universität freiburg

The accounting ecosystem AUDITOR for PUNCH2.0

TA-Meeting

Michael Böhler
July 9th 2025

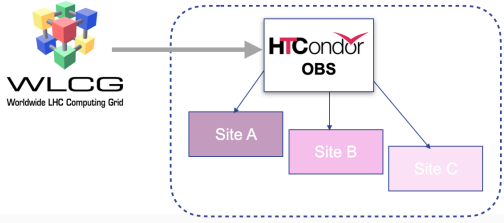
GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

IDIUM

Physikalisches Institut
Albert-Ludwigs-Universität Freiburg

Original Motivation

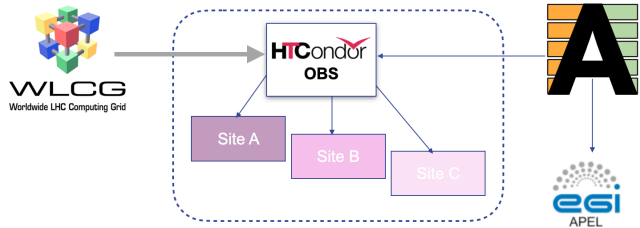
Accounting opportunistic resources



- COBaID/TARDIS allows multiple resources to be clustered in an **O**verlay **B**atch **S**ystem
  - Sub clusters **cannot be accounted individually** with existing tools
  - Requires a dedicated mechanism for accounting
- Challenges
  - Vastly different infrastructures
  - Many potential use cases
- **AUDITOR** provides multi-purpose accounting ecosystem

Original Motivation

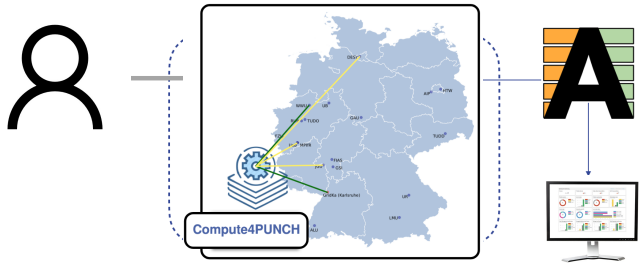
Accounting opportunistic resources



- COBaID/TARDIS allows multiple resources to be clustered in an **O**verlay **B**atch **S**ystem
  - Sub clusters **cannot be accounted individually** with existing tools
  - Requires a dedicated mechanism for accounting
- Challenges
  - Vastly different infrastructures
  - Many potential use cases
- **AUDITOR** provides multi-purpose accounting ecosystem

Original Motivation

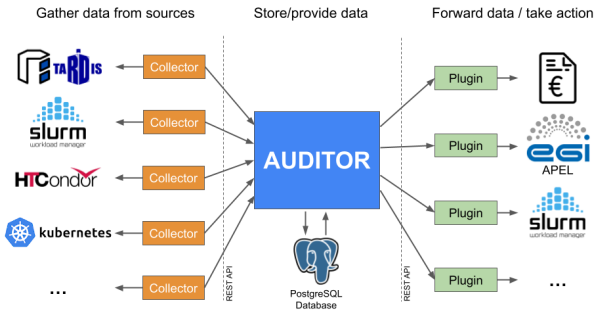
Accounting opportunistic resources



- COBaID/TARDIS allows multiple resources to be clustered in an **O**verlay **B**atch **S**ystem
  - Sub clusters **cannot be accounted individually** with existing tools
  - Requires a dedicated mechanism for accounting
- Challenges
  - Vastly different infrastructures
  - Many potential use cases
- **AUDITOR** provides multi-purpose accounting ecosystem

AUDITOR

Accounting Ecosystem



Modular accounting ecosystem

- **Collectors**
  - Accumulate data
- **Core component**
  - Accept data
  - Store data
  - Provide data
- **Plugins**
  - Take action based on stored data

Documentation and code

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, GPU, . . .)
  - `scores`: (multiple) accounting scores supported
- `start_time`, `end_time`: datetime in UTC
- `runtime`: calculated as `end_time - start_time`

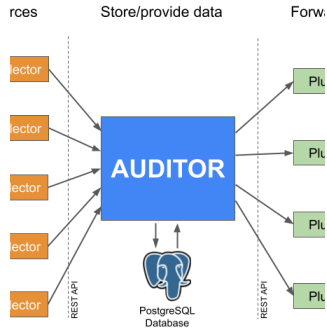
- `meta` & `component` fields allow for maximal flexibility
- supports individual accounting of different CPU types/ corepower values

```
{
    "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
},
```

AUDITOR

Available Collectors and plugins

- **TARDIS Collector**
  - Collect drone information
- **SLURM Collectors**
  - Collect information about SLURM jobs via SLURM CLI commands
- **HTCondor Collector** (dev @ KIT)
  - Equivalent of SLURM collector for HTCondor
- **Kubernetes Collector** (dev @ Wup)
  - accounts resources from kubernetes clusters

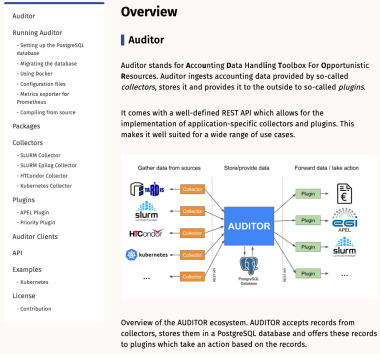


- **Priority plugin**
  - trigger command based on delivered resources (e.g. adjust group priority)
- **APEL accounting plugin**
  - Reports accounting data to the WLCG accounting service (APEL) - also sub clusters

AUDITOR

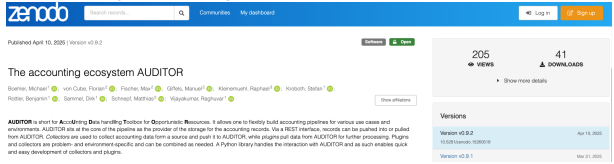
Contributors and Documentation

- Extensive documentation

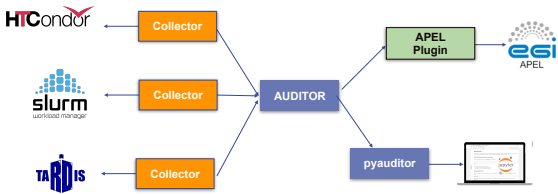


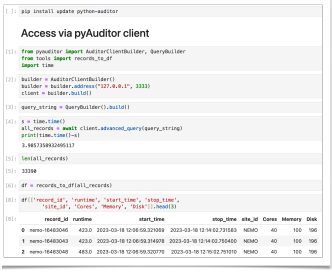
- 12 contributors
- from 3 sites
  - Freiburg (main developement), KIT, Uni Wuppertal
- 23 releases - latest v0.9.3
- Continuous improvements: Commits

- Also on https://doi.org/10.5281/zenodo.12653483



Collecting Accounting Info with AUDITOR



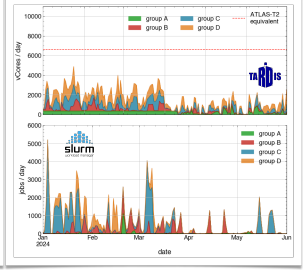


```
[ ]: pip install update python-auditor
```

Access via pyAuditor client

```
[1]: from pyauditor import AuditorClientBuilder, QueryBuilder
     from tools import records_to_df
     import time

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

[3]: query_string = QueryBuilder().build()

[4]: s = time.time()
     all_records = await client.advanced_query(query_string)
     print(time.time()-s)
     3.9457359932465117

[5]: len(all_records)
     33350

[6]: df = records_to_df(all_records)

[8]: df[['record_id', 'runtime', 'start_time', 'stop_time',
         'site_id', 'Cores', 'Memory', 'Disk']].head(3)
```

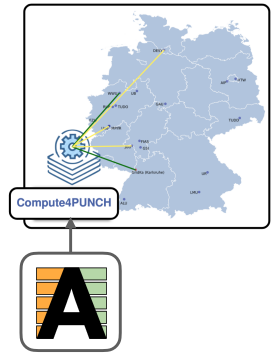
	record_id	runtime	start_time	stop_time	site_id	Cores	Memory	Disk
0	nemo-164830446	423.0	2023-03-18 12:06:59.327089	2023-03-18 12:14:02.731583	NEMO	40	100	196
1	nemo-164830453	423.0	2023-03-18 12:06:59.314078	2023-03-18 12:14:02.758400	NEMO	40	100	196
2	nemo-164830448	483.0	2023-03-18 12:06:59.320770	2023-03-18 12:16:02.751010	NEMO	40	100	196



- Accounting data can be collected in one or more AUDITOR instances from multiple sources
- APEL plugin can report for one or more sites
- **pyauditor** allows to integrate AUDITOR client into python env

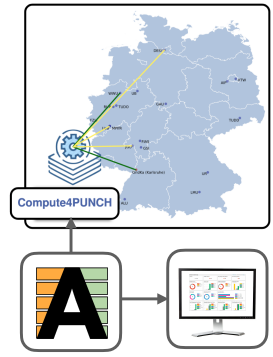
Plans for PUNCH 2.0

1. Introduction of the AUDITOR accounting ecosystem for the compute component in PUNCH 2.0, develop required adjustments
   - Support all availble batch systems
   - Ensure that all required metrics stored in records
   - Support users of participating communities
2. Develop a new plugin which allows to export all data to a graphical web interface - which allows central job monitoring
3. Development of automated end-to-end tests
   - Monitoring the compute component
   - Requires workflows of participating communities



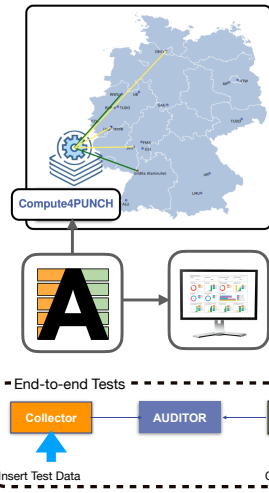
Plans for PUNCH 2.0

1. Introduction of the AUDITOR accounting ecosystem for the compute component in PUNCH 2.0, develop required adjustments
   - Support all availble batch systems
   - Ensure that all required metrics stored in records
   - Support users of participating communities
2. Develop a new plugin which allows to export all data to a graphical web interface - which allows central job monitoring
3. Development of automated end-to-end tests
   - Monitoring the compute component
   - Requires workflows of participating communities



Plans for PUNCH 2.0

1. Introduction of the AUDITOR accounting ecosystem for the compute component in PUNCH 2.0, develop required adjustments
   - Support all availble batch systems
   - Ensure that all required metrics stored in records
   - Support users of participating communities
2. Develop a new plugin which allows to export all data to a graphical web interface - which allows central job monitoring
3. Development of automated end-to-end tests
   - Monitoring the compute component
   - Requires workflows of participating communities

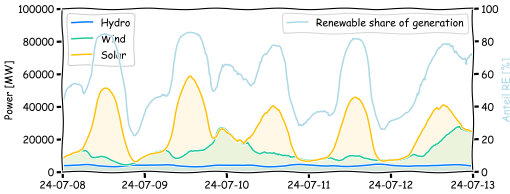


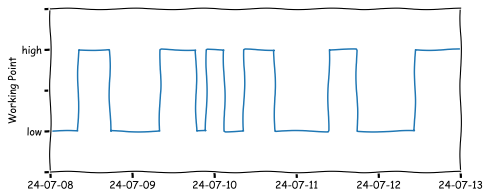
Outlook

What's next?

- Role based access control
- Automatic archiving of AUDITOR db
- Further improve documentation

- Enable the accounting of time-resolved compute performance

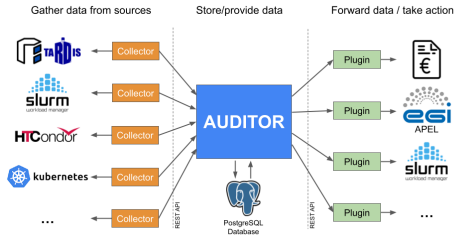




Conclusion

AUDITOR

- Provides accounting ecosystem for various use cases
- Allows to collect accounting data from multiple sources
- Provision via containers or rpms
- Flexible structure of records and ecosystem allows to quickly adapt to future use cases
  - e.g. GPU resources



- Fits very well into Compute4PUNCH, but requires extensions

References

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

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

FIDIUM: https://fidium.erumdatahub.de

Email: auditor@physik.uni-freiburg.de

Michael Boehler

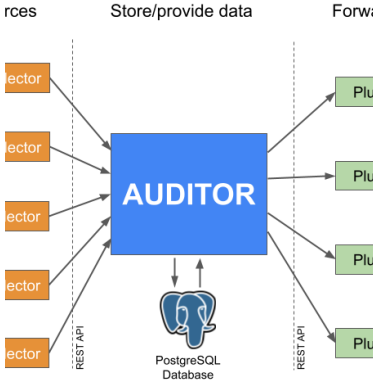
Albert-Ludwigs-Universität Freiburg

michael.boehler@physik.uni-freiburg.de

Back-Up...

Auditor

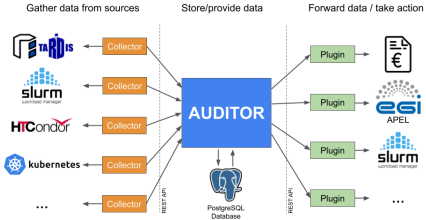
Core component



- Implemented in **Rust**
  - Access via REST interface
- Unit of accountable resources: **Record**
- Data stored in PostgreSQL
- Completely stateless
  - No dataloss
  - Suitable for high availability setups
- Provided as **RPM** or **Docker container**
- Client libraries in Rust and Python (pyauditor)

AUDITOR-Demo

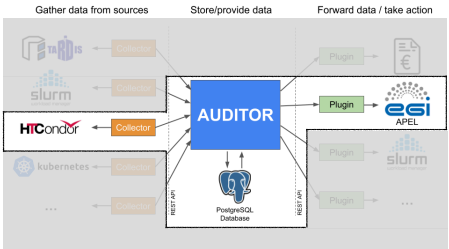
Sandbox to try AUDITOR



- Which components do I need?
- How can I try it out?
- What do I need to configure?

AUDITOR-Demo

Sandbox to try AUDITOR



- Which components do I need?
- How can I try it out?
- What do I need to configure?

README

AUDITOR-demo

This is a mini tutorial on how to install an AUDITOR accounting pipeline from scratch using rpms. The pipeline consists of an HTCondor collector, an AUDITOR instance with a PostgreSQL database and an APEL plugin. All components can be installed together on a small VM. The demo here was set up on a VM with 1 vCore, 2GB RAM and 20 GB disc space on an Alma 9.5 OS.

Prerequisits

https://github.com/ALU-Schumacher/auditor-demo

- Step-by-step instructions
- Install rpms on small VM
- Configure services - with example data
- Let it run!

How to determine the hepscore value?

hep-benchmark-suite

- run HEPScore23-benchmark on **empty** compute node
- benchmark executes 7 HEP workloads
- choose working point (number of cores)
- calculate hepscore/vCore value
  - e.g. 196 $\rightarrow \frac{3610hs23}{196vCores} = 18.4hs23/vCore$