universität freiburg

IDIUM

Bundesministerium für Bildung und Forschung

Physikalisches Institut
Albert-Ludwigs-Universität Freiburg

AUDITOR

Fair sharing of resources between clusters with AUDITOR

Benjamin Rottler, Michael Böhler, Anton J. Gamel, Dirk Sammel, Markus Schumacher

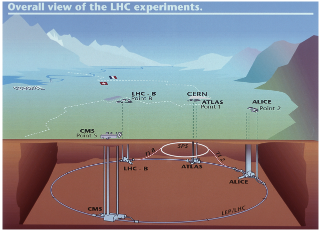
9th bwHPC Symposium
23 October 2023
Mannheim

Introduction

High energy physics (HEP)

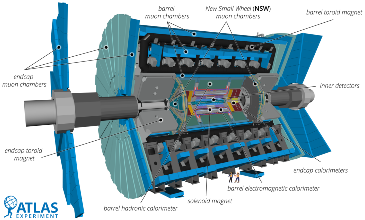
Goal: Find new elementary particles or measure properties of existing ones
→ collide particles, analyze recorded data with the help of simulations

Large Hadron Collider (LHC)



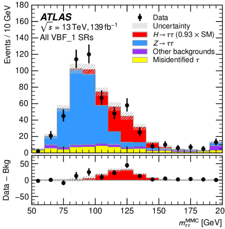
- accelerate protons to very high energies and collide them
- 40 000 000 collisions per second

ATLAS detector



- general-purpose particle detector
- $\sim$ 3000 collisions per second
- $\sim$ 10 PB/year

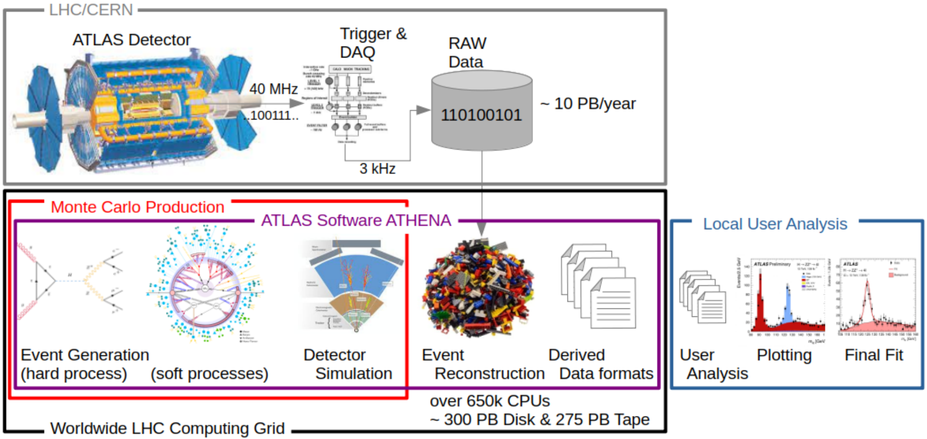
Data analysis



- Measurement of Higgs-boson properties in the $H \rightarrow \tau^+\tau^-$ decay channel

Introduction

ATLAS Data-Analysis Workflow



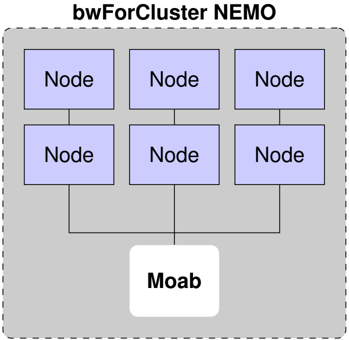
- Simulation (Monte Carlo Production), event reconstruction, and initial post-processing on **Worldwide LHC Computing Grid (WLCG)** → ATLAS production/analysis jobs
- Final steps of analysis on local university cluster

Introduction

Integration of opportunistic resources

bwForCluster NEMO[1]

- ≈ 18000 cores / ≈ 800 TB storage (BeeGFS)
- General software setup
- Scheduler: Moab
- Single user policy
- Jobs by users of local HEP research groups



[1] Neuroscience, Elementary Particle Physics, Microsystems Engineering and Material Science

Introduction

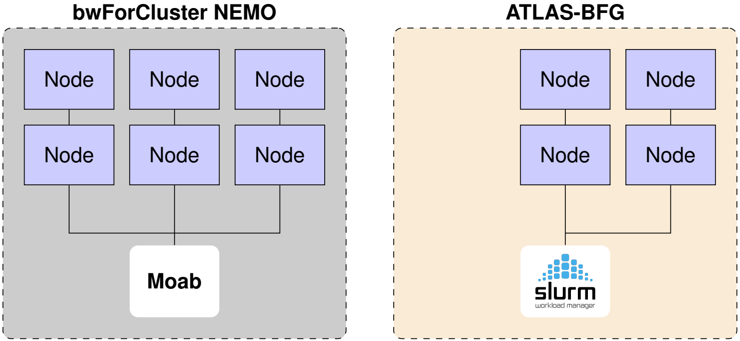
Integration of opportunistic resources

bwForCluster NEMO[1]

- $\approx$ 18000 cores / $\approx$ 800 TB storage (BeeGFS)
- General software setup
- Scheduler: Moab
- Single user policy
- Jobs by users of local HEP research groups

ATLAS-BFG

- $\approx$ 3400 cores / $\approx$ 4 PB storage (dCache)
- Part of WLCG
- Environment specific to ATLAS
- Scheduler: SLURM
- ATLAS production/analysis jobs
- Jobs by users of local HEP research groups



[1] Neuroscience, Elementary Particle Physics, Microsystems Engineering and Material Science

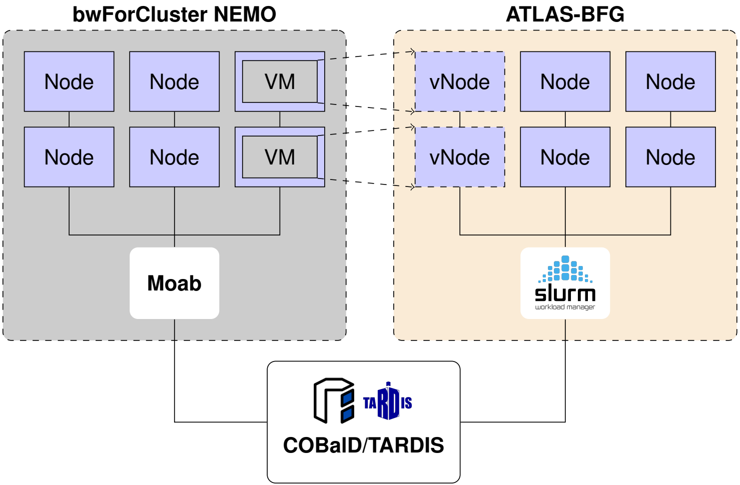
Introduction

Integration of opportunistic resources

COBaID/TARDIS

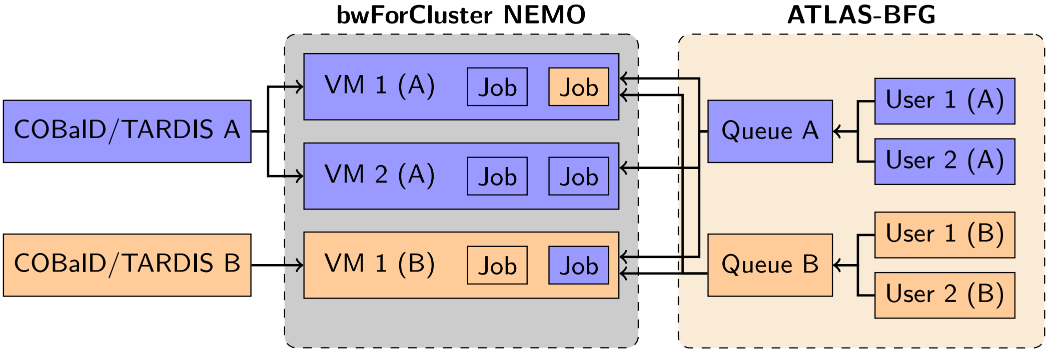
- Opportunistically integrates resources from NEMO into ATLAS-BFG
- Based on demand and availability
- Main developers from group of Prof. Günter Quast (KIT)
  - Also contributions from Freiburg

→ https://github.com/MatterMiners



Motivation

Local setup @ ATLAS-BFG



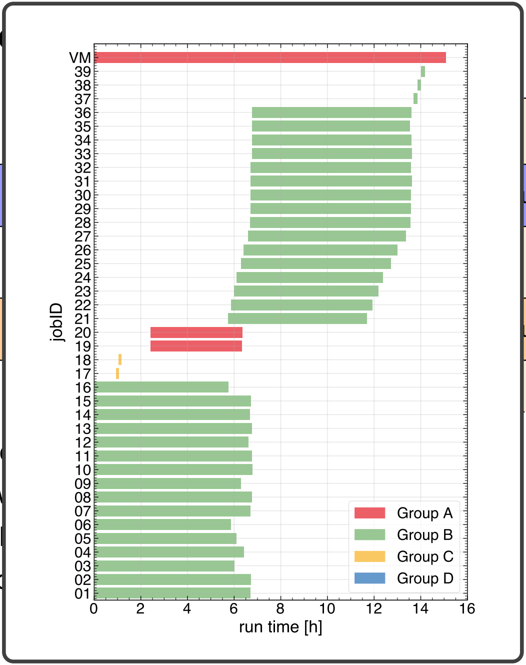
- Four local HEP research groups (A to D) with a share in NEMO
- Each served with its own COBaID/TARDIS instance
- Each has its own SLURM partition (job queue)
- Efficient use of resources due to sharing VMs across HEP groups

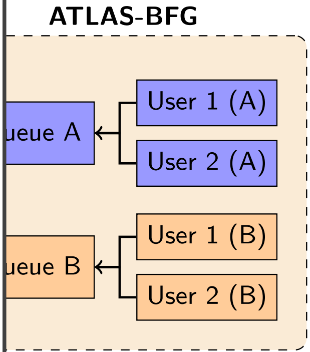
Motivation

Local setup @ ATLAS-BF

COBaID/TARDIS

COBaID/TARDIS

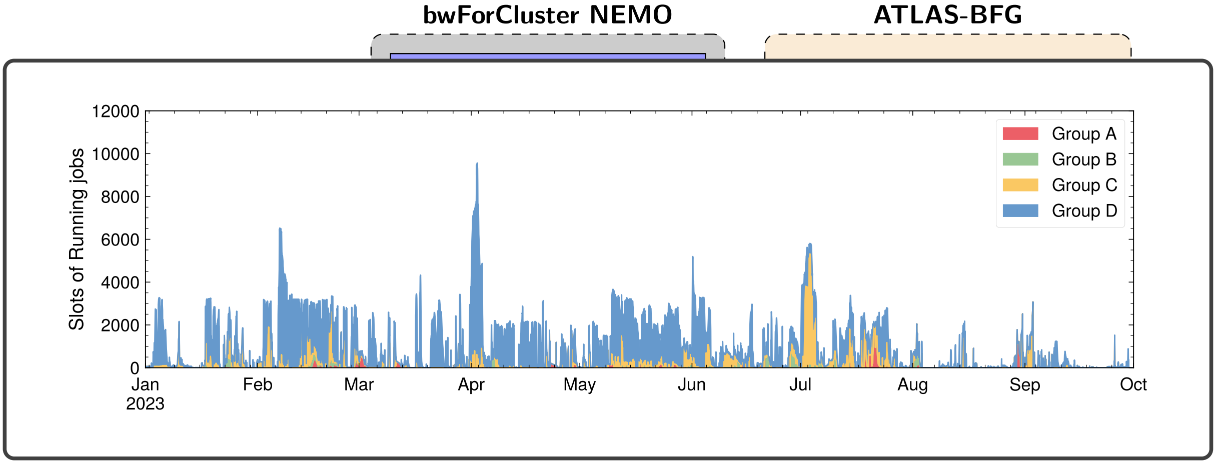




- Four local HEP resear
- Each served with its ov
- Each has its own SLUI
- Efficient use of resourc

Motivation

Local setup @ ATLAS-BFG



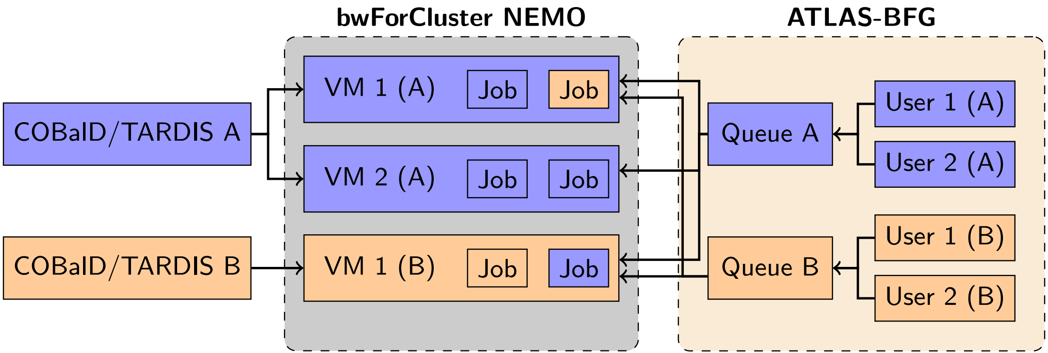
bwForCluster NEMO

ATLAS-BFG

- Each has its own SLURM partition (job queue)
- Efficient use of resources due to sharing VMs across HEP groups

Motivation

Local setup @ ATLAS-BFG

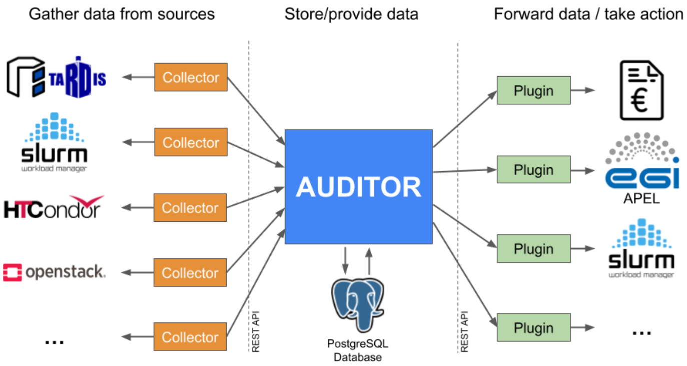


- Four local HEP research groups (A to D) with a share in NEMO
- Each served with its own COBaID/TARDIS instance
- Each has its own SLURM partition (job queue)
- Efficient use of resources due to sharing VMs across HEP groups

How to reflect amount of provided NEMO resources in ATLAS-BFG?

Auditor

Accounting Ecosystem



AUDITOR: Acco**U**nting **D**ata handl**I**ng **T**oolbox for **O**pportunistic **R**esources

Modular accounting ecosystem

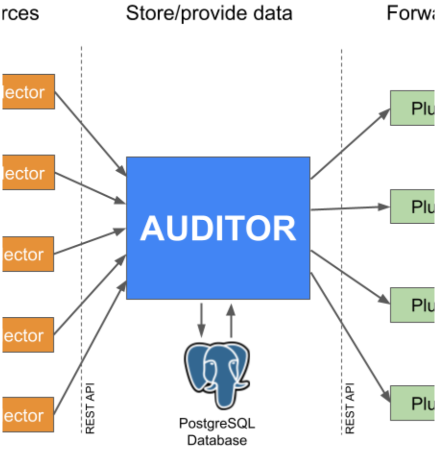
- **Collectors**
  - Accumulate data
- **Core component**
  - Accept data
  - Store data
  - Provide data
- **Plugins**
  - Take action based on stored data

Documentation and code

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

Auditor

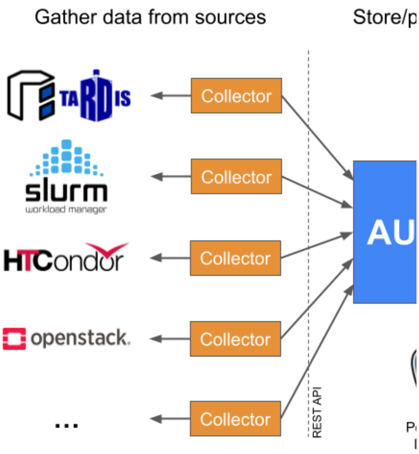
Core component



- 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

Collectors

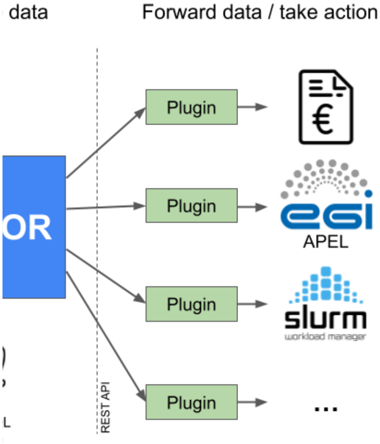
Accumulate data



- **TARDIS Collector**
  - Collect drone information
- **SLURM Collectors** (2 types)
  - Collect information about SLURM jobs via SLURM CLI commands
- **HTCondor Collector** (developed @ KIT)
  - Equivalent of SLURM collector for HTCondor
- Planned collectors
  - OpenStack
  - Kubernetes
  - . . .

Plugins

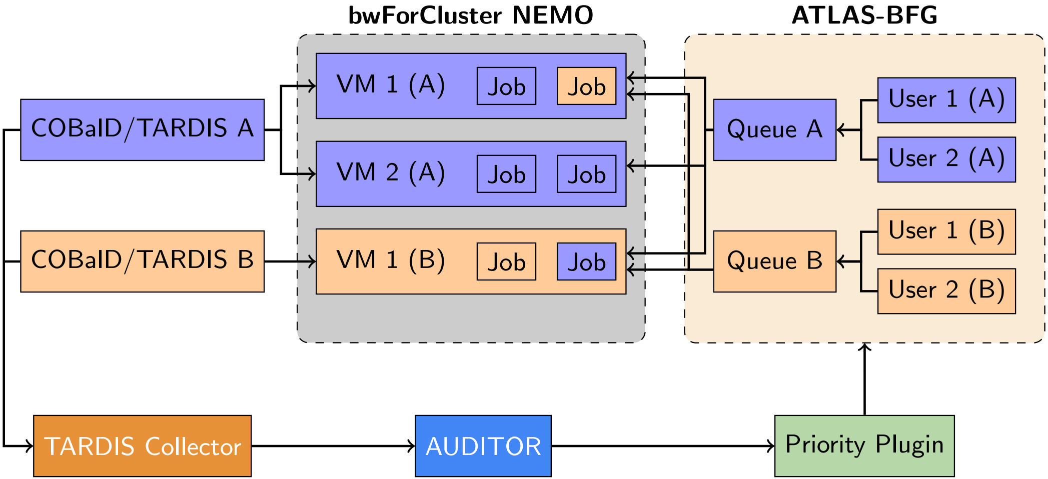
Take action based on stored data



- **Priority plugin**
  - Compute priorities from a list of records
  - Update priorities on a batch cluster
- **APEL accounting plugin** (work in progress)
  - Report accounting data to WLCG accounting service (APEL)
- **Utilization report** (future project)
  - Analyse requested vs. consumed resources of a user
  - Send a weekly report with possible savings and CO_2 footprint

AUDITOR Priority plugin

Adapting the ATLAS-BFG priority based on provided NEMO resources

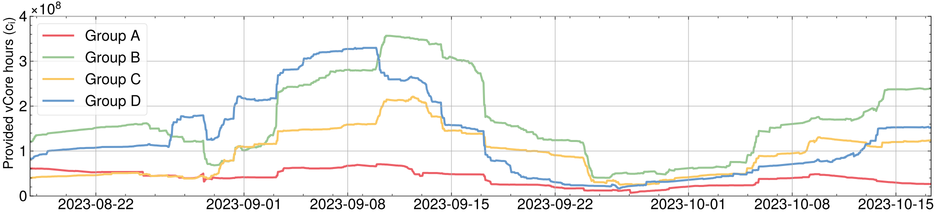


- use TARDIS Collector/AUDITOR/Priority Plugin pipeline to adjust group priorities on ATLAS-BFG

AUDITOR Priority plugin

Results

- Provided resources of the four local HEP groups

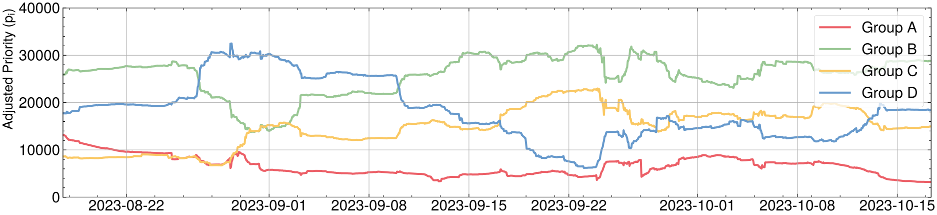


Integral of provided vCore hours over last 14 days for each group:

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

with $i \in A, B, C, D$

- Priority is adjusted according to the provided resources



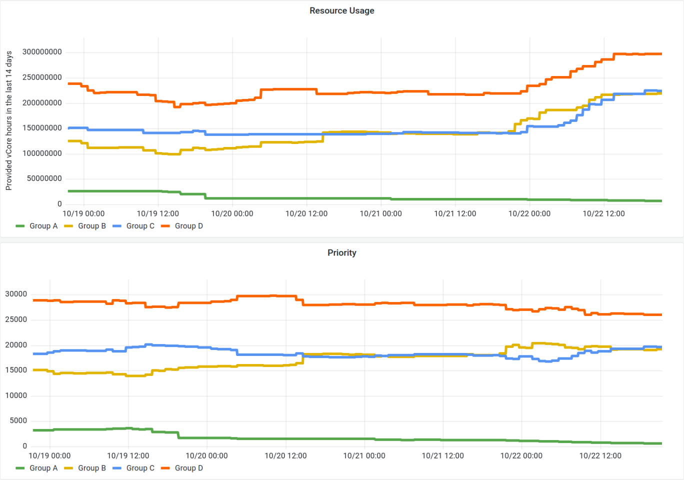
Priority p_i is defined as:

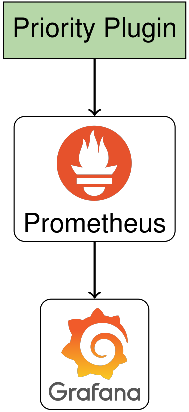
$$p_i = \frac{c_i}{\sum_j c_j}(p_\mathrm{max} - p_\mathrm{min}) + p_\mathrm{min}$$

with $i, j \in A, B, C, D$;
$p_\mathrm{min} = 1$; $p_\mathrm{max} = 65335$

AUDITOR Priority plugin

Real-time monitoring





Priority Plugin

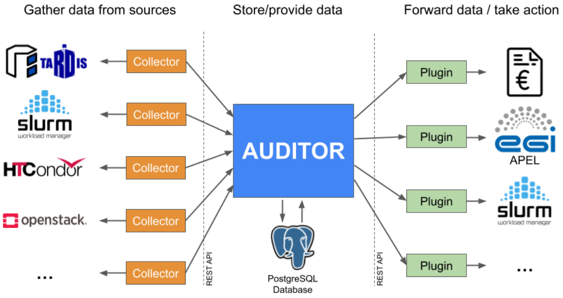
Prometheus

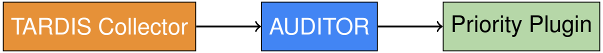
Grafana

- Recent development: Prometheus exporter for priority plugin
- Real-time monitoring of priority adjustments (with e.g. Grafana)

Conclusion

- AUDITOR provides an accounting ecosystem for various use cases
- NEMO resources are opportunistically integrated into ATLAS-BFG cluster
  - known environment for local HEP users
  - sharing resources across HEP groups allows for efficient use
- Priority plugin guarantees fair distribution of integrated resources between HEP groups





Future challenge
Huge growth in dataset size and raising demand for sustainable compute resources increases need for efficient resource utilization

References

AUDITOR

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

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

FIDIUM: https://fidium.erumdatahub.de

Benjamin Rottler

Albert-Ludwigs-Universität Freiburg

benjamin.rottler@physik.uni-freiburg.de

Backup

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

```
{
    "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
},
```