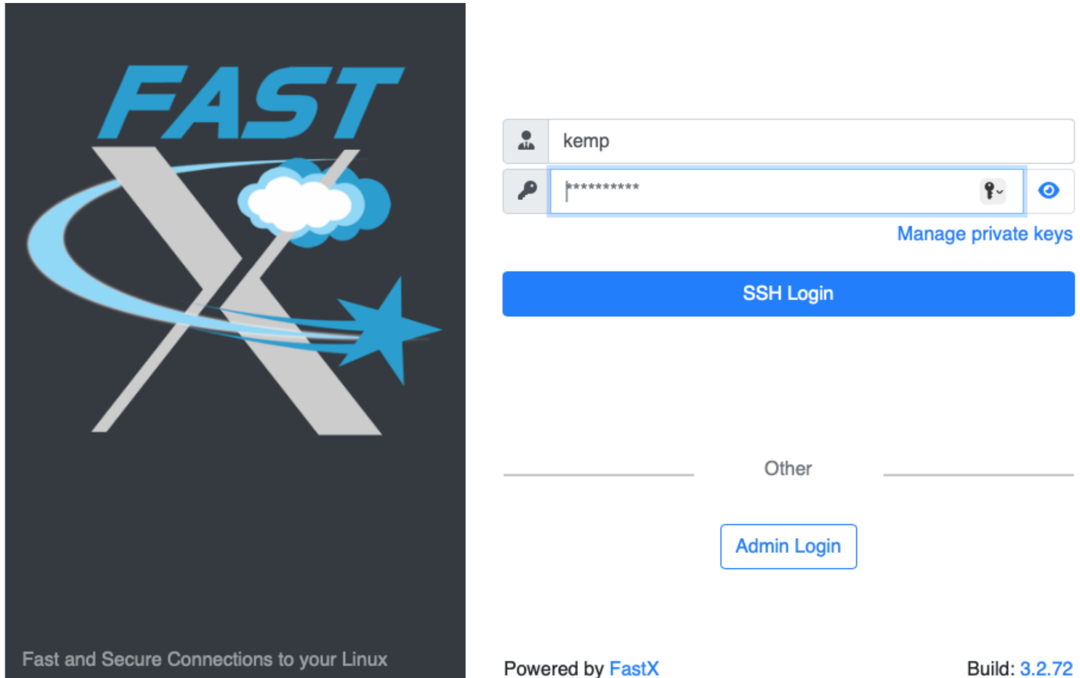
# Access to the NAF



kemp — kemp@naf-belle22:~ — ssh kemp@naf-belle2.desy.de — 115×35

```
[kemp@zitrone ~ % ssh kemp@naf-belle2.desy.de
[(kemp@naf-belle2.desy.de) Password:

########################################################################################################
#
# Welcome to BIRD-HTC-SubmitHost 'naf-belle22' of hostgroup 'batch/rsub/krb5/belle2'
#
# Red Hat Enterprise Linux release 9.4 (Plow) (Plow) end-of-life: 2032-05-31
#
#
# Please report problems and issues to naf-helpdesk@desy.de
#
#
# Docu:
# ----
#
# https://confluence.desy.de/display/BI/Analysis+Facility+at+DESY
# https://confluence.desy.de/display/IS/NAF+-+National+Analysis+Facility
# https://confluence.desy.de/display/IS/BIRD
#
#
# Tipps and tricks:
# ----------------
#
# * WGS' are NOT meant to run compute tasks (jobs) as they are needed to log in.
#   Jobs are supposed to be run on the batch farm BIRD
#
# * CHECK your DUST quota with 'my-dust-quota [-g]'.
#
# * Belle II software set-up:
#
#   > source /cvmfs/belle.cern.ch/tools/b2setup <release>
#
#   > source /cvmfs/belle.kek.jp/grid/gbasf2/pro/bashrc -g belle
```



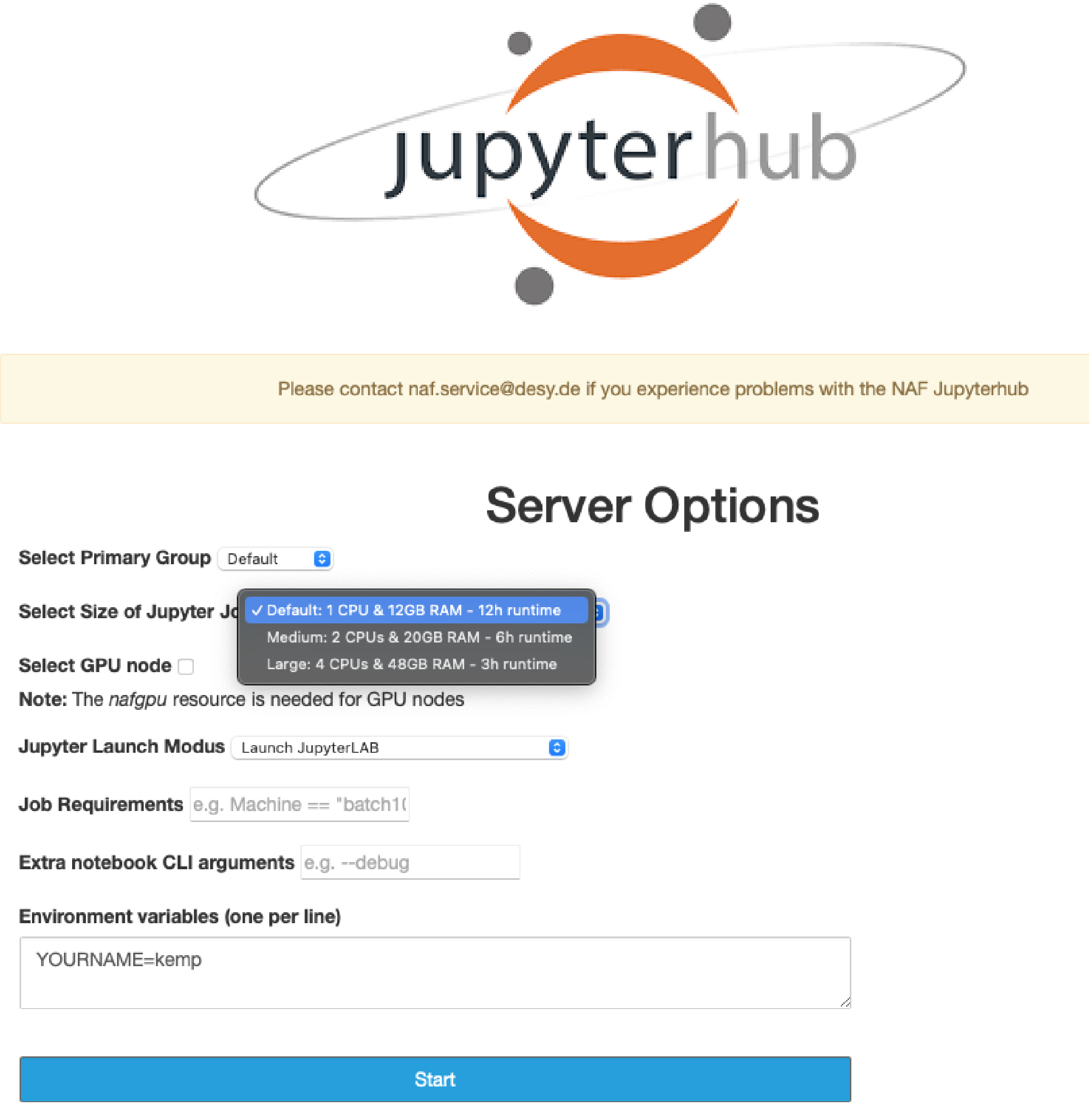
kemp

**********

Fast and Secure Connections to your Linux

Powered by FastX

Build: 3.2.72



Please contact naf.service@desy.de if you experience problems with the NAF Jupyterhub

## Server Options

Select Primary Group: Default

Select Size of Jupyter Jo

✓ Default: 1 CPU & 12GB RAM - 12h runtime
Medium: 2 CPUs & 20GB RAM - 6h runtime
Large: 4 CPUs & 48GB RAM - 3h runtime

Select GPU node □

**Note:** The *nafgpu* resource is needed for GPU nodes

Jupyter Launch Modus: Launch JupyterLAB

Job Requirements: e.g. Machine == "batch1(

Extra notebook CLI arguments: e.g. --debug

Environment variables (one per line)

YOURNAME=kemp

# JUPYTER Notebooks

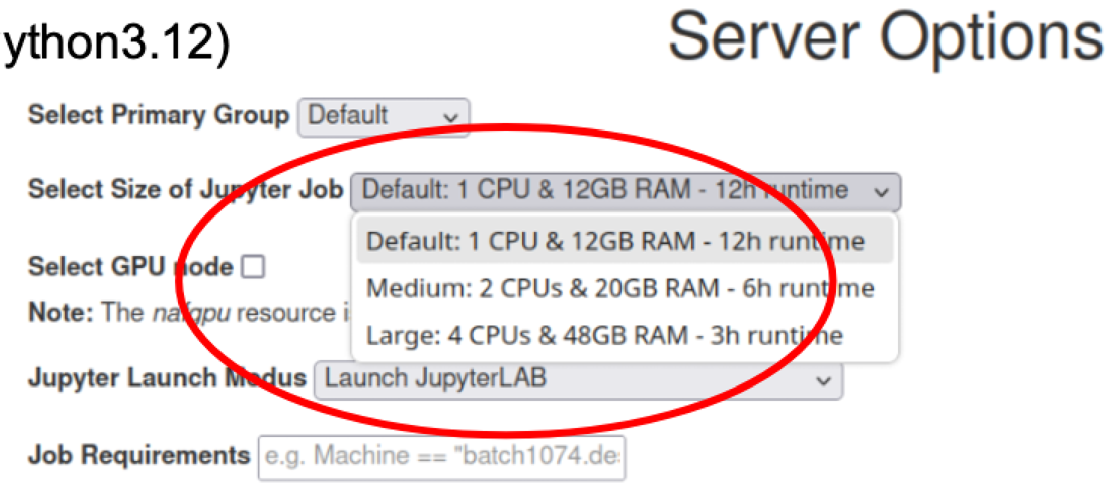
## Interactive access to the NAF through www

- Introduced 2018 @ DESY when IT infrastructure collaboration with BELLE2 was intensified
- Jupyterhub bridging the NAF into the WAN
- Small reserved slots for notebooks on the NAF workers – one per workenode
  - 1 core 1,5 GB memory reserved ressources
  - Soft policy, notebook stopped if mem-usage > 4,5 GB
- Fast start of notebooks due to separate negotiator/collector (<30 secs)
- Users can use htmap and python bindings to outsource workload into the pool
- BELLE as suspected early adapters but soon more usage also by other individuals and VOs e.g. ATLAS
- The recent update to EL9 and re-installation of the pool included the JUPYTER infrastructure and was also used to improve the service and hopefully make the notebooks more enjoyable

# JUPYTER Notebooks

## Next generation

- JHUB & notebooks upgraded (JupyterHub version 5.0.0, Python3.12)
- New notebook classes:
  - Default: 1 CPU / 12 GB RAM / 12h runtime
  - Medium: 2 CPUs / 20 GB RAM / 6h runtime
  - Large: 4 CPUs / 48 GB RAM / 3h runtime



Server Options

Select Primary Group: Default

Select Size of Jupyter Job: Default: 1 CPU & 12GB RAM - 12h runtime
- Default: 1 CPU & 12GB RAM - 12h runtime
- Medium: 2 CPUs & 20GB RAM - 6h runtime
- Large: 4 CPUs & 48GB RAM - 3h runtime

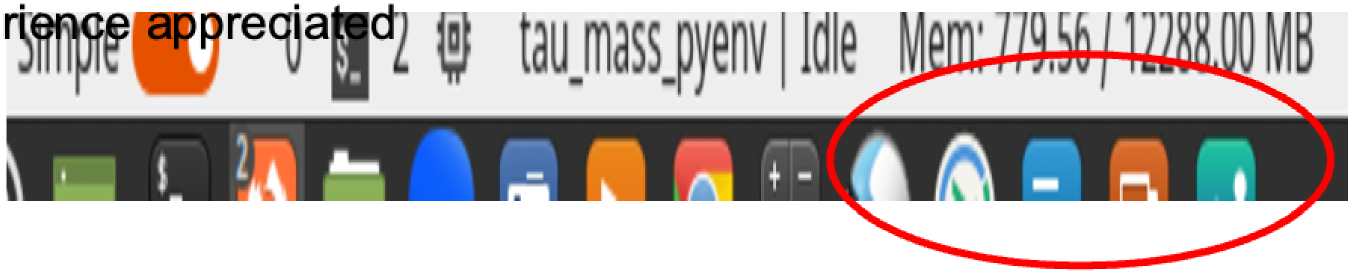
Select GPU node □

Note: The *naf-gpu* resource i

Jupyter Launch Modus: Launch JupyterLAB

Job Requirements: e.g. Machine == "batch1074.de

- Default notebooks run on all pool nodes in reserved slots (similar setup to old pool)
- Medium & large notebooks run on 2 dedicated servers
- Feedback about new sizing and user experience appreciated
- RAM taxometer now in place



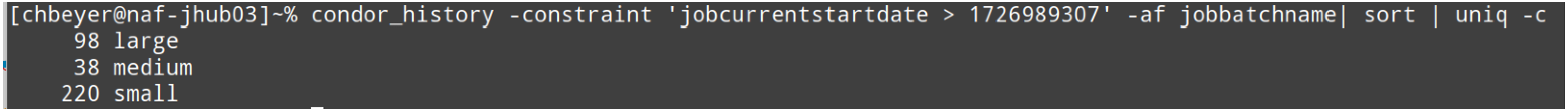
Simple 0 2 tau_mass_pyenv | Idle Mem: 779.56 / 12288.00 MB

- Suggestion: Have a ‘show-us-your-notebook’ session later this year in order to connect notebook users over VO/batchsystem borders and discuss further experiences and needs

# JUPYTER Notebooks

## Some issues

- Users need to adapt their Python envrionments, pathes etc.
- Some not yet fully understood problems
  - Some notebook processes don’t get killed after notebook finishes and keep users from starting a fresh one (503 error – spawn a RT-ticket in that case)
  - Memory accounting is sometimes not correct
  - Some notebooks become slow and unresponsive on particular nodes
  - All of the above is under surveillance and mostly due to the integration on Jupyterhub into EL9 & cgroup V2 – we hope to eliminate these problems in the upcoming weeks
- Usage during a typical week:

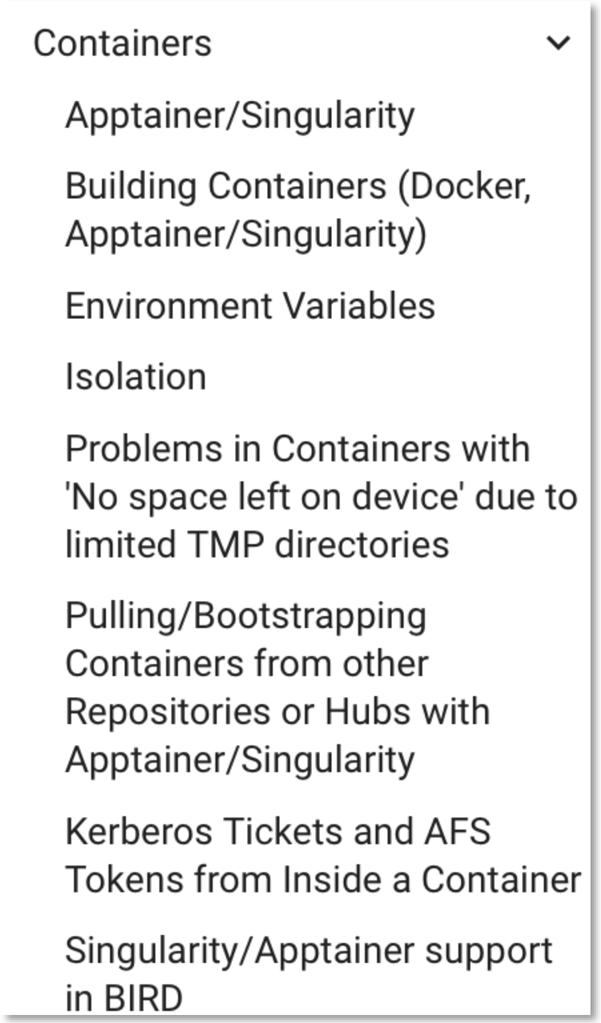


```
[chbeyer@naf-jhub03]~% condor_history -constraint 'jobcurrentstartdate > 1726989307' -af jobbatchname| sort | uniq -c
     98 large
     38 medium
    220 small
```

# Using containers

- You can start your own containers (Apptainer)
- You can get your image e.g. from CVMFS … or pull it from somewhere else
- The batch system allows for configuration of an image where your job will be run on

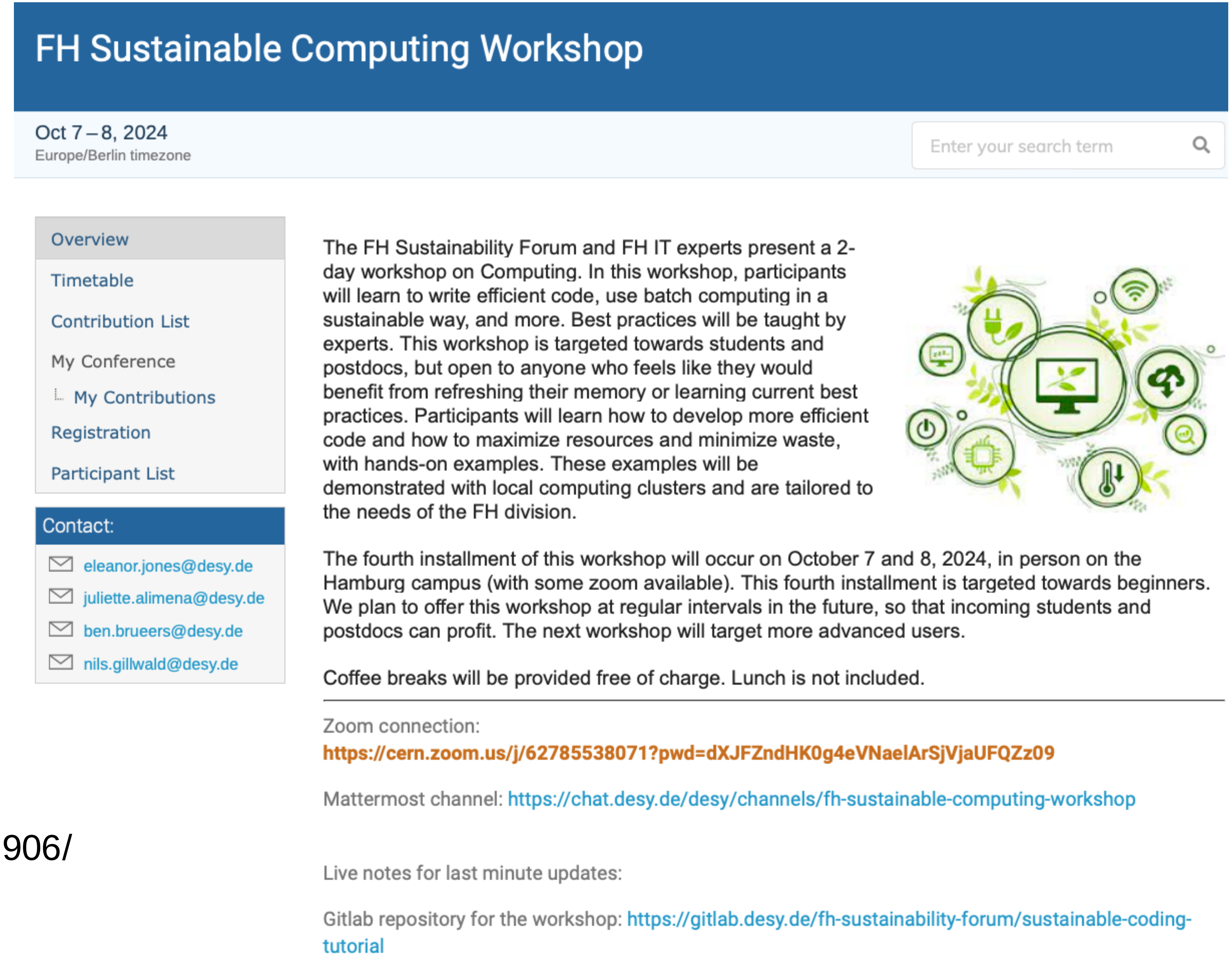
https://naf.desy.de/ →



Containers

- Apptainer/Singularity
- Building Containers (Docker, Apptainer/Singularity)
- Environment Variables
- Isolation
- Problems in Containers with 'No space left on device' due to limited TMP directories
- Pulling/Bootstrapping Containers from other Repositories or Hubs with Apptainer/Singularity
- Kerberos Tickets and AFS Tokens from Inside a Container
- Singularity/Apptainer support in BIRD

# NAF intro school



## FH Sustainable Computing Workshop

Oct 7 – 8, 2024
Europe/Berlin timezone

Contact:

- eleanor.jones@desy.de
- juliette.alimena@desy.de
- ben.brueers@desy.de
- nils.gillwald@desy.de

The FH Sustainability Forum and FH IT experts present a 2-day workshop on Computing. In this workshop, participants will learn to write efficient code, use batch computing in a sustainable way, and more. Best practices will be taught by experts. This workshop is targeted towards students and postdocs, but open to anyone who feels like they would benefit from refreshing their memory or learning current best practices. Participants will learn how to develop more efficient code and how to maximize resources and minimize waste, with hands-on examples. These examples will be demonstrated with local computing clusters and are tailored to the needs of the FH division.

The fourth installment of this workshop will occur on October 7 and 8, 2024, in person on the Hamburg campus (with some zoom available). This fourth installment is targeted towards beginners. We plan to offer this workshop at regular intervals in the future, so that incoming students and postdocs can profit. The next workshop will target more advanced users.

Coffee breaks will be provided free of charge. Lunch is not included.

Zoom connection:
https://cern.zoom.us/j/62785538071?pwd=dXJFZndHK0g4eVNaelArSjVjaUFQZz09

Mattermost channel: https://chat.desy.de/desy/channels/fh-sustainable-computing-workshop

Live notes for last minute updates:

Gitlab repository for the workshop: https://gitlab.desy.de/fh-sustainability-forum/sustainable-coding-tutorial

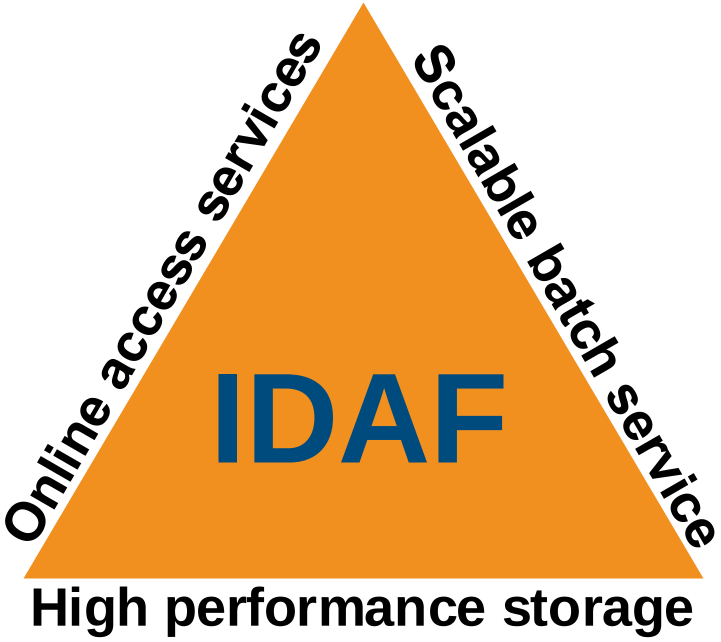
https://indico.desy.de/event/43906/

# What is an Analysis Facility?

# What is an Analysis Facility?

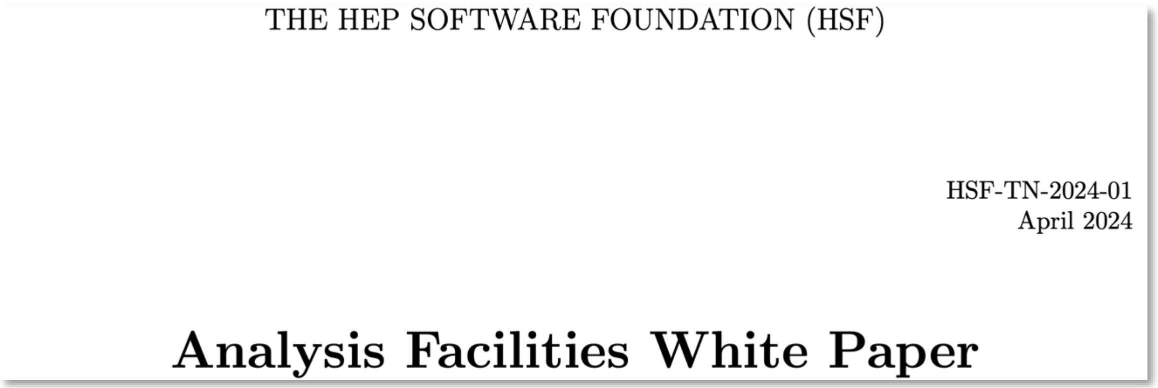
- Basic concept of the IDAF are:
  - Data locality
  - Access services and compute integrated with storage
  - Present a holistic service to the analyst

- With the advent of machine learning and new compute technologies and storage concepts:
  - Does data locality still hold?
  - Is current storage integration still OK?
  - Do people need an integrated service, or rather flexible infrastructure?



# Discussions going on at different levels

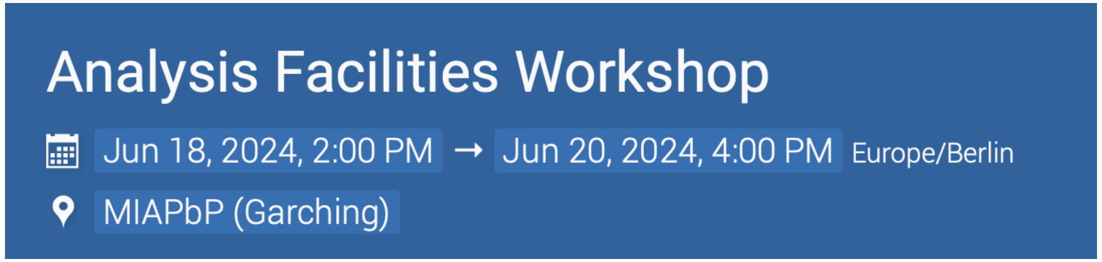
- Sometimes user driven … who drives?
  - 10% pioneer users?
  - 10% special requirements users?
  - 80% normal users?
- Research at facilities needed:
  - E.g. PhD students / young postdocs that spend some time doing their analysis in a novel way – in close interaction with IDAF experts
- Feedback to IDAF at all levels: Do not name technology: name functionality, describe user stories



THE HEP SOFTWARE FOUNDATION (HSF)

HSF-TN-2024-01
April 2024

Analysis Facilities White Paper



Analysis Facilities Workshop

Jun 18, 2024, 2:00 PM → Jun 20, 2024, 4:00 PM Europe/Berlin

MIAPbP (Garching)

# Recent developments: EL9, MFA

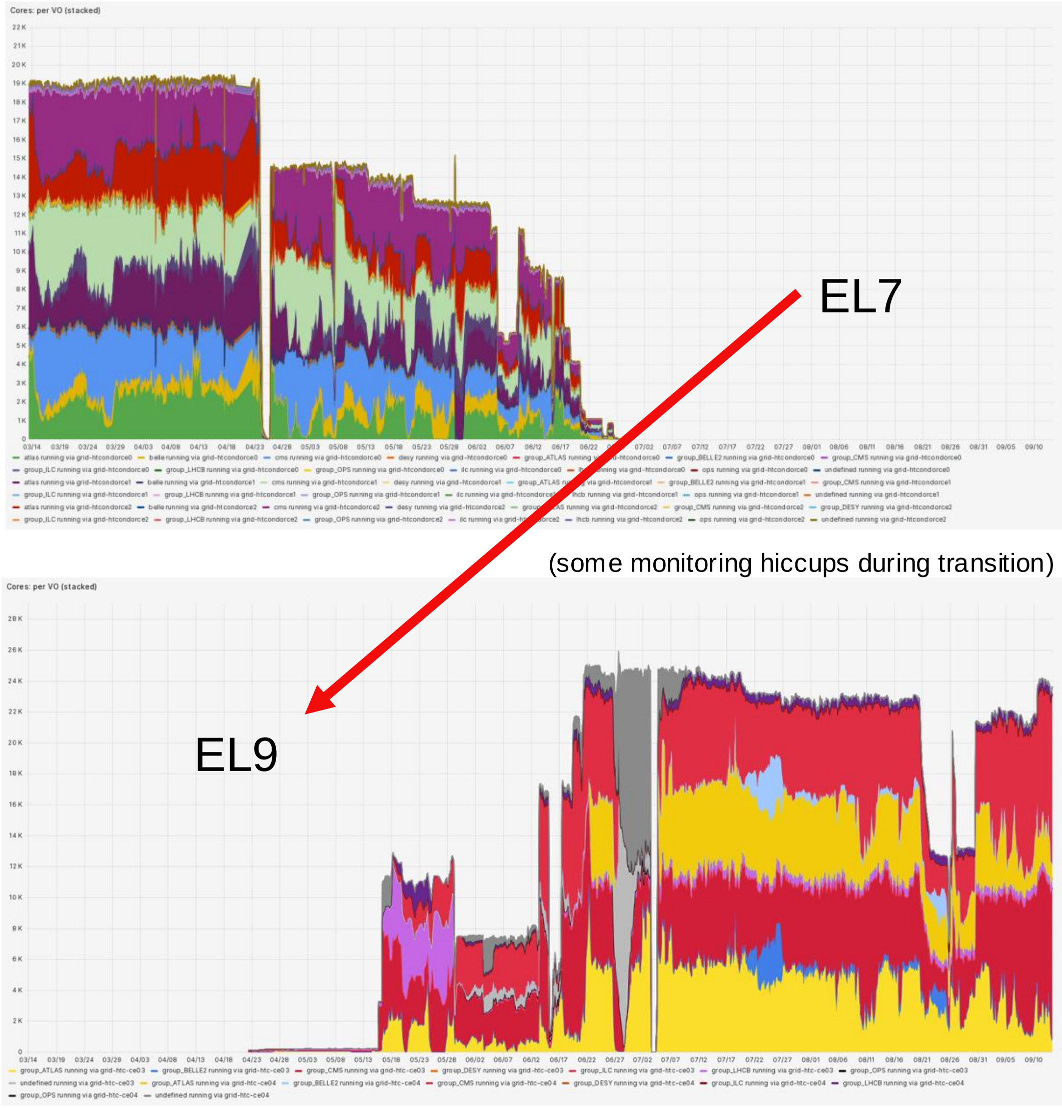
- Migration away from CentOS 7 (EOL 30.6.24)
- Opting for RHEL 9 (because of RHEL flavour, and RHEL campus contract incl. support)
- First test systems in March
- Ramp up in June
- Final migration mid July

- Secure accounts against loss of passwords:
- Mulit Factor Authentication (MFA) since Autumn ‘23
  - Ssh, Jupyter, FastX
  - (Mail, Sync’n’Share, ….)
- More to come (but mostly of interest for DESY people)

# Grid @ DESY-HH

- HTCondor Cluster
  - ~25k Cores, ~402 kHS23
    (mix HS06 mapped and HS23 natively measured)
  - Started now to cull older, inefficient nodes
  - Recycled old nodes into dedicated preprod cluster
- Rolling migration to Condor23 & CondorCE6 on RHEL9
  - Reinstalled last remaining set of EL7 nodes with EOL
  - EL9 migration worked reasonably
    - Some minor remaining elements still have to migrated
      (py2.7, cgroups v1,... monitoring scripts etc.) :-/
- Getting non-token ready groups onto the cluster was/is a PITA
  - SSL authz working, but expertise on the user/group site limited

## HTC Clusters at DESY-HH



(some monitoring hiccups during transition)

# Belle II Grid Resources @ DESY

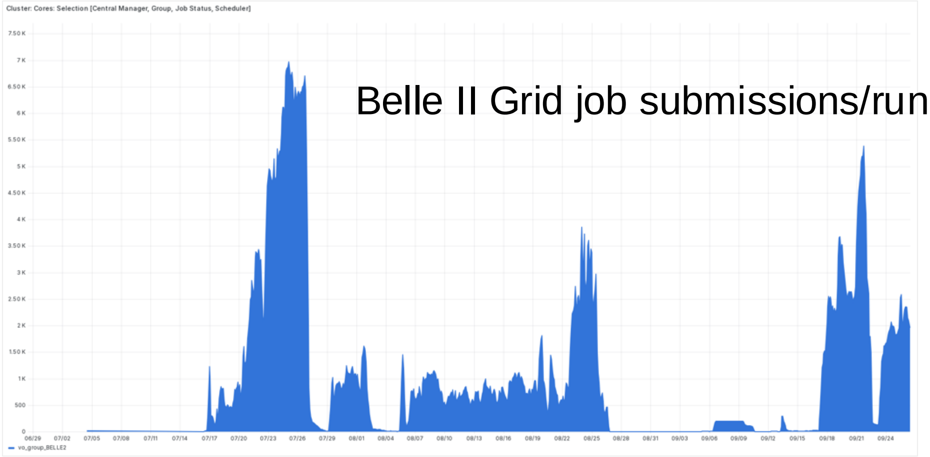
## Belle II Grid Resources @ DESY

| | |
|---|---|
| CPU pledges: | 31.47kHS23 (Apr 2024) |
| Usage: | 22kHS23 (Jan-May 2024) |
| Contribution: | ~20% of total Belle II Grid computing |
| Disk installed: | 2305TB (incl. RAW write pools) |
| Disk pledges: | 2040TB (47% of German fraction) |
| Disk usage: | 1564TB |
| Tape usage: | 173TB tape |

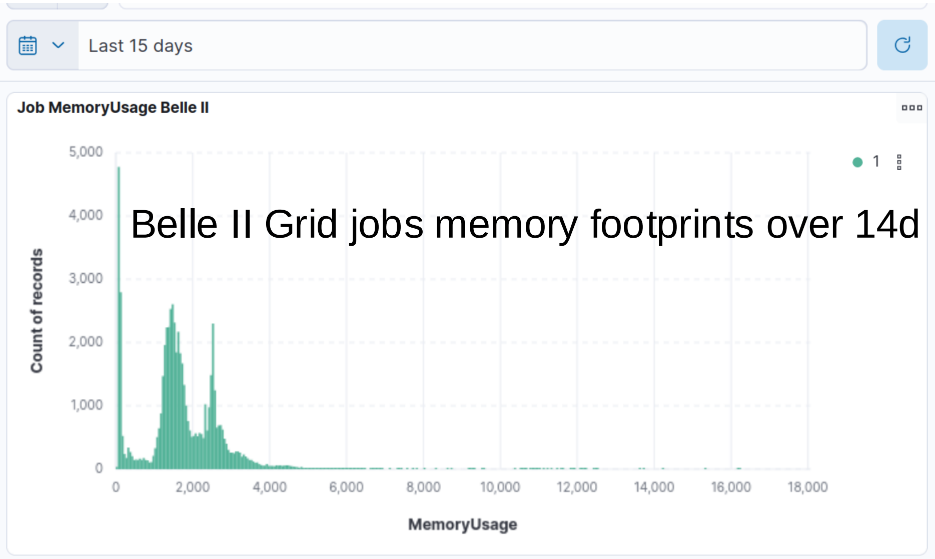
# Status Belle II

## Belle II Grid Production

- CondorCE authz via legacy SSL user/robot DN
  - Status token workflows on factory side?
- Utilization of grid cores below nominal share
  - Local or upstream issue?
  - Steady queue pressure desirable
- Memory footprints seem to leak beyond 2GB/core
  - Plans for mcores or more dynamic core & mem scheduling?
  - VOCard still valid?



Belle II Grid job submissions/runs


Last 15 days

Job MemoryUsage Belle II

Belle II Grid jobs memory footprints over 14d

# Grid @ DESY-HH

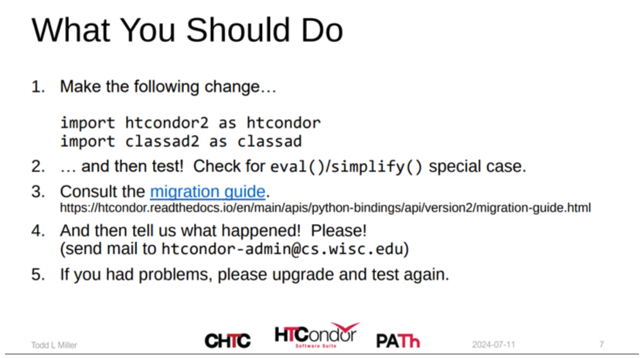
## Auxilliary Grid Services

- Accounting borked on EL9
  - No working AUDITOR or APEL in production yet :-/
- Authz services at us ~ VOMS
  - still EL7 kept alive for legacy users
    - EL9 package only recently released
  - IAM probably way to go (but support backing unclear)
    - limited manpower ~ no time yet
- Supporting legacy services becomes prohibitive
  - BDII, SRM,… only kept alive for EGI
    - support existing & long-term reliable?
    - dropping support for such legacy services and protocols

# What’s new as well

## (upcoming) Changes in the (Grid-related) World

- HTCondor Python bindings got version 2
- Auxiliary Condor CLI interface growing
  - `htcondor noun verb`
- cgroup v2 resource control in production
  - Fixes upcoming to current issues
  - More aggressive memory control desirable
  - Device cgroup controller wrt GPUs available
- LVM-based job home isolation/volumes
  - would enforce strong disk requests/limits
- with Condor 24 no old style job routes more supported

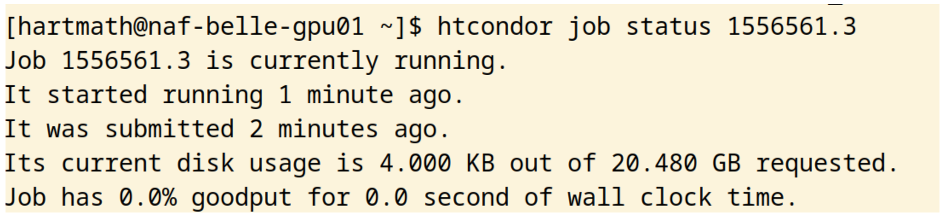


### What You Should Do

1. Make the following change…

   ```
   import htcondor2 as htcondor
   import classad2 as classad
   ```
2. … and then test! Check for `eval()/simplify()` special case.
3. Consult the migration guide.
   https://htcondor.readthedocs.io/en/main/apis/python-bindings/api/version2/migration-guide.html
4. And then tell us what happened! Please!
   (send mail to `htcondor-admin@cs.wisc.edu`)
5. If you had problems, please upgrade and test again.

Todd L Miller 2024-07-11 7



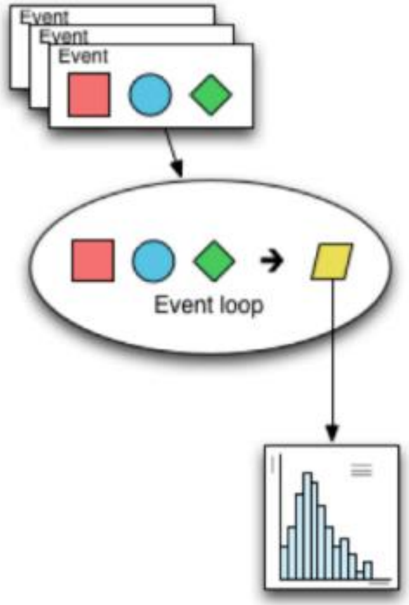
```
[hartmath@naf-belle-gpu01 ~]$ htcondor job status 1556561.3
Job 1556561.3 is currently running.
It started running 1 minute ago.
It was submitted 2 minutes ago.
Its current disk usage is 4.000 KB out of 20.480 GB requested.
Job has 0.0% goodput for 0.0 second of wall clock time.
```

# Columnar Analysis

Illustrations by Nick Smith

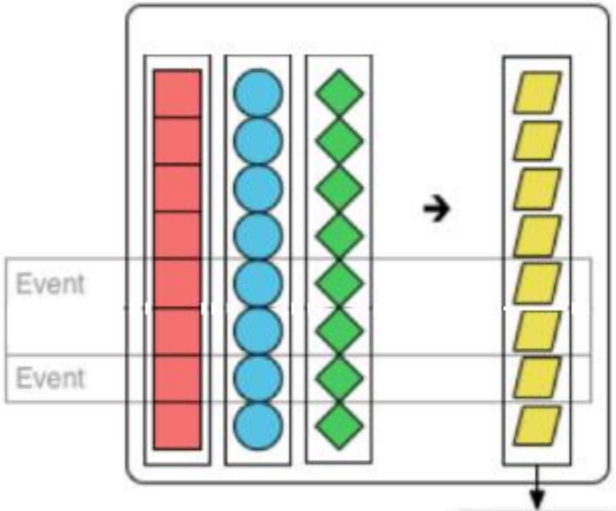
## A Different Paradigm for Data Analysis – But is it really?

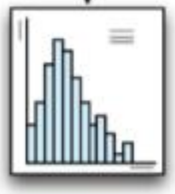
### Event Loop



- Typically Object based data ➜ Particle with associated track
- Read data from event into local variables
- Calculate and store derived variables
- Rinse and repeat for all entries
- Explicit outer loop

### Columnar Structure



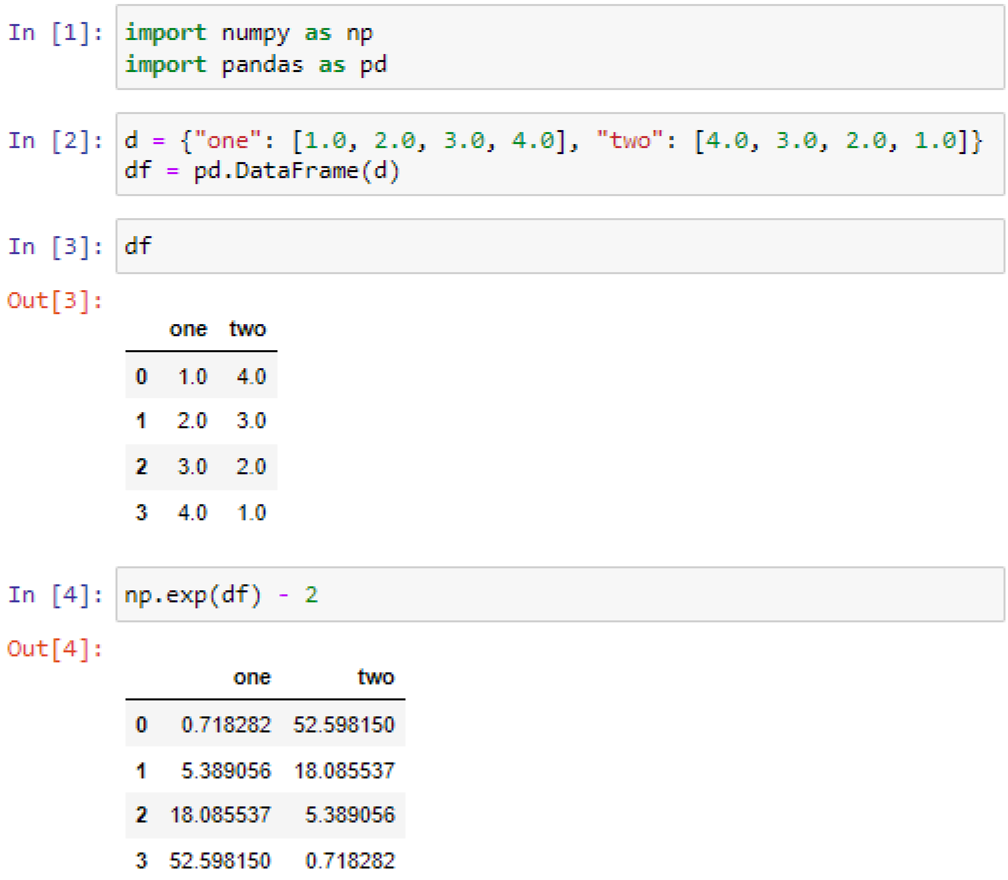


- Load data into individual arrays
- Nested structure (arrays of arrays)
- Perform calculation on all elements in array
- Implicit inner loop
- Data structure in TTree and Pandas dataframe
- **Advantageous with modern CPU architectures**

# Columnar Analysis

## Simple Example and Scale Out

- Columnar data structure default in Pandas dataframes
- Inspired by long legacy of relational databases in computer science ⍰
- Core element in `numpy` and `scipy` present in most python based data analyses ⍰
- Memory handling problematic import of full dataset
- Observed issues on NAF of Jupyter notebooks using ~20GiB RAM



```
In [1]: import numpy as np
        import pandas as pd

In [2]: d = {"one": [1.0, 2.0, 3.0, 4.0], "two": [4.0, 3.0, 2.0, 1.0]}
        df = pd.DataFrame(d)

In [3]: df

Out[3]:
           one  two
        0  1.0  4.0
        1  2.0  3.0
        2  3.0  2.0
        3  4.0  1.0

In [4]: np.exp(df) - 2

Out[4]:
                 one        two
        0   0.718282  52.598150
        1   5.389056  18.085537
        2  18.085537   5.389056
        3  52.598150   0.718282
```

### External Analysis Frameworks for Scale-Out

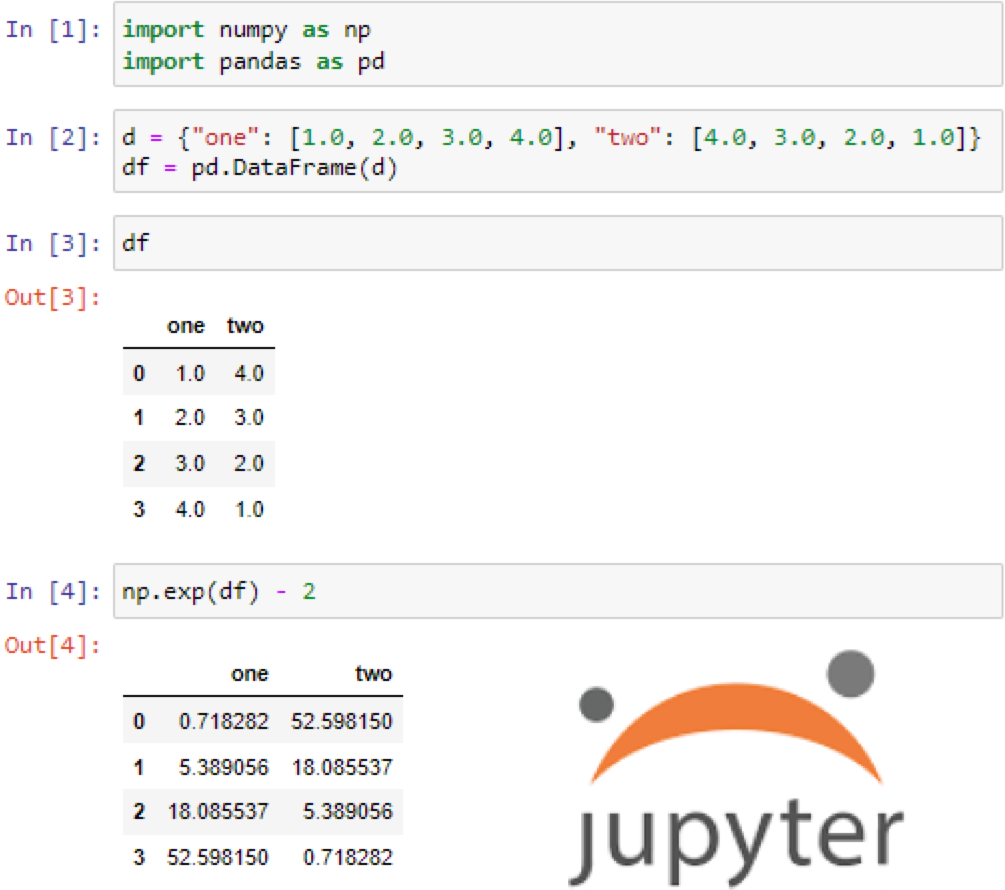
- ROOT RDataframe
- Apache Spark
- Python Dask
- Integration of the tools above : Coffea/Casa or CERN Swan

# Jupyter Notebooks

## Need for Scale-Out

### Why to Talk about Scale-Out:

- Use Jupyter notebooks as convenient way to analyse data interactively: **integration into NAF/Batch**
- Memory-handling of numpy and pandas often suboptimal
  ➜ **large datasets being loaded completely into memory**
- Makes deployment tricky:
  - On the Workgroup-Servers a single user can take the whole node offline by reading to many events
  - In our setup: Jupyter slots are limited by CPU and RAM
    ➜ Loading too many events will cause your job to be killed
  - Increase the amount of RAM ➜ makes scheduling more inefficient
  - Look at external tools for scale-out of CPUs, memory, disk-I/O, and network



```
In [1]: import numpy as np
        import pandas as pd

In [2]: d = {"one": [1.0, 2.0, 3.0, 4.0], "two": [4.0, 3.0, 2.0, 1.0]}
        df = pd.DataFrame(d)

In [3]: df

Out[3]:
   one  two
0  1.0  4.0
1  2.0  3.0
2  3.0  2.0
3  4.0  1.0

In [4]: np.exp(df) - 2

Out[4]:
         one        two
0   0.718282  52.598150
1   5.389056  18.085537
2  18.085537   5.389056
3  52.598150   0.718282
```

# Columnar Analysis Tool-Kits

## Scale-out Option Similar to Pandas

- Single Node Jupyter Notebooks are limited in resources and Python by default not clever at data import
- Ease of use of Pandas et al. further escalate problem
- Investigate scale-out tools, most commonly used: Apache Spark and Python Dask

### Apache Spark

- Written in Java; industry tool
- Different syntax than Pandas
- Not Python bound → easier to deploy/maintain
- Extensive data-caching in memory possible
- Easy to plug external data sources and data types (e.g. PostgreSQL import, ROOT data-types)
- Limited support for HEP data formats

### Python Dask

- Pure pythonic; developed in HPC community
- Very similar syntax than Pandas
- Pure Python → tedious to deploy/maintain (versioning)
- Extensive data-caching in memory possible
- Support for HEP data formats (through uproot/awkward arrays)
- Supports only numpy compatible data-types

- Community tool-boxes build around these tools → Coffea/Casa
- ROOT recent addition RDataframe can be connected to Dask/Spark setups (CERN Swan)

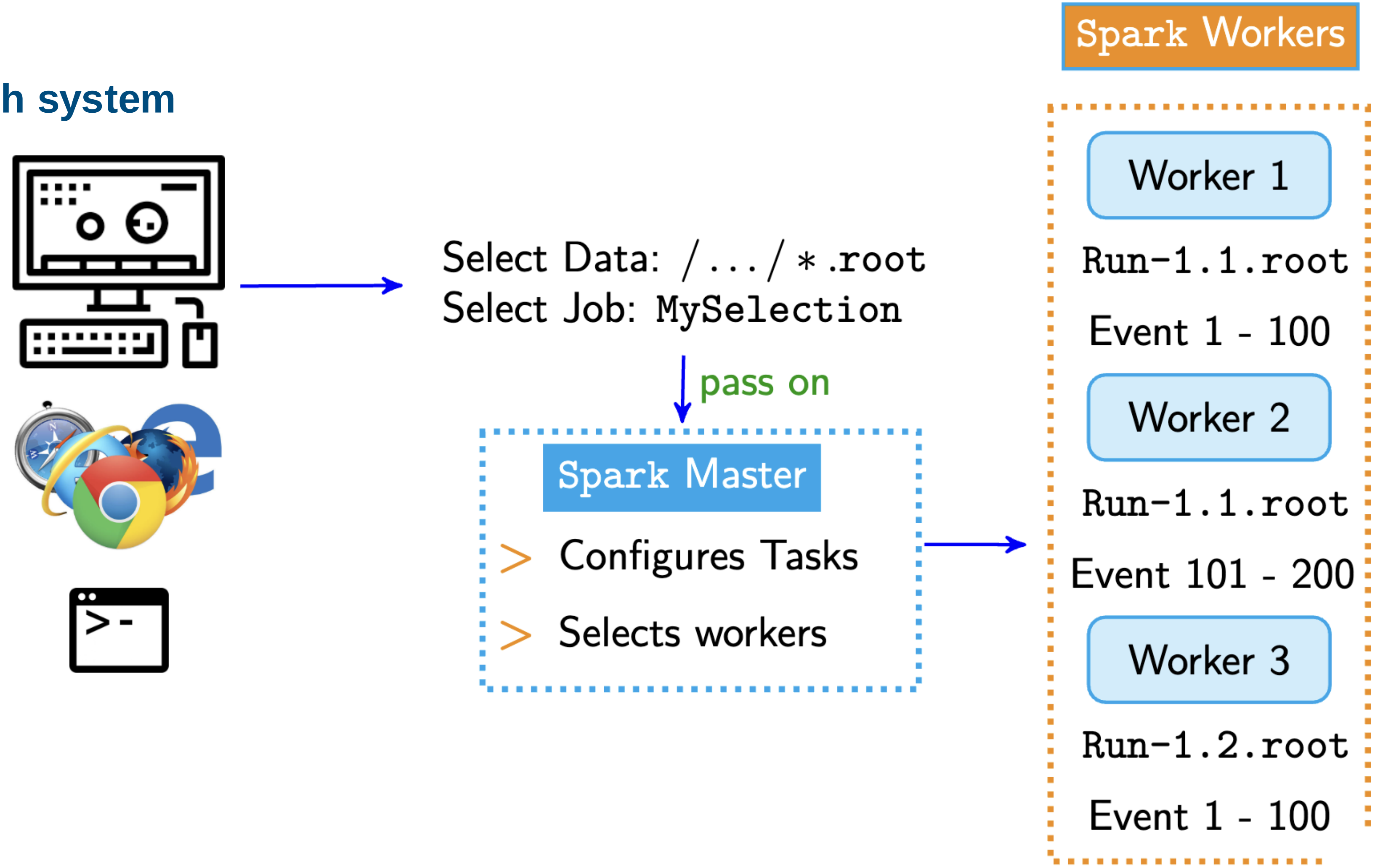
# Apache Spark and Python Dask

## Architecture of the systems

- Some might say, **just another batch system**

### Basic Layout

- There is a master and there are worker
- User controls an application on the cluster communicating with the master
- Master deploys the workloads over the cluster of workers
- Ideally with efficient access to the storage
- Possibility for extended data caching
- Master must be known by all workers
  ➜ makes a deployment in the NAF trickier



Select Data: /.../*.root
Select Job: MySelection

pass on

Spark Master

> Configures Tasks

> Selects workers

Spark Workers

Worker 1
Run-1.1.root
Event 1 - 100

Worker 2
Run-1.1.root
Event 101 - 200

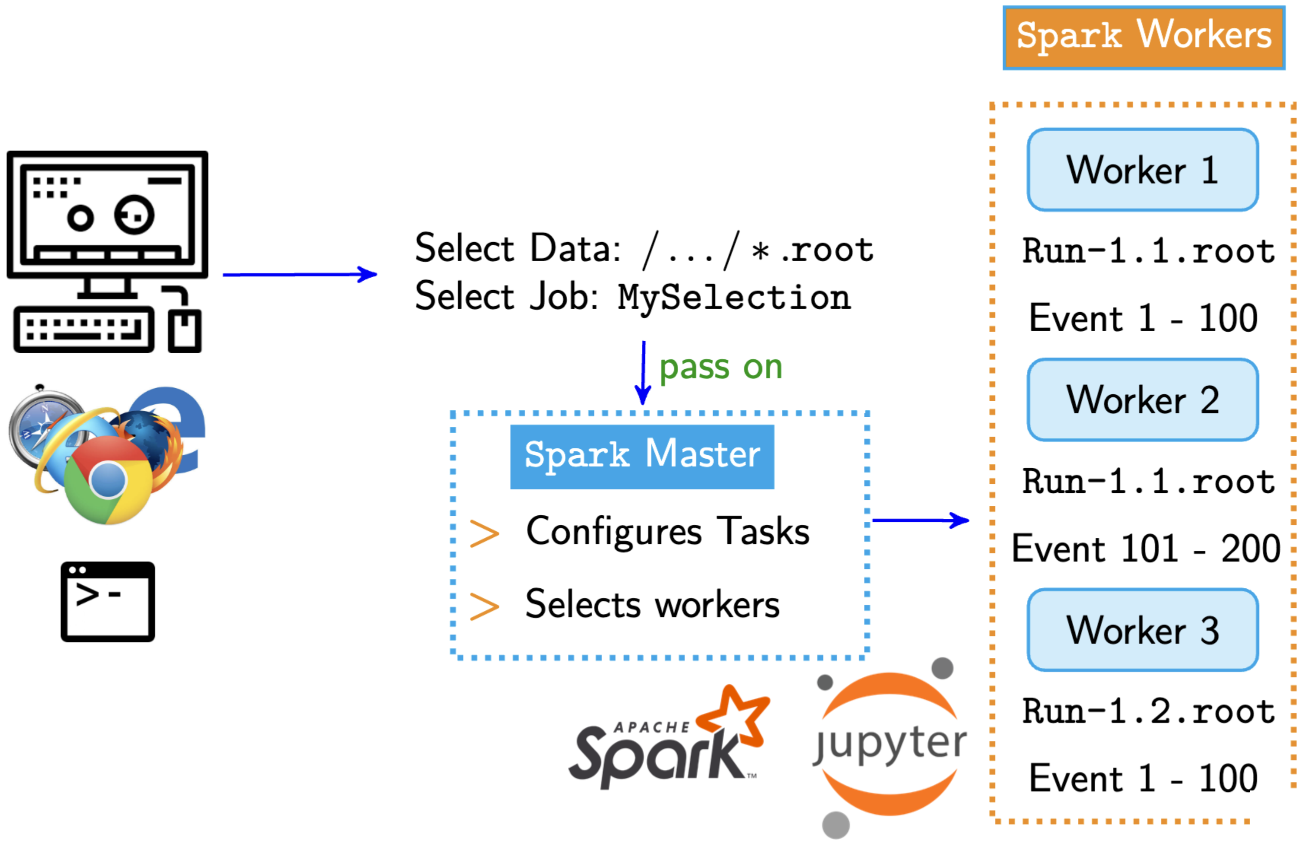
Worker 3
Run-1.2.root
Event 1 - 100

# Toolsets at Work

## Running Apache Spark on NAF

### Apache Spark on National Analysis Facility:

- IT uses Spark for log-file, transfer, and database analyses
- Deployment manual: Master on WGS; Worker on NAF
- Worker are containerised and deploys as Condor jobs
- Import data from /pnfs using NFS
- Analysis of up to 20TiB worth of data on NAF
- Use memory caching extensively



Spark Workers

Select Data: /.../*.root
Select Job: MySelection

pass on

Spark Master

> Configures Tasks

> Selects workers

Worker 1

Run-1.1.root

Event 1 - 100

Worker 2

Run-1.1.root

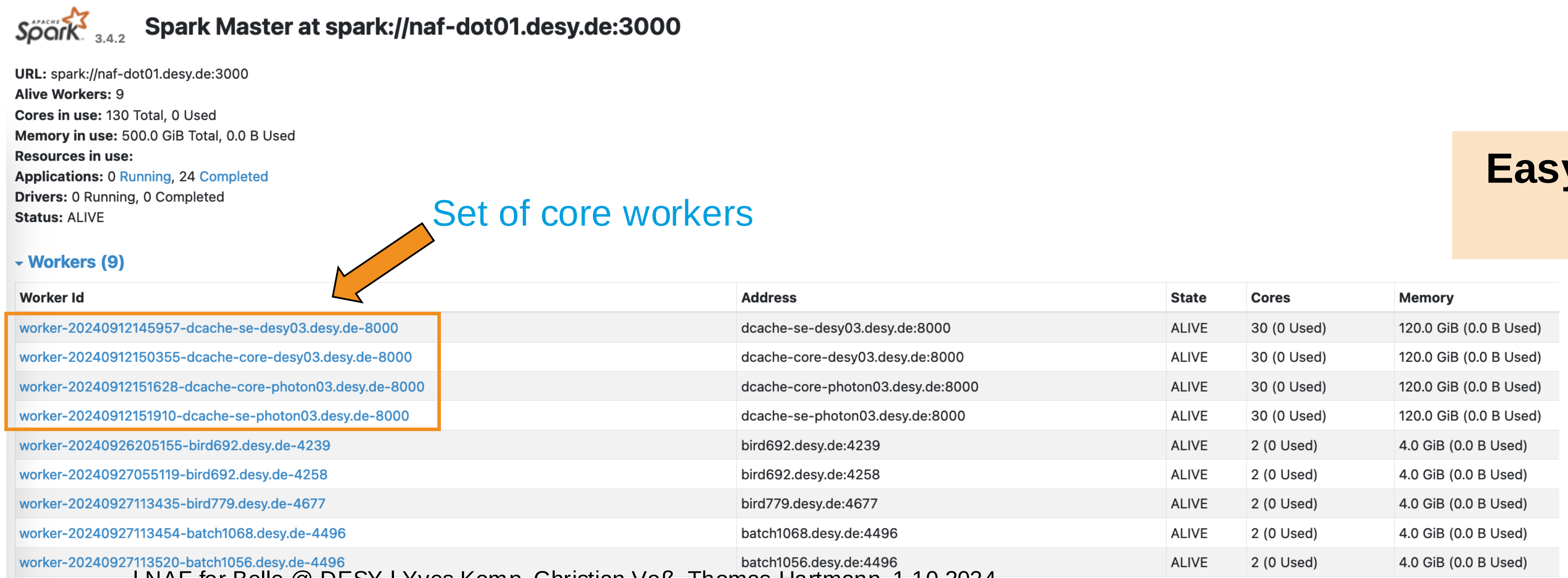
Event 101 - 200

Worker 3

Run-1.2.root

Event 1 - 100

**Easy to Deploy a Similar Setup for Python Dask**



**Spark Master at spark://naf-dot01.desy.de:3000**

**URL:** spark://naf-dot01.desy.de:3000
**Alive Workers:** 9
**Cores in use:** 130 Total, 0 Used
**Memory in use:** 500.0 GiB Total, 0.0 B Used
**Resources in use:**
**Applications:** 0 Running, 24 Completed
**Drivers:** 0 Running, 0 Completed
**Status:** ALIVE

Set of core workers

**Workers (9)**

| Worker Id | Address | State | Cores | Memory |
|---|---|---|---|---|
| worker-20240912145957-dcache-se-desy03.desy.de-8000 | dcache-se-desy03.desy.de:8000 | ALIVE | 30 (0 Used) | 120.0 GiB (0.0 B Used) |
| worker-20240912150355-dcache-core-desy03.desy.de-8000 | dcache-core-desy03.desy.de:8000 | ALIVE | 30 (0 Used) | 120.0 GiB (0.0 B Used) |
| worker-20240912151628-dcache-core-photon03.desy.de-8000 | dcache-core-photon03.desy.de:8000 | ALIVE | 30 (0 Used) | 120.0 GiB (0.0 B Used) |
| worker-20240912151910-dcache-se-photon03.desy.de-8000 | dcache-se-photon03.desy.de:8000 | ALIVE | 30 (0 Used) | 120.0 GiB (0.0 B Used) |
| worker-20240926205155-bird692.desy.de-4239 | bird692.desy.de:4239 | ALIVE | 2 (0 Used) | 4.0 GiB (0.0 B Used) |
| worker-20240927055119-bird692.desy.de-4258 | bird692.desy.de:4258 | ALIVE | 2 (0 Used) | 4.0 GiB (0.0 B Used) |
| worker-20240927113435-bird779.desy.de-4677 | bird779.desy.de:4677 | ALIVE | 2 (0 Used) | 4.0 GiB (0.0 B Used) |
| worker-20240927113454-batch1068.desy.de-4496 | batch1068.desy.de:4496 | ALIVE | 2 (0 Used) | 4.0 GiB (0.0 B Used) |
| worker-20240927113520-batch1056.desy.de-4496 | batch1056.desy.de:4496 | ALIVE | 2 (0 Used) | 4.0 GiB (0.0 B Used) |

Scale-out using NAF light jobs

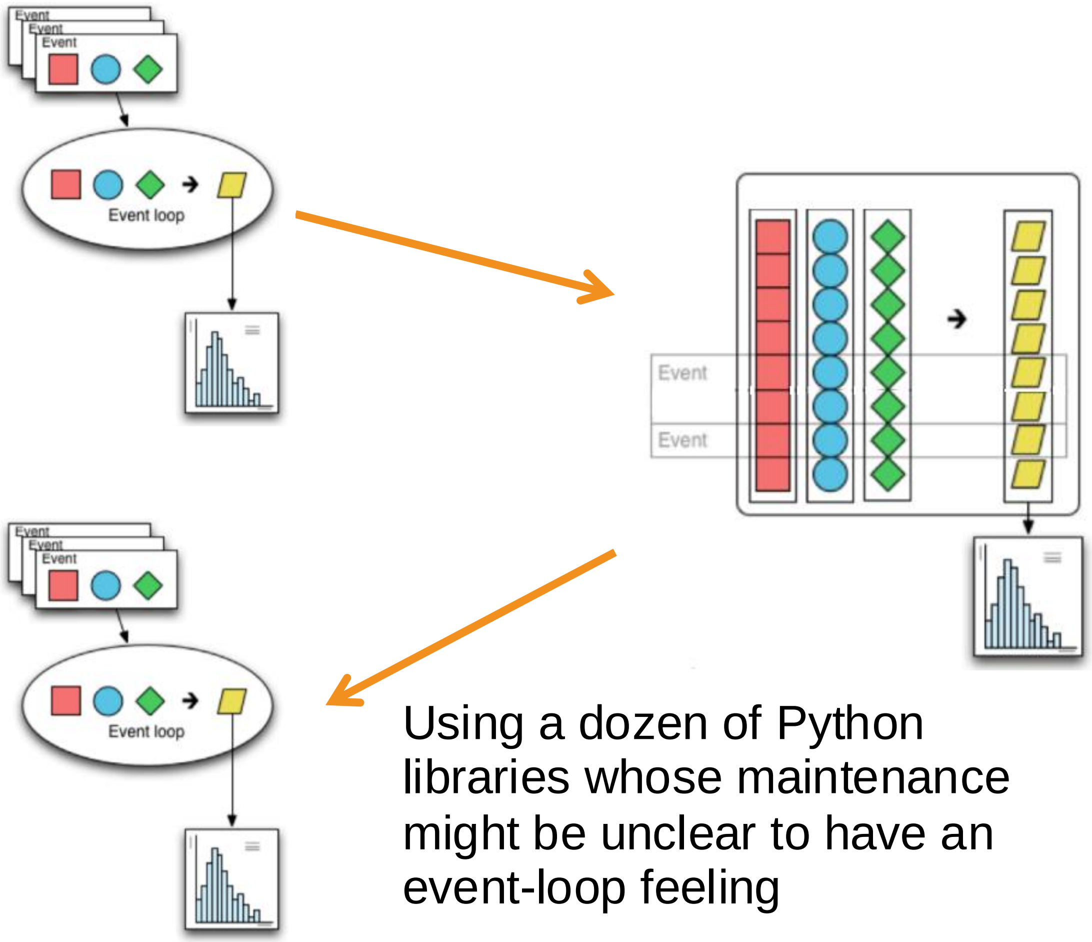
# Taking a Step Back

## Putting Analysis Tools in to the Wider Picture

- In General: we’re happy to embark on another attempt to further test and deploy either Spark or Dask

### Refocus and leave the HEP Ivory Tower

- Our students are now growing up using Python based data analysis tools: bachelor and master student is typically quite familiar with using numpy and pandas
- In their later careers they will keep using numpy and pandas
- Industry will keep advancing and we would to well to join them
- ROOT and our HEP specific protocols become a hindrance
- Lucky on NAF with our NFS support (Spark can’t speak XrootD)
- Might want to refocus also our input format ➜ convert ROOT files to e.g. Parquet files
- Reconsider the hole uproot/awkward array and either do event loops or fully embrace columnar analyses
- Forcing columnar workflows to feel like event loops will not help our field ➜ students might better served with Photon Science



Using a dozen of Python libraries whose maintenance might be unclear to have an event-loop feeling