universität freiburg

AUDITOR

Accounting for opportunistic resources

Stefan Kroboth, Michael Boehler, Anton J. Gamel, Benjamin Rottler, Dirk Sammel, Markus Schumacher

26th International Conference on Computing in High Energy & Nuclear Physics

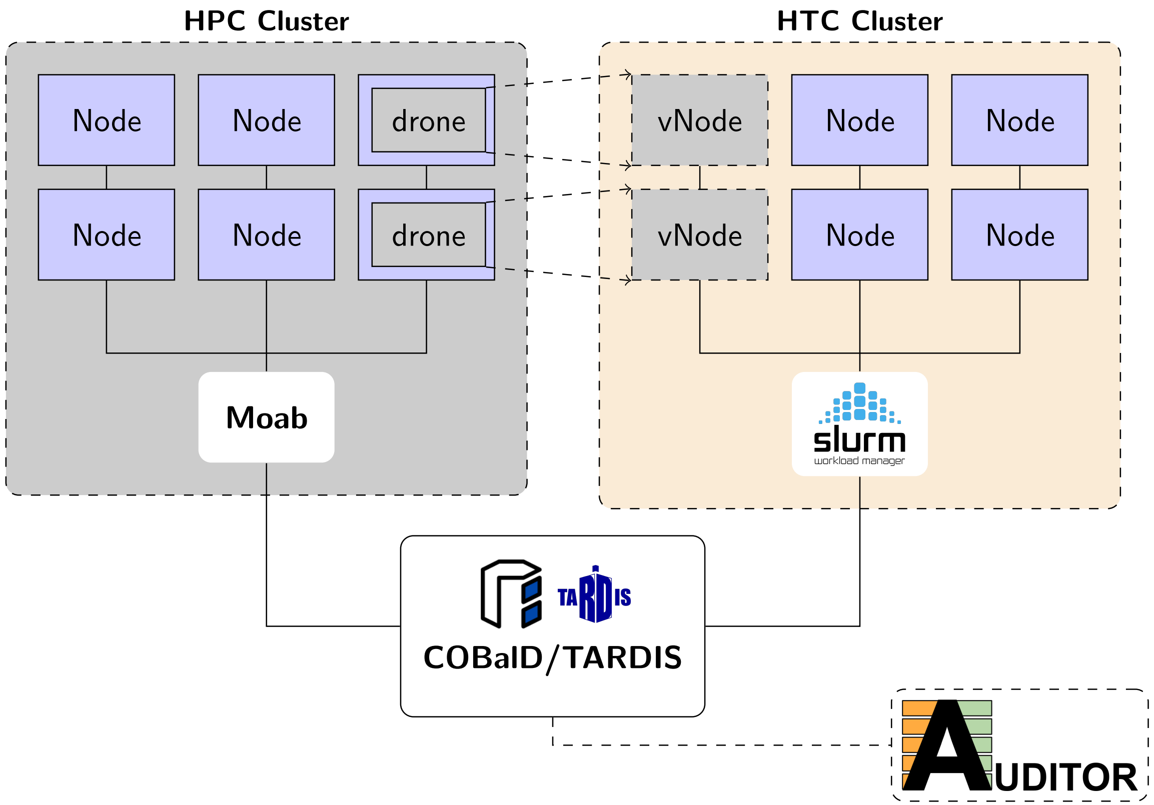
Norfolk, Virginia, USA, May 8-12 2023

Physikalisches Institut
Albert-Ludwigs-Universität Freiburg

IDIUM

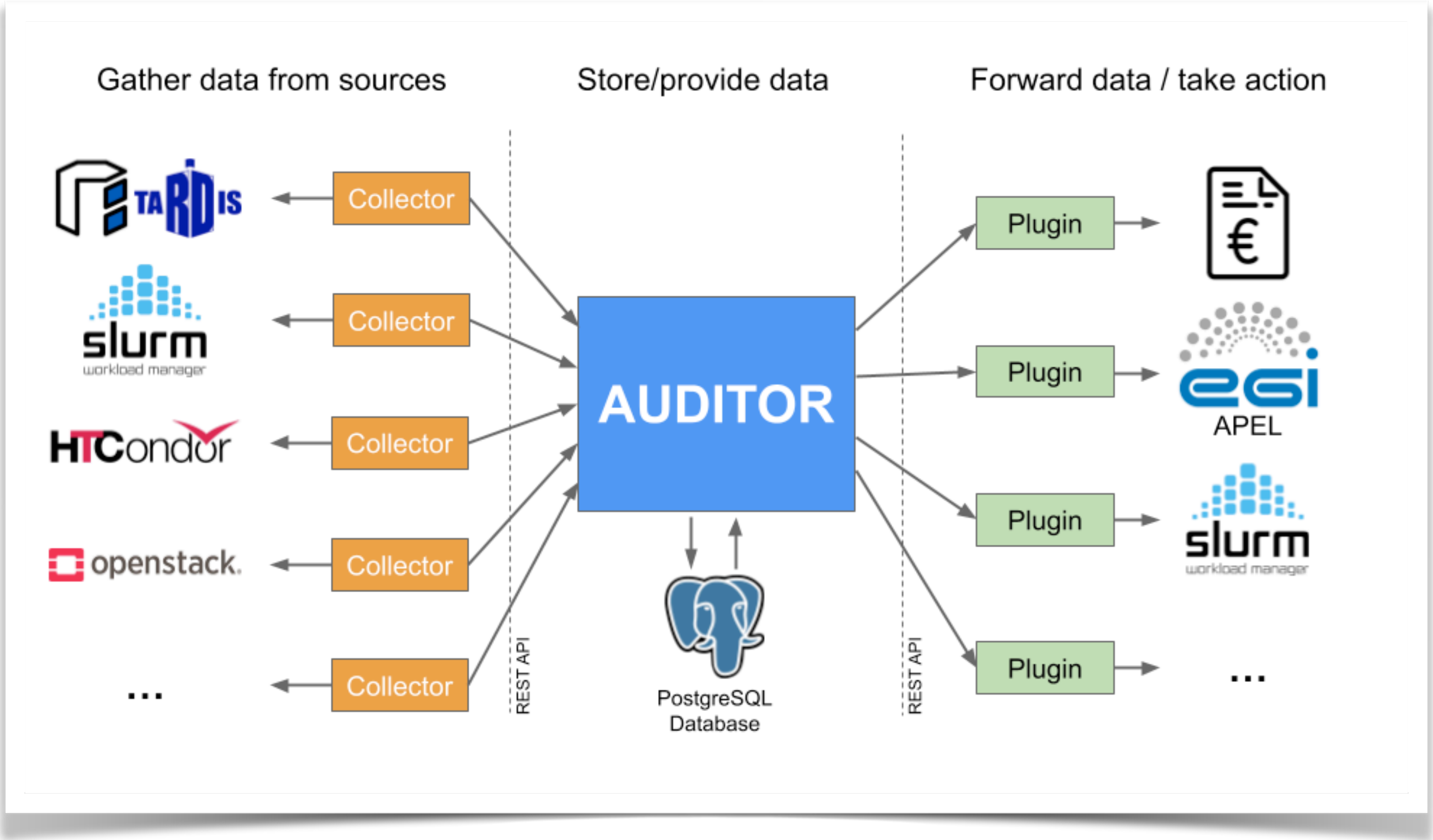
Bundesministerium für Bildung und Forschung

Motivation
Accounting opportunistic resources



- Opportunistic resources can be integrated dynamically and transparently into clusters with COBalD/TARDIS
- Requires a mechanism for accounting
- Challenges
  - Vastly different infrastructures
  - Many potential use cases
- **AUDITOR** provides a multi-purpose accounting ecosystem

AUDITOR
Accounting Ecosystem

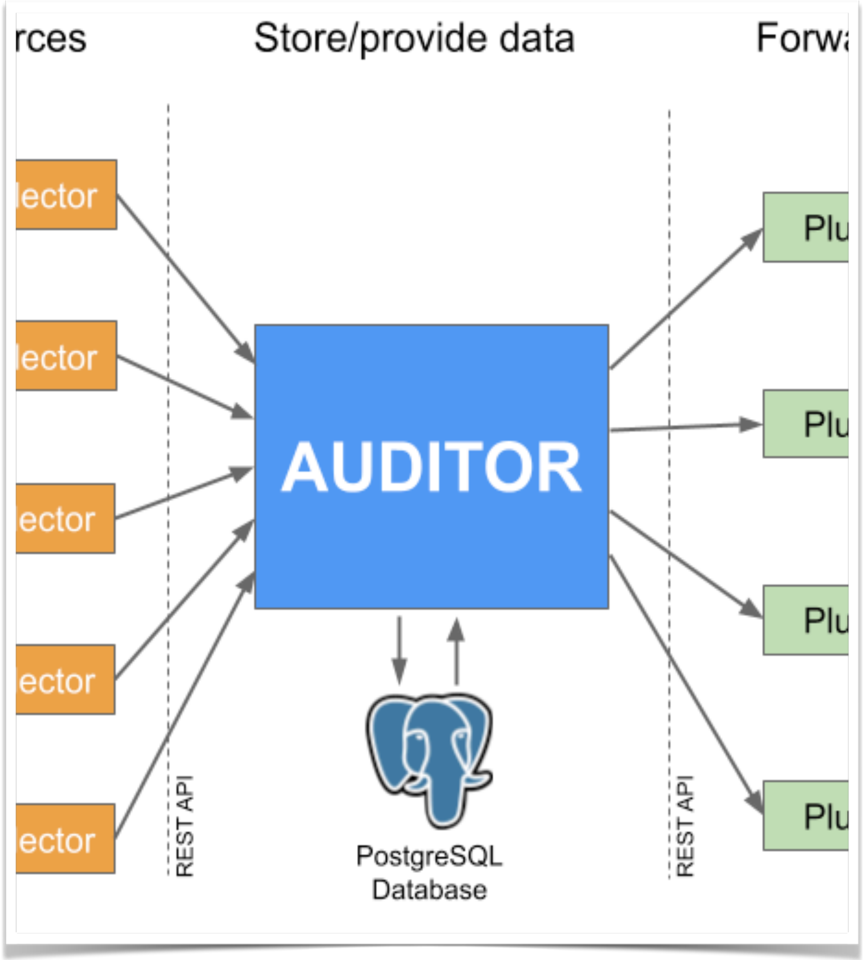


AUDITOR: AccoUnting Data handlIng Toolbox for Opportunistic Resources

Modular accounting ecosystem

- **Collectors:**
  - Accumulate data
- **Core component:**
  - Accepting data
  - Storing data
  - Providing data
- **Plugins:**
  - Take action based on stored data

AUDITOR
Core component



- Implemented in **Rust**
  - Access via REST Interface
- Data stored in PostgreSQL
- Completely stateless
  - No dataloss
  - Suitable for high availability setup
- CI provides **RPMs** and **Docker containers**
- Client libraries in Rust and Python (pyauditor)
- Documentation and code on github:
  - https://github.com/ALU-Schumacher/AUDITOR

Record
Unit of accountable resources

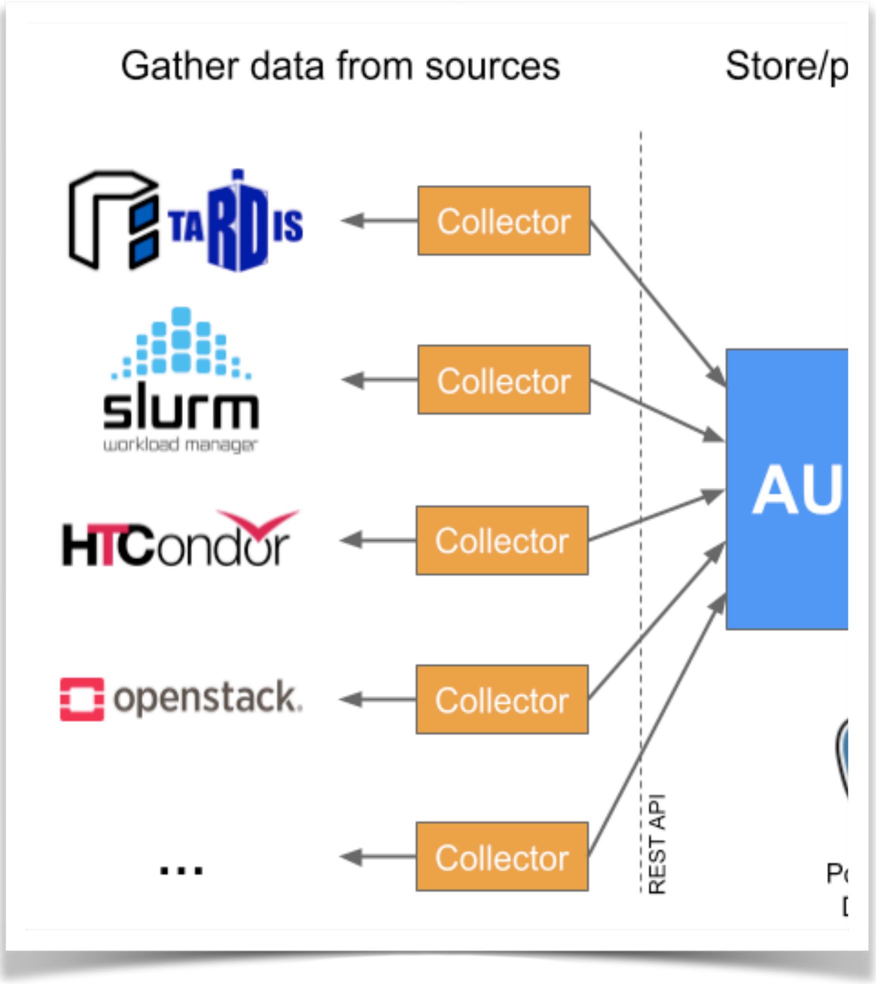
- `record_id`: uniquely identifies the record
- `meta`: multiple key value pairs of the form `String -> [String]`
- `components`: arbitrary number of resources that are to be accounted for (CPU, RAM, Disk, …)
    - `scores`: (multiple) accounting scores supported
- `start_time`, `end_time`: datetime in UTC
- `runtime`: calculated as `end_time - start_time`

- `meta & component` fields allow for maximal flexibility


```json
{
    "record_id": "hpc-4126142",
    "meta": {
      "group_id": [ "atlpr" ],
      "site_id": [ "hpc" ],
      "user_id": [ "atlpr001" ]
    },
    "components": [
      {
        "name": "Cores",
        "amount": 8,
        "scores": [
          {
            "name": "HEPSPEC06",
            "value": 10.0
          },
          {
            "name": "HEPScore23",
            "value": 10.0
          }
        ]
      },
      {
        "name": "Memory",
        "amount": 16000,
        "scores": []
      }
    ],
    "start_time": "2023-02-24T00:27:58Z",
    "stop_time": "2023-02-24T03:41:35Z",
    "runtime": 11617
},
```

Collectors

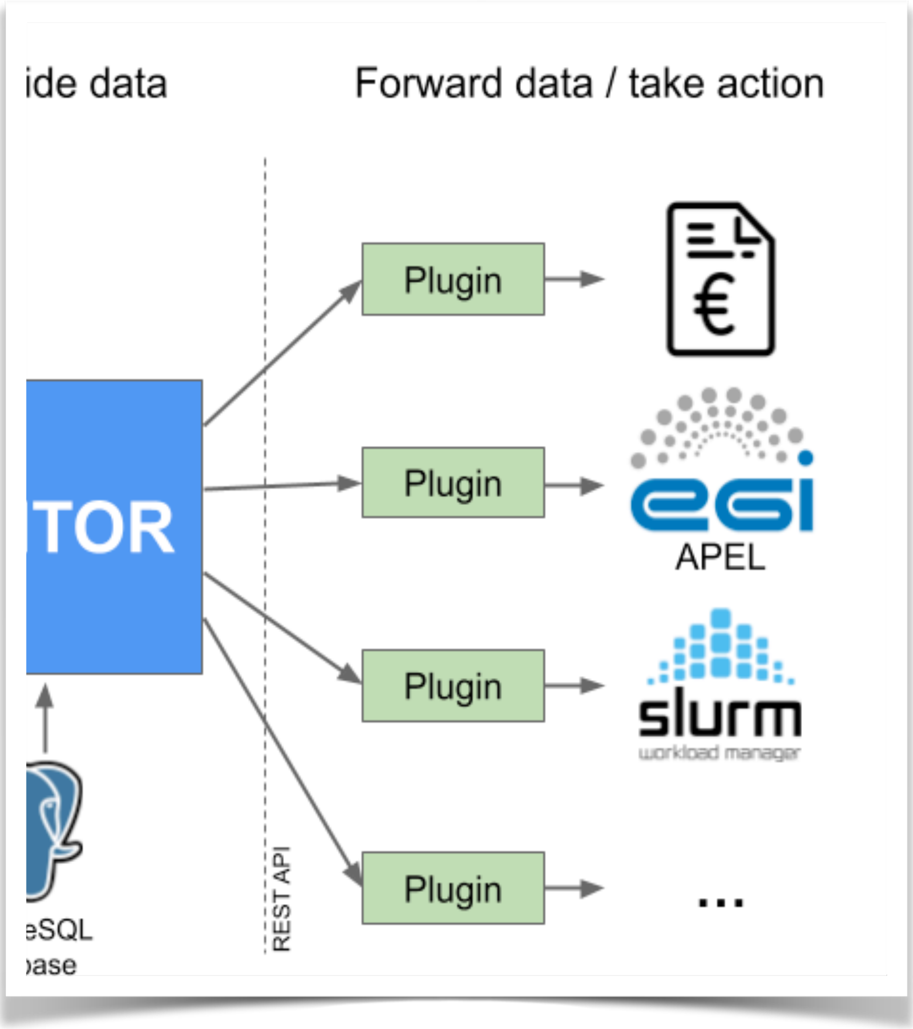
Accumulate data



- **TARDIS Collector**:
  - Collects drone information
- **Slurm Collectors**:
  - Slurm Epilog collector
    - Runs in the epilog of a slurm job
    - Collects data from `scontrol`
  - Slurm Collector
    - Runs as service independent from slurm
    - Collects data from `sacct`
- **HTCondor Collector**: (@ Karlsruher Institute of Technology)
  - Equivalent to Slurm Collector for HTCondor

Plugins

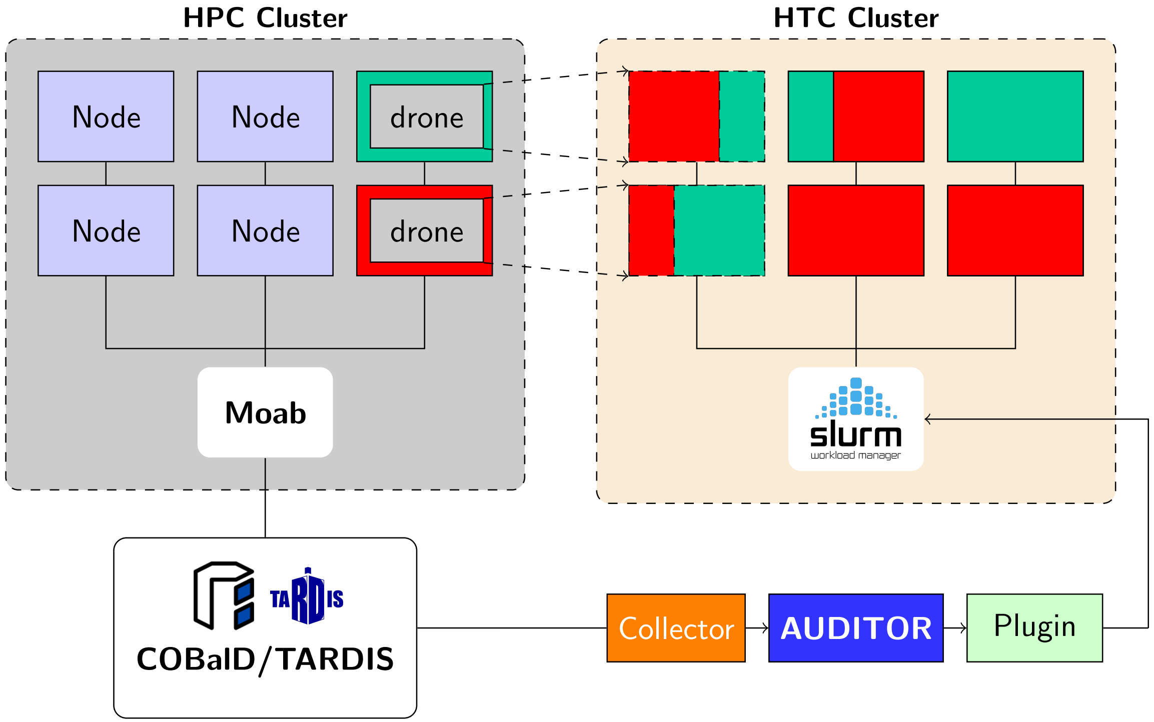
Take action based on stored data



- **Priority plugin**
  - Computes priorities from a list of records
  - Updates priorities on a batch scheduler
- **APEL accounting plugin** (in progress)
  - Properly accounts individual sites behind a COBalD/TARDIS instance
  - Reports accounting data to the APEL accounting Plattform of EGI
- **Utilisation Report** (future project)
  - Analyses requested vs consumed resources of a user
  - Sends a weekly report with possible savings and CO2 footprint

Example Use Case

Adapting the Priority based on provided Resources

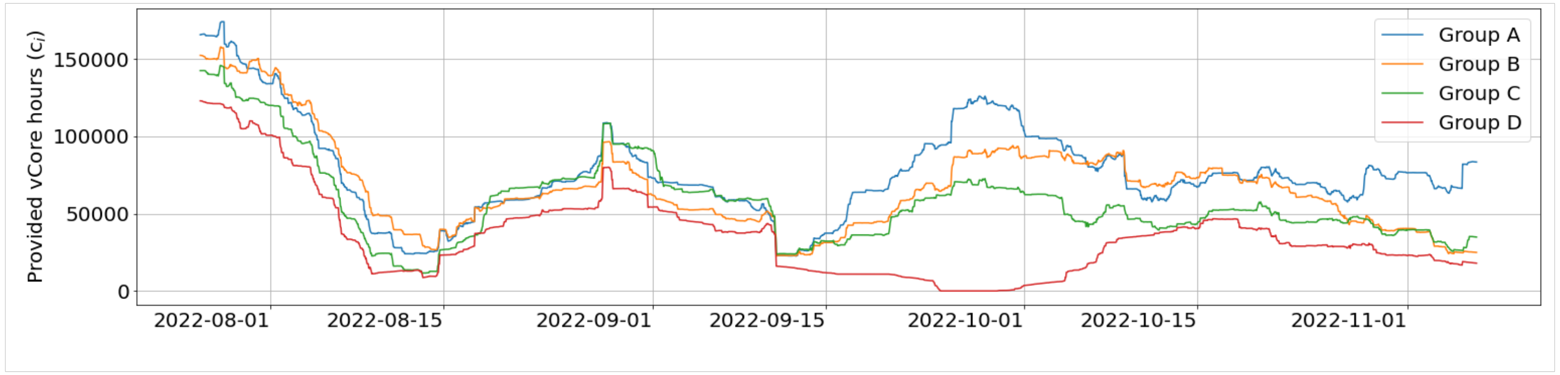


- Opportunistic resources integrated dynamically into HTC cluster with COBalD/TARDIS
  - Several groups request resources on HPC
  - How to guarantee fair share on HTC cluster?

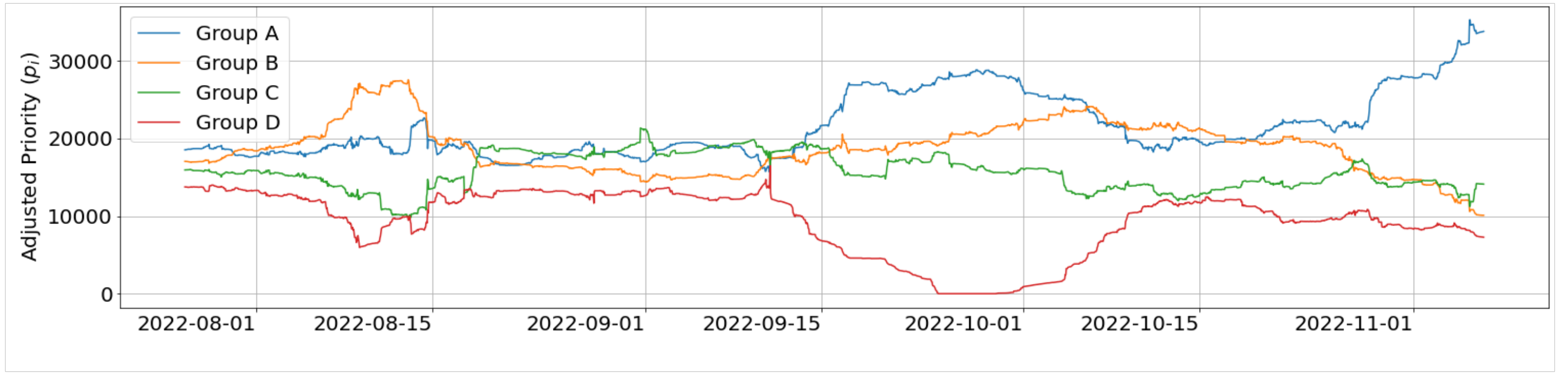
- **TARDIS collector** retrieves info of provided resources on HPC cluster
- **AUDITOR** accounts for provided resources of individual groups [**A** and **B**]
- **Priority plugin** adjusts priorities in HTC cluster

Priority Use Case
Results from HEP Groups @ University of Freiburg

- In total, there are 4 HEP groups [A..D], providing resources



- Priority is adjusted according to the provided resources



Algorithm:

Integral of provided vCores hours over last 14 days for each group:

$$c_i = \int_{t_{now}-14d}^{t_{now}} N_i(t)dt$$

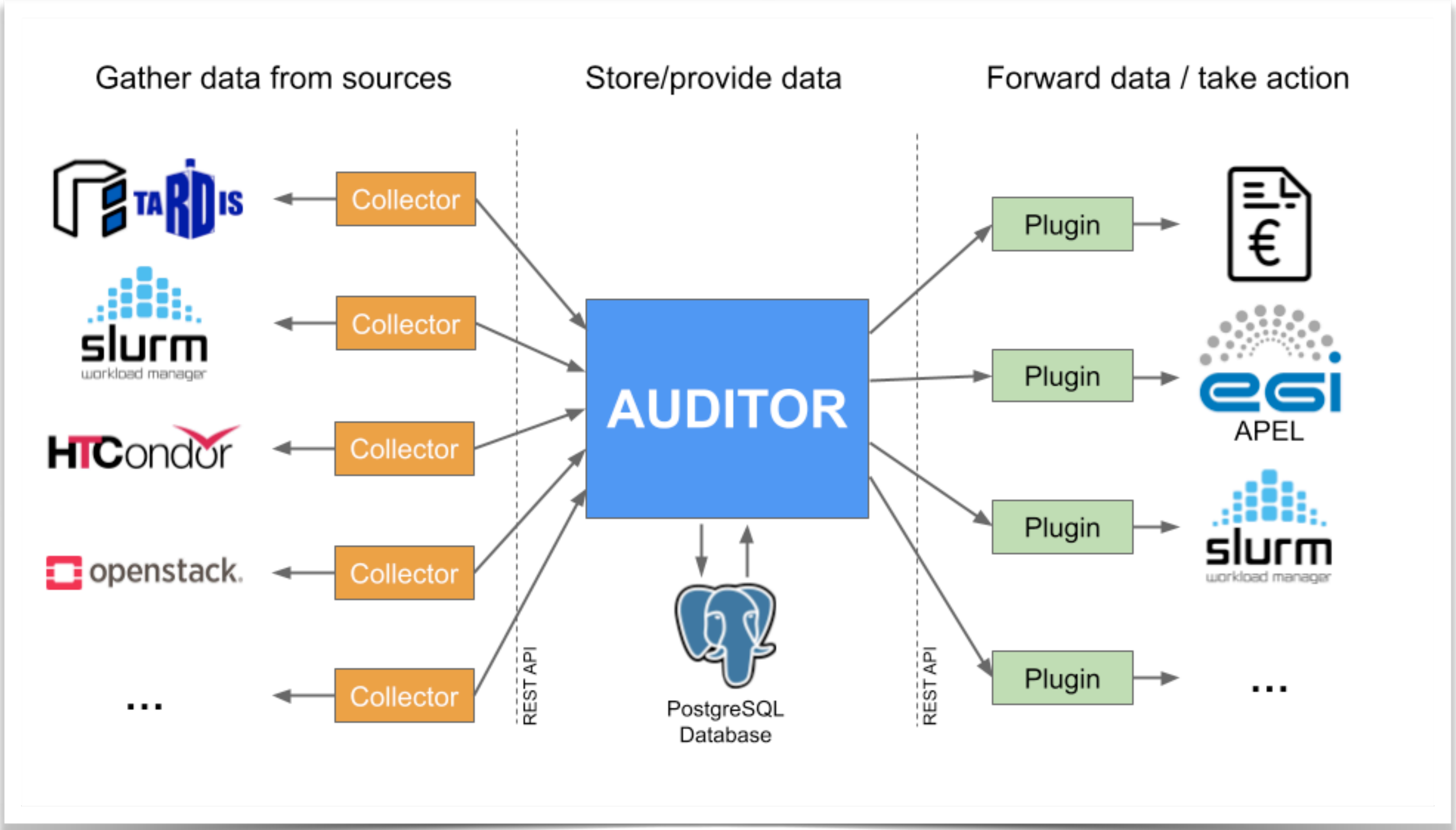
where $i \in \{A, B, C, D\}$

The priority p_i is defines as follows:

$$p_i = \frac{c_i}{\sum_j c_j} \cdot (p_{max} - p_{min}) + p_{min}$$

where $i, j \in \{A, B, C, D\}$

Conclusion



- AUDITOR provides an accounting ecosystem to implement various use cases!

References

Michael Boehler

University of Freiburg

Phone +49 761 203 8409

michael.boehler@physik.uni-freiburg.de

Website: https://alu-schumacher.github.io/AUDITOR/

GitHub: https://github.com/ALU-Schumacher/AUDITOR

FIDIUM: https://fidium.erumdatahub.de

Example pyAuditor

Accessing Data in AUDITOR via a jupyter notebook with pyauditor

- Simple installation via pip:

```
pip install python-auditor
Collecting python-auditor
  Using cached python_auditor-0.1.0-cp38-cp38… (6.0 MB)
Installing collected packages: python-auditor
Successfully installed python-auditor-0.1.0
```

- Make sure that your application can connect to the AUDITOR instance (e.g. setup ssh tunnel)
- Setup the required module & connections

In [1]:
```python
from pyauditor import AuditorClientBuilder
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```python
builder = AuditorClientBuilder()
builder = builder.address("127.0.0.1", 3333)
client = builder.build()
```

In [3]:
```python
list_of_records = await client.get()
```

- transform records into Pandas DataFrame for further data analysis

In [5]:
```python
df = records_to_df(list_of_records)
```

In [6]:
```python
df[['record_id', 'runtime', 'start_time', 'stop_time',
    #'user_id','group_id', 'site_id',
    #'Memory', 'NNodes', 'MaxRSS', 'ReqMem'
    'UserCPU', 'TotalCPU', 'SystemCPU', 'Cores',]].head(3)
```

Out[6]:

	record_id	runtime	start_time	stop_time	UserCPU	TotalCPU	SystemCPU	Cores
0	slurm-4041342	4862	2023-02-20 07:05:44	2023-02-20 08:26:46	3490000	4878000	3547000	2
1	slurm-4041326	5459	2023-02-20 06:56:38	2023-02-20 08:27:37	3490000	4878000	3547000	2
2	slurm-4041340	4963	2023-02-20 07:05:14	2023-02-20 08:27:57	3490000	4878000	3547000	2

In [7]:
```python
_ = df.runtime.plot(kind='hist',bins=np.linspace(0,50000,100),
                    figsize=(6,2))
```