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

**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
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 COBaID/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 defined 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
FIDIMUM: 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

```python
from pyauditor import AuditorClientBuilder
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

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

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

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

• Transform records into Pandas DataFrame for further data analysis

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

```python
In [6]:
1    df[['record_id', 'runtime', 'start_time', 'stop_time',
2        'user_id', 'group_id', 'site_id',
3        'memory', 'nNodes', 'maxRSS', 'ReqMem',
4        'UserCPU', 'TotalCPU', 'SystemCPU', 'Cores']].head(3)
```

```python
Out[6]:
```

<table>
<thead>
<tr>
<th>record_id</th>
<th>runtime</th>
<th>start_time</th>
<th>stop_time</th>
<th>UserCPU</th>
<th>TotalCPU</th>
<th>SystemCPU</th>
<th>Cores</th>
</tr>
</thead>
<tbody>
<tr>
<td>slurm-4043342</td>
<td>4862</td>
<td>2023-02-20 07:00:44</td>
<td>2023-02-20 08:26:46</td>
<td>3490000</td>
<td>4876000</td>
<td>3547000</td>
<td>2</td>
</tr>
<tr>
<td>slurm-4041326</td>
<td>5459</td>
<td>2023-02-20 06:56:38</td>
<td>2023-02-20 08:27:57</td>
<td>3490000</td>
<td>4876000</td>
<td>3547000</td>
<td>2</td>
</tr>
<tr>
<td>slurm-4041340</td>
<td>4963</td>
<td>2023-02-20 07:05:14</td>
<td>2023-02-20 08:27:57</td>
<td>3490000</td>
<td>4876000</td>
<td>3547000</td>
<td>2</td>
</tr>
</tbody>
</table>

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

![Histogram of runtime data](image-url)