Data challenges in scientific computing

Physics is overrated

João Alvim, Anton Schwarz, Konrad Kockler
DESY Hamburg, 04.09.2024

HELMHOLTZ

Scientific Data Challenges

Data processing

Ingest

- High data ingest rate
- Multiple parallel streams
- High durability
- Effective handling of large number of files

Sharing & Exchange

- 3rd party copy
- Effective WAN Access
- In-flight data protection
- Identity federation
- Access control

Long Term Preservation

- High Reliability
- Self-healing
- Automatic technology migration
- Persistent identifier

Analysis

- High CPU efficiency
- Unstructured access patterns
- Standard access protocols
- Access control
- Local user management

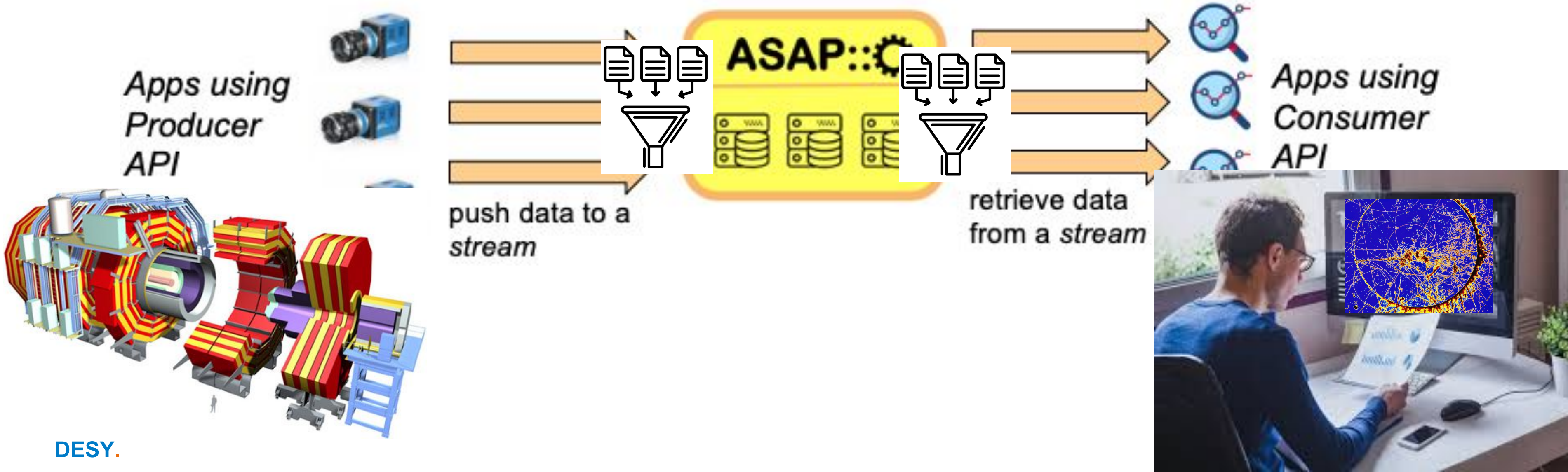
Improving ASAP::O Monitoring

João Alvim, Mikhail Karnevskiy

ASAP::O

What is it?

1. **High performance distributed streaming platform**
2. **Provides API to ingest/retrieve data to the system**

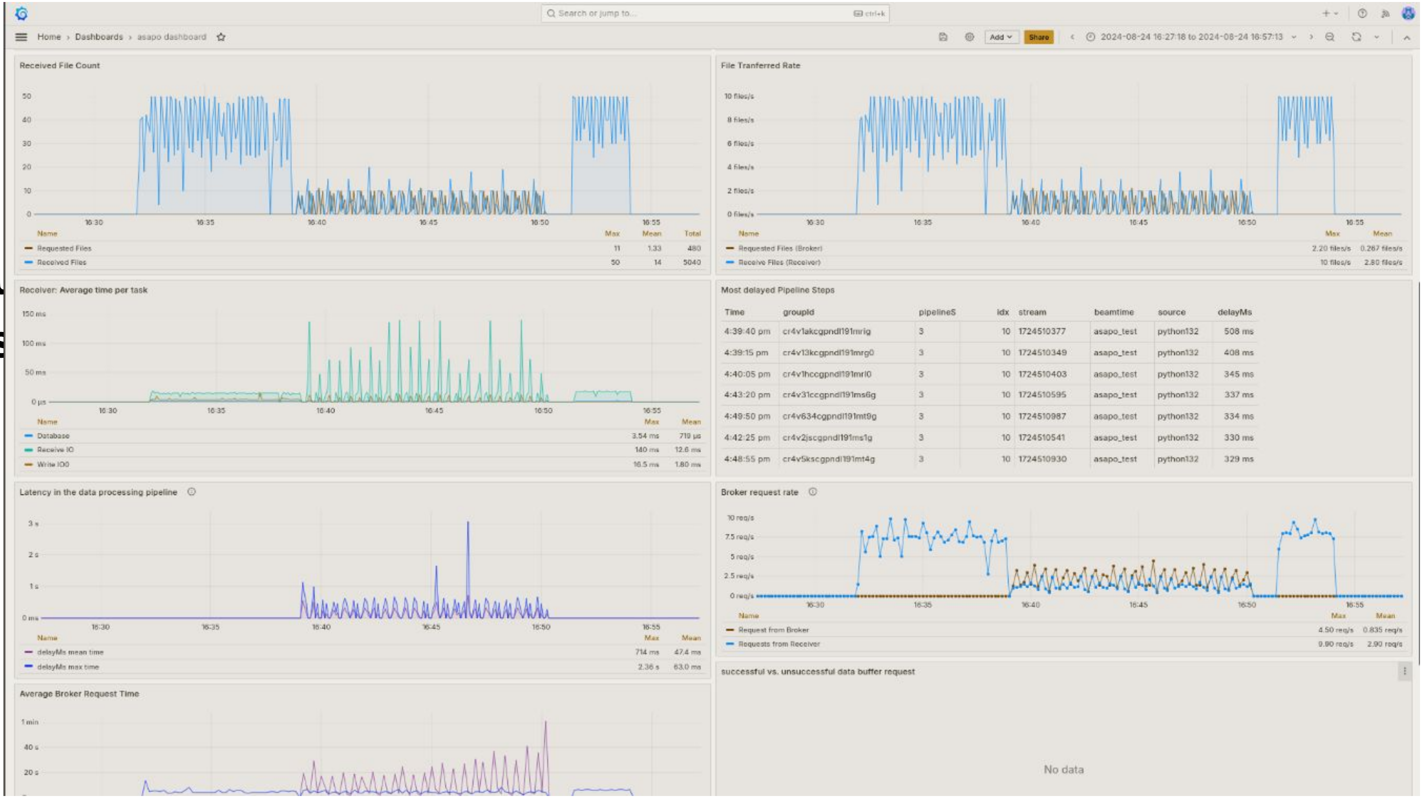


Monitoring Improvements

New Approach

1. Grafana
+ flexibility, bu
  components
+ reliable
- bugs

2. Collect new



Search or jump to...
ctrl+k
Home › Dashboards › asapo dashboard
Add
Share
2024-08-24 16:27:18 to 2024-08-24 16:57:13

Received File Count

Name	Max	Mean	Total
Requested Files	11	1.33	480
Received Files	50	14	5040

File Tranferred Rate

Name	Max	Mean
Requested Files (Broker)	2.20 files/s	0.267 files/s
Receive Files (Receiver)	10 files/s	2.80 files/s

Receiver: Average time per task

Name	Max	Mean
Database	3.54 ms	719 µs
Receive IO	140 ms	12.6 ms
Write IO0	16.5 ms	1.80 ms

Most delayed Pipeline Steps

Time	groupId	pipelineS	idx	stream	beamtime	source	delayMs
4:39:40 pm	cr4v1akcgpndl191mrig	3	10	1724510377	asapo_test	python132	508 ms
4:39:15 pm	cr4v13kcgpndl191mrg0	3	10	1724510349	asapo_test	python132	408 ms
4:40:05 pm	cr4v1hccgpndl191mrl0	3	10	1724510403	asapo_test	python132	345 ms
4:43:20 pm	cr4v31ccgpndl191ms6g	3	10	1724510595	asapo_test	python132	337 ms
4:49:50 pm	cr4v634cgpndl191mt9g	3	10	1724510987	asapo_test	python132	334 ms
4:42:25 pm	cr4v2jscgpndl191ms1g	3	10	1724510541	asapo_test	python132	330 ms
4:48:55 pm	cr4v5kscgpndl191mt4g	3	10	1724510930	asapo_test	python132	329 ms

Latency in the data processing pipeline

Name	Max	Mean
delayMs mean time	714 ms	47.4 ms
delayMs max time	2.36 s	63.0 ms

Broker request rate

Name	Max	Mean
Request from Broker	4.50 req/s	0.835 req/s
Requests from Receiver	9.90 req/s	2.90 req/s

Average Broker Request Time

successful vs. unsuccessful data buffer request

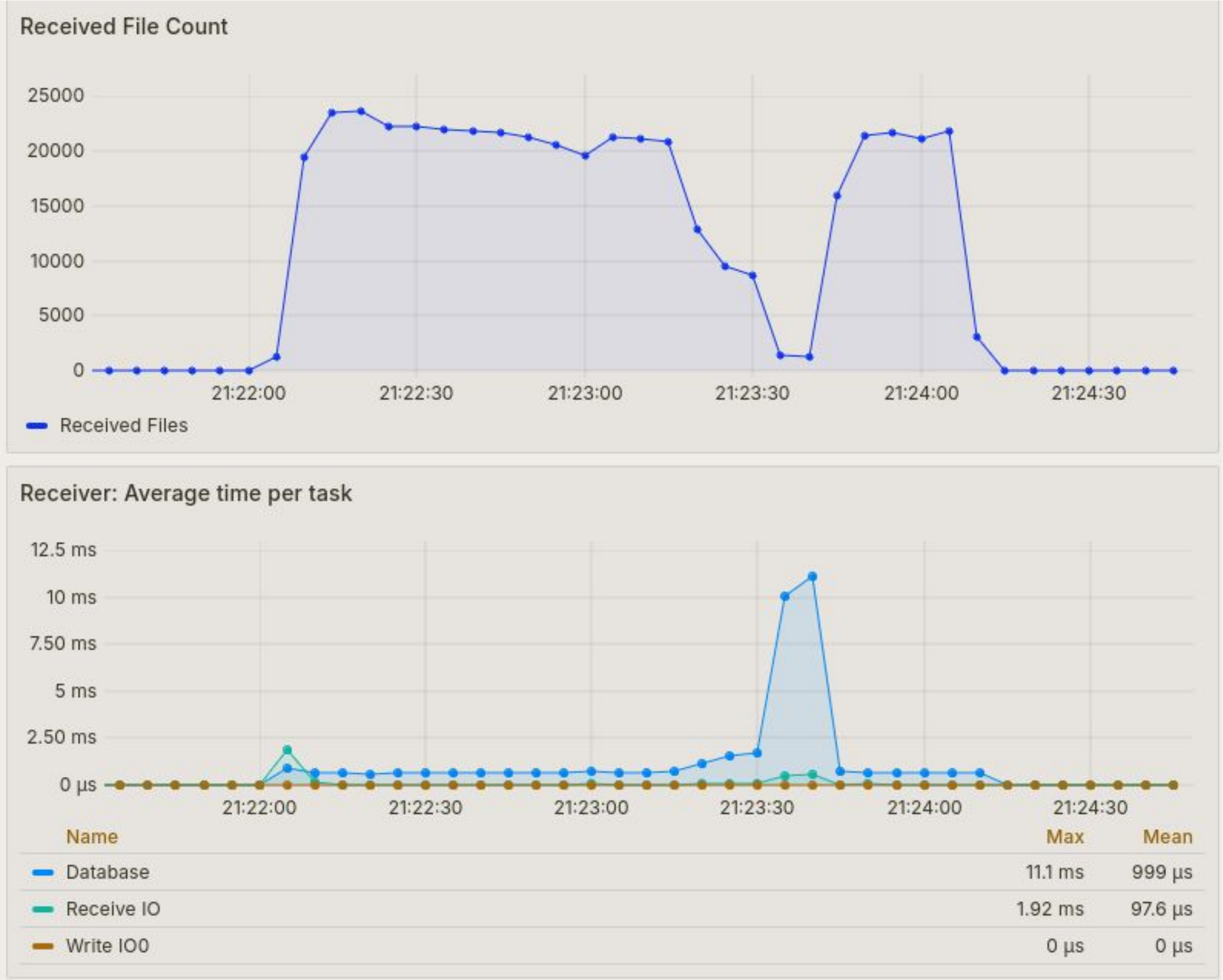
No data

Example case

Amount of received files dropped significantly, because the time to access database increased

Raise Suspicion Potential Bottleneck

=> More Reliable Software



Received File Count

Receiver: Average time per task

Name	Max	Mean
Database	11.1 ms	999 µs
Receive IO	1.92 ms	97.6 µs
Write IO0	0 µs	0 µs

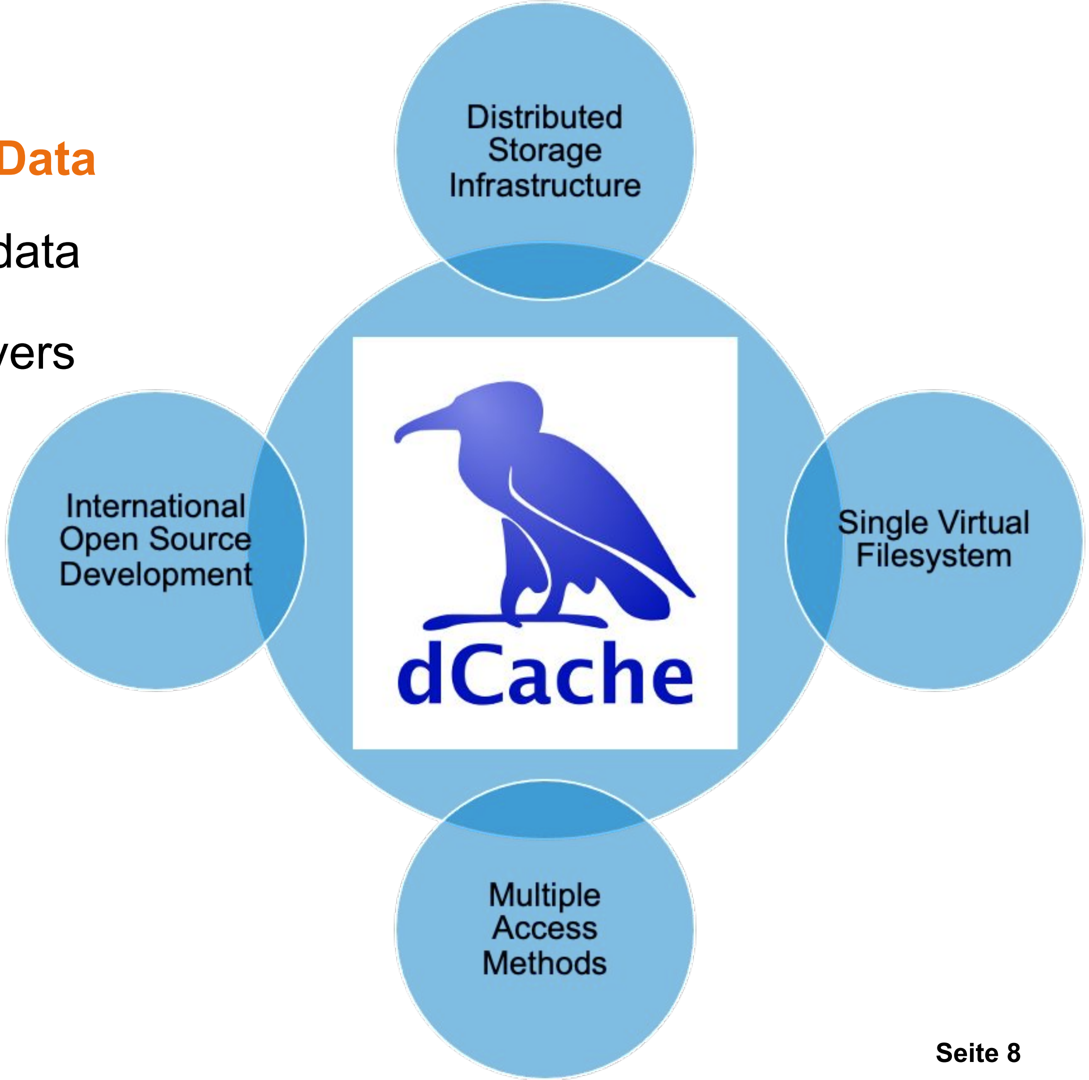
Scientific Data Management with dCache

Anton Schwarz, Tigran Mkrtchyan

dCache

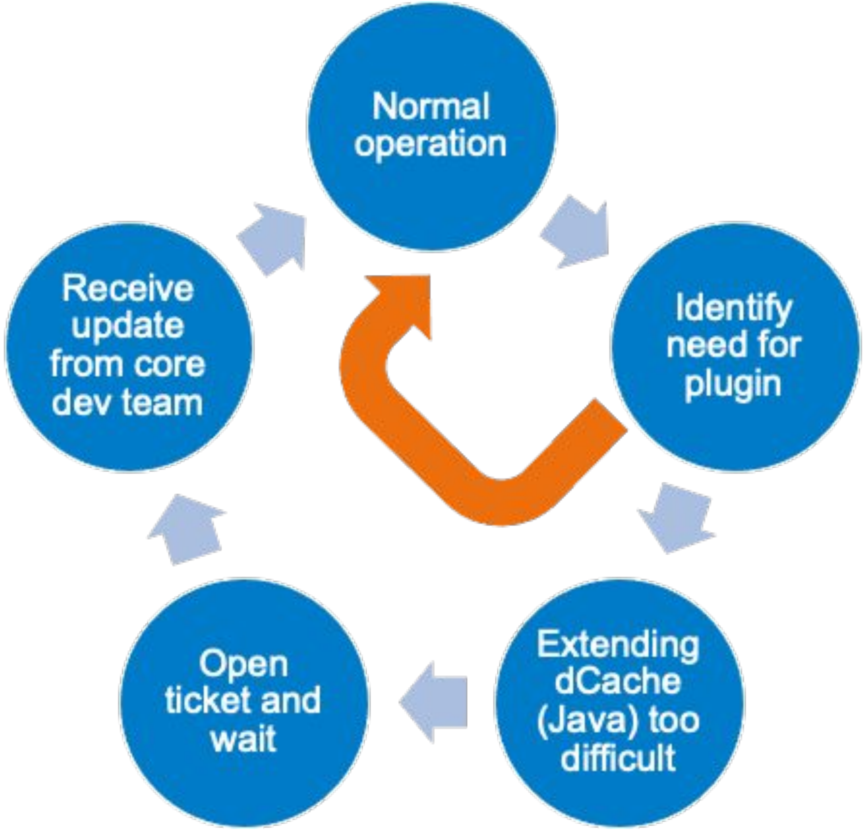
A System to Store and Retrieve Data

- System for storing and retrieving data
- Data is distributed over many servers
- In full production since 2001
- Written in Java
- Authentication and Authorisation


Distributed Storage Infrastructure
International Open Source Development
dCache
Single Virtual Filesystem
Multiple Access Methods

Adapting the Authentication Process

Motivation for my project



Normal operation

Identify need for plugin

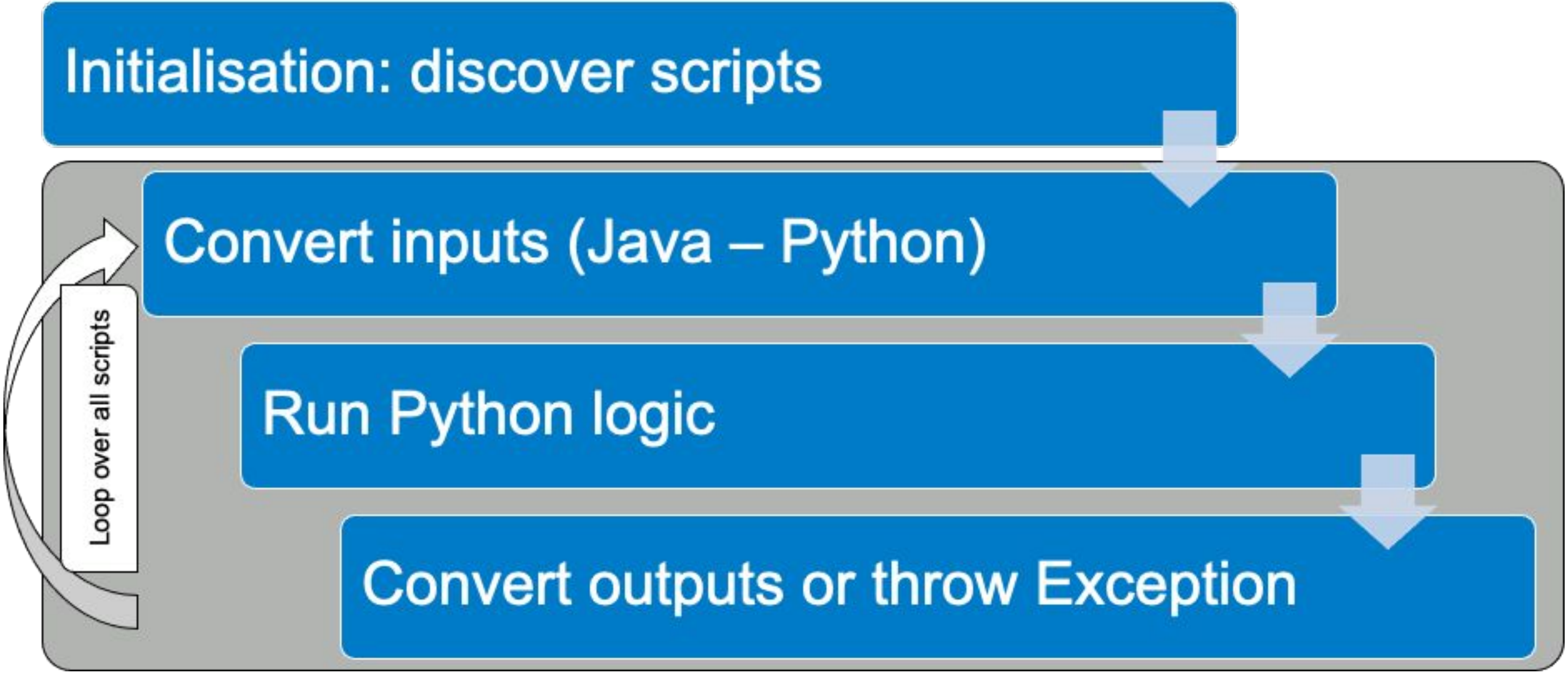
Extending dCache (Java) too difficult

Open ticket and wait

Receive update from core dev team

Allow users to extend the authentication process with their own plugins

Implementation



Initialisation: discover scripts

Convert inputs (Java – Python)

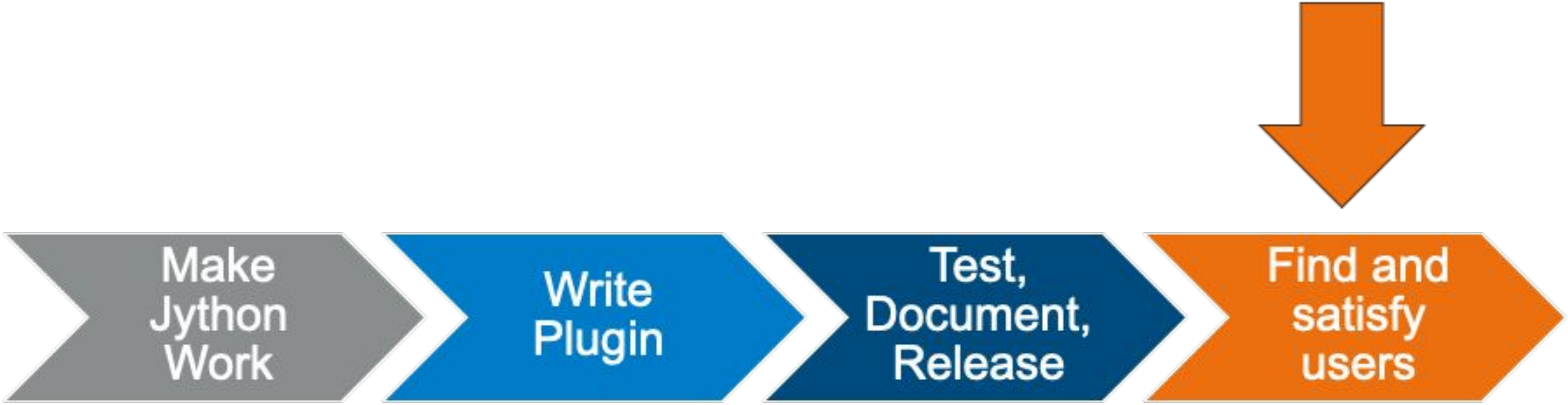
Run Python logic

Convert outputs or throw Exception

Loop over all scripts

Timeline

Where are we?



Research facility 2.0
Sustainable computing

Konrad Kockler, Martin Gasthuber

Current energy consumption

Per year (2023):

12.9GWh

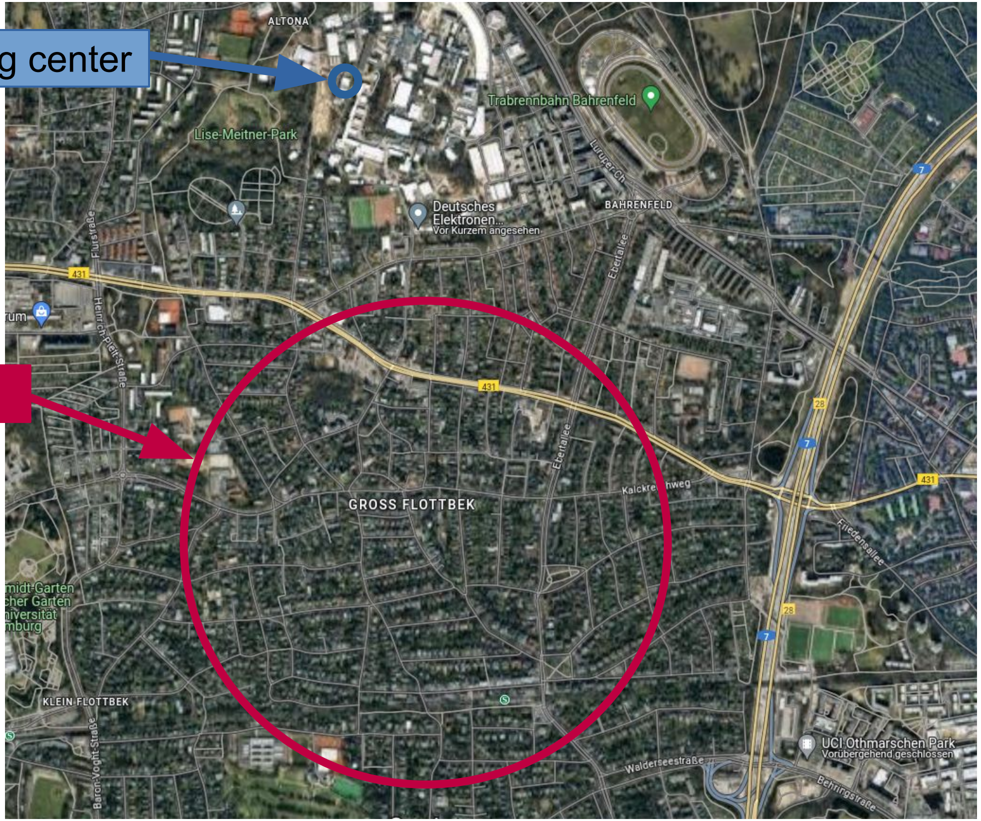
5160 tCO_2

This is:

- 250 000 trees
- 3 700 Single-family homes
- 1 Groß Flottbek

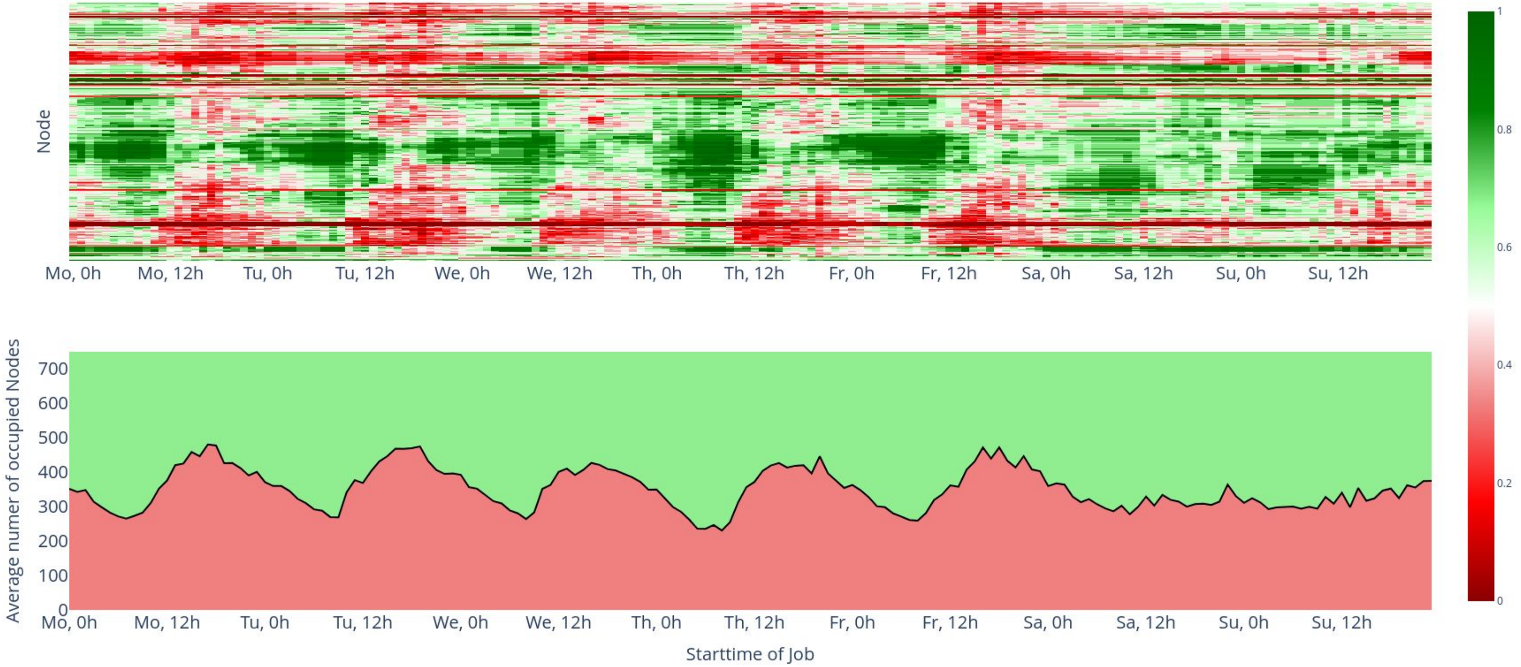
DESY computing center

Groß Flottbek



Source: maps.google.com

How much is actually used?

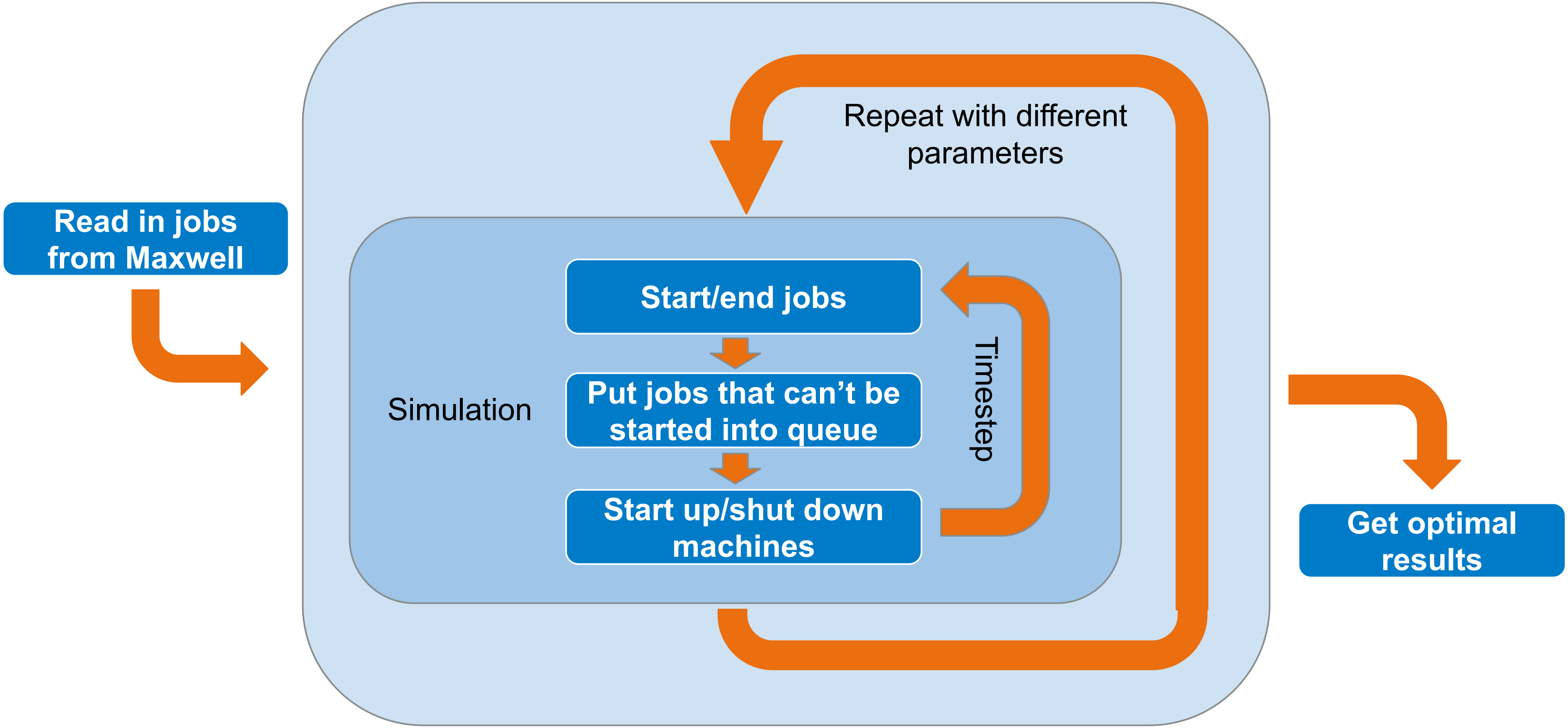


Increasing the energy and CO_2 efficiency of the Maxwell cluster

1. More efficient utilization → Put jobs from other grids on Maxwell
2. Switching off idle nodes → Up to 20% savings without the user noticing
3. Throttling CPUs → Save Electricity at night, when it has a high Carbon intensity

Huge savings potential!
Save ~30% in Carbon emissions

How my simulation worked



Read in jobs from Maxwell

Repeat with different parameters

Simulation

Start/end jobs

Put jobs that can’t be started into queue

Start up/shut down machines

Timestep

Get optimal results

Contact

Deutsches Elektronen-Synchrotron DESY

www.desy.de

João Alvim - joao.alvim@desy.de
Anton Schwarz - anton.schwarz@desy.de
Konrad Kockler - konrad.kockler@desy.de

Scientific Computing