Job-specific performance monitoring on HPC clusters: Challenges and Solutions

presented by
Christian Iwainsky

TECHNISCHE UNIVERSITÄT DARMSTADT

Friedrich-Alexander-Universität Erlangen-Nürnberg

NHR4CES NHR for Computational Engineering Science

NHR FAU

Hessisches Kompetenzzentrum für Hochleistungsrechnen

brain ware for science

HESSEN Hessisches Ministerium für Wissenschaft und Kunst

Todays Topics

1) Background: Who, What and Why?
2) Introduction Cluster Cockpit
3) Introduction PathoJobs
4) Usage experience / What does it take to use it?

Background

- HPC @ TU Darmstadt
  - One of 9 NHR centers in Germany
  - > 1000 compute nodes
  - > 10000 total cores @ > 4GB RAM/ core
  - > 500 active users (students, phd students, post-doc, researchers and profs.)

- Contributors & Co-Developers:

  » Jan Eitzinger, Thomas Zeiser, Alex Wiens, Thobias Watermann, Christian Iwainsky

NHR4CES NHR for Computational Engineering Science

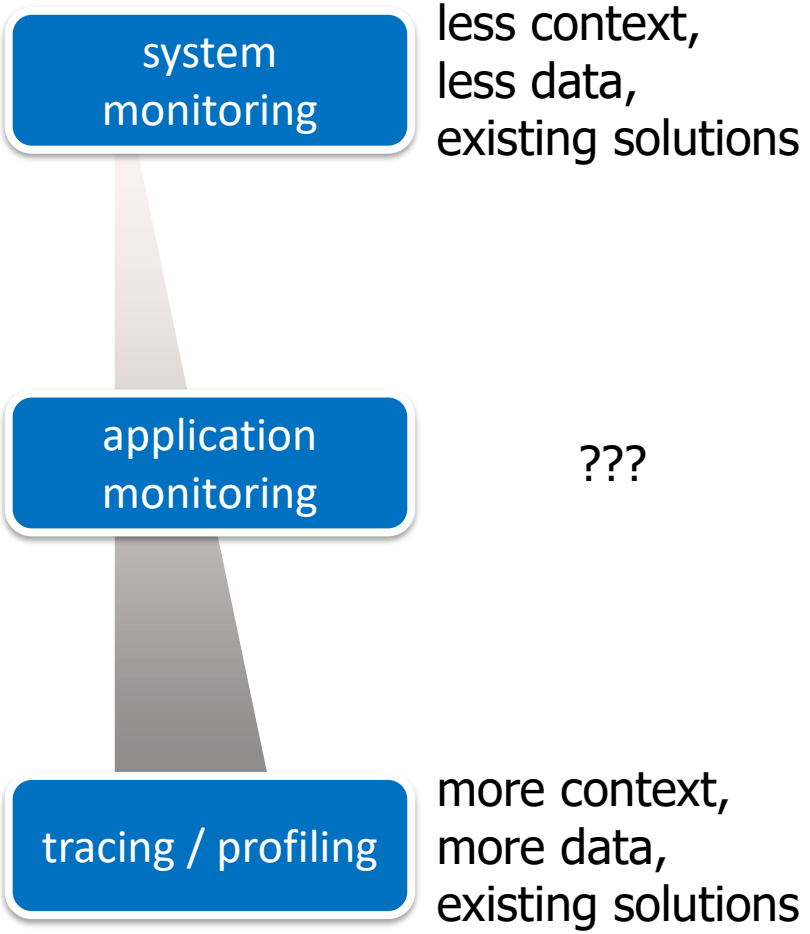
Paderborn Center for Parallel Computing

NHR FAU

ZIB ZUSE INSTITUTE BERLIN

Challenges

- System operation:
  - system healthy?
  - all hardware operational?
  - providing peak performance?
- Usage monitoring:
  - what ressources are used ?
  - which ressources are the limiting factor?
    (compute units, memory bandwidth, memory capacity ...)?
  - are ressouces utilized, or just occupied?
- Performance monitoring / analysis:
  - what is the software doing?
  - why is the software behaving like that?



CusterCockpit: Concept and Purposes

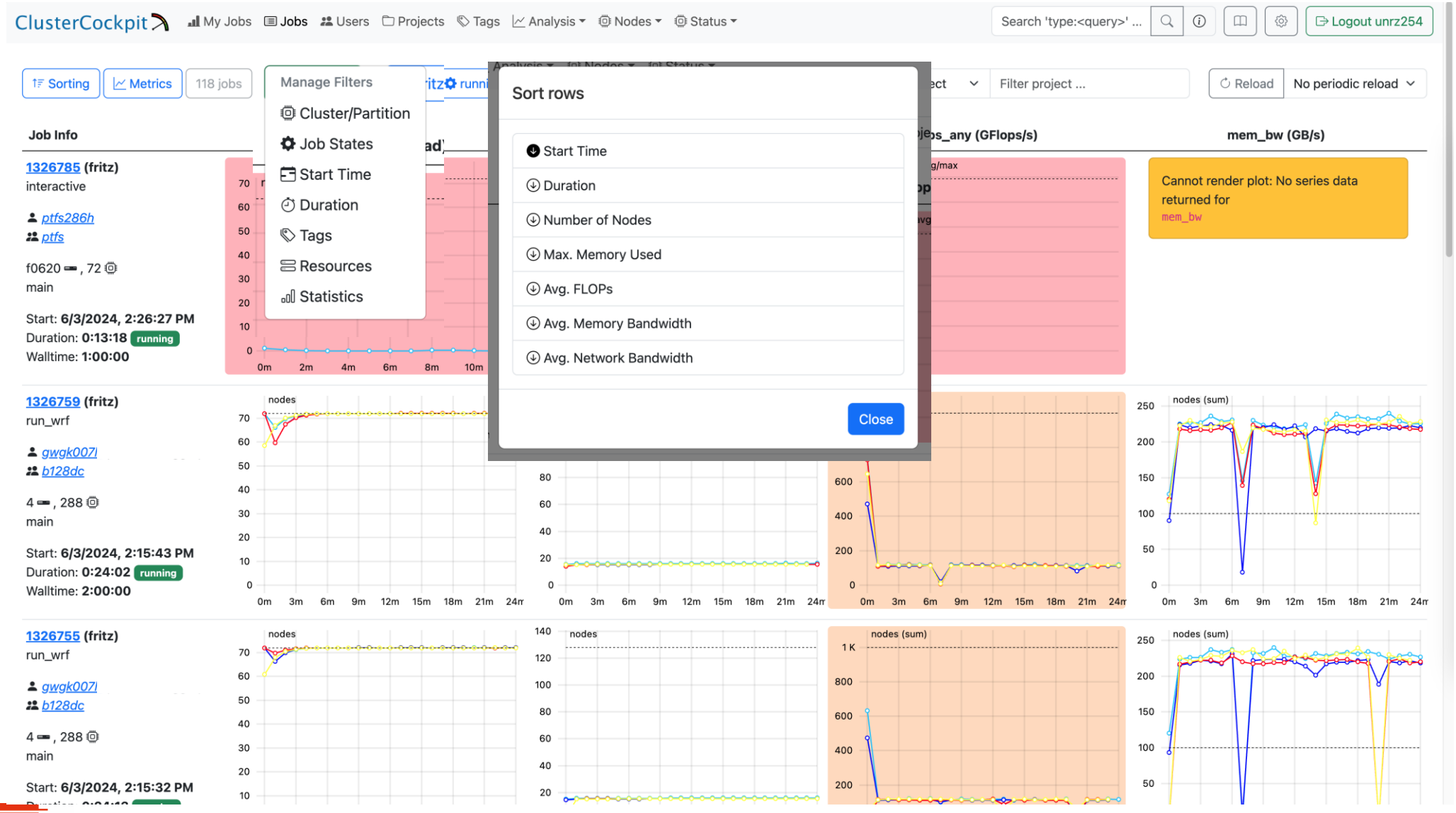
- Cluster-wide continuous monitoring
  - Hardware Performance Monitoring (HPM) metrics: CPI, GFlops/s, Memory bandwidth; load, memory usage, File IO, Network utilization, RAPL energy metrics, GPUs …
  - Fixed frequency measurements for native granularities (core, memory domain, socket, node)
- UI for
  - job monitoring,
  - early detection of performance or runtime issues
  - access to aggregate statistics for users and jobs
  - Web-interface (different views for roles: admin, support, manager, user)
    - Search, filter, tag, and sort jobs, users, and projects
    - Job-specific views with metric plots, job statistics, job meta information
    - Cluster-specific views with overall utilization, job and user statistics

ClusterCockpit: Overview

- Job-specific Performance and Energy Monitoring Framework in HPC computing centers

https://github.com/ClusterCockpit/

CusterCockpit: Job list



CusterCockpit: Job view

4223878 (woody)
MCMC_Ch6

iwtt015h
iwtt

w2202 (shared) , 12
icelake

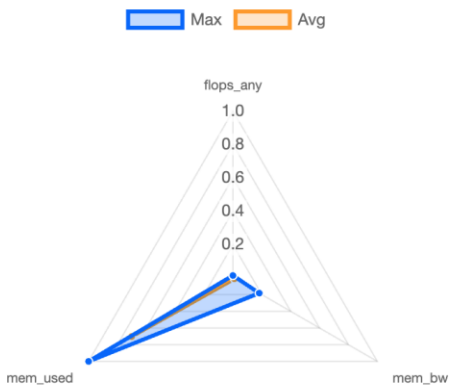
Start: **6/3/2024, 8:25:27 AM**
Duration: **6:31:58** running
Walltime: **23:59:00**

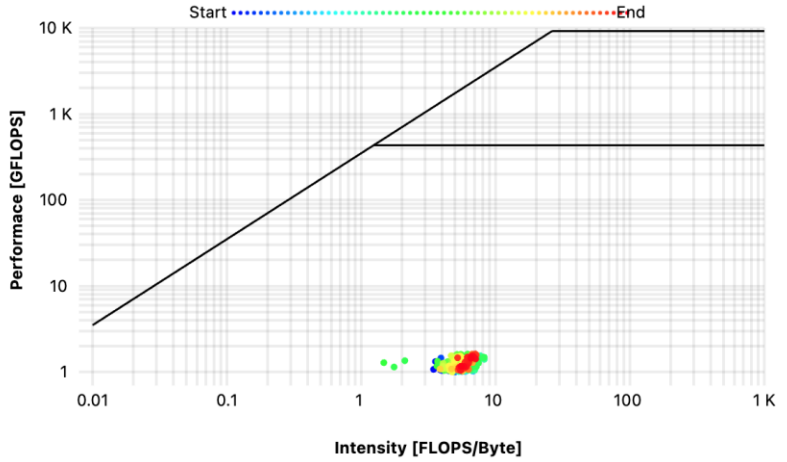
Core Metrics Footprint

flops_any	1.22 / 421	GFlops/s
mem_used	22.48 / 43	GB

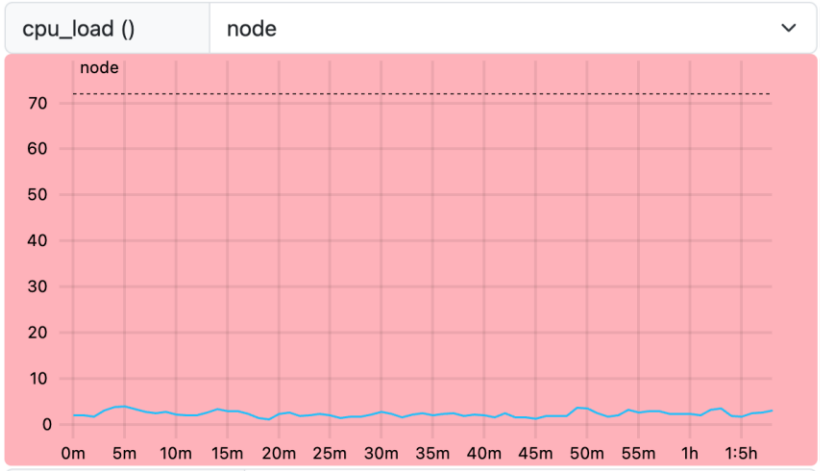
Concurrent Jobs

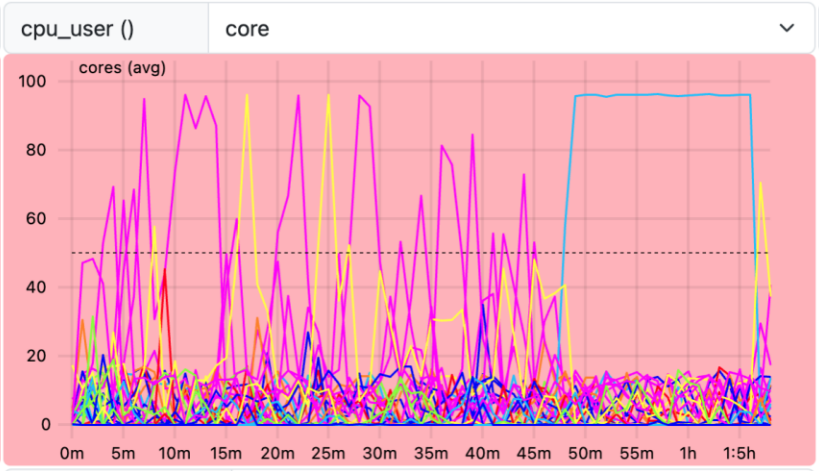
- See All
- 4224735
- 4224740
- 4225073
- 4225074
- 4225095
- 4225096
- 4225147
- 4225148
- 4224823
- 4225174
- 4225175
- 4225030
- 4225032
- 4225039
- 4225040
- 4225046
- 4225061
- 4225332
- 4225146
- 4225196
- 4225253
- 4224749
- 4225298

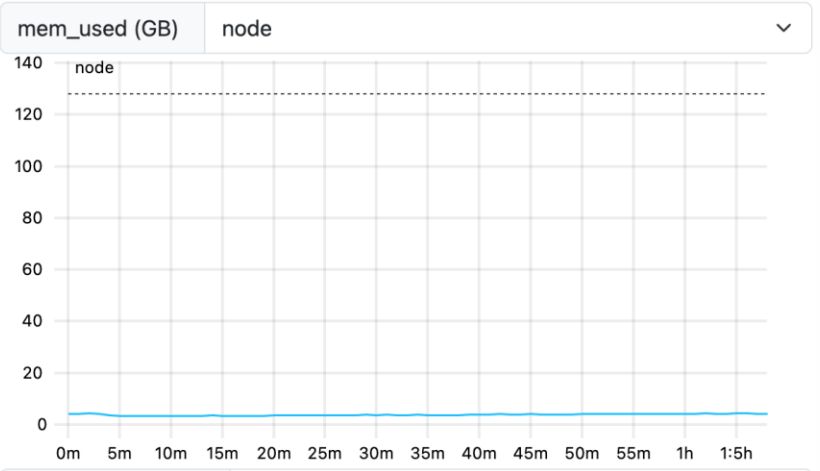




Manage Tags | Metrics





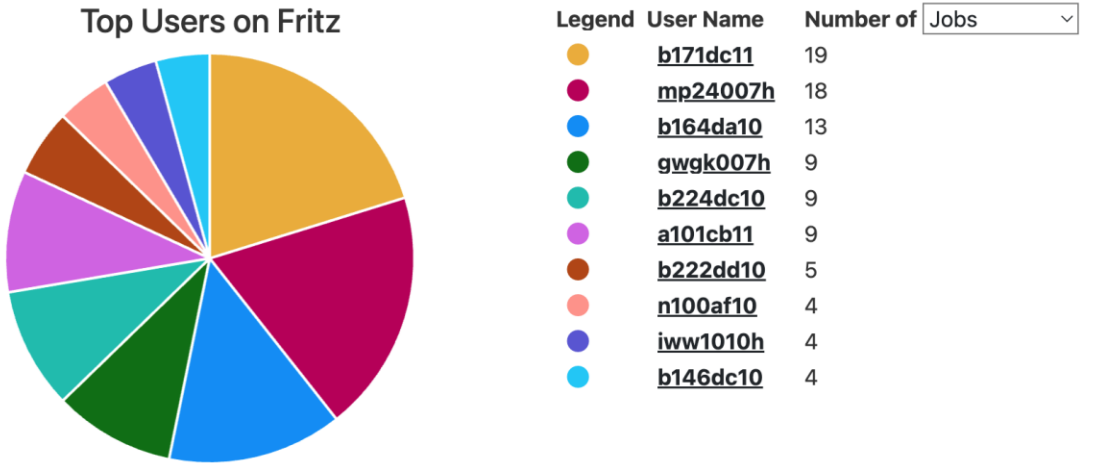


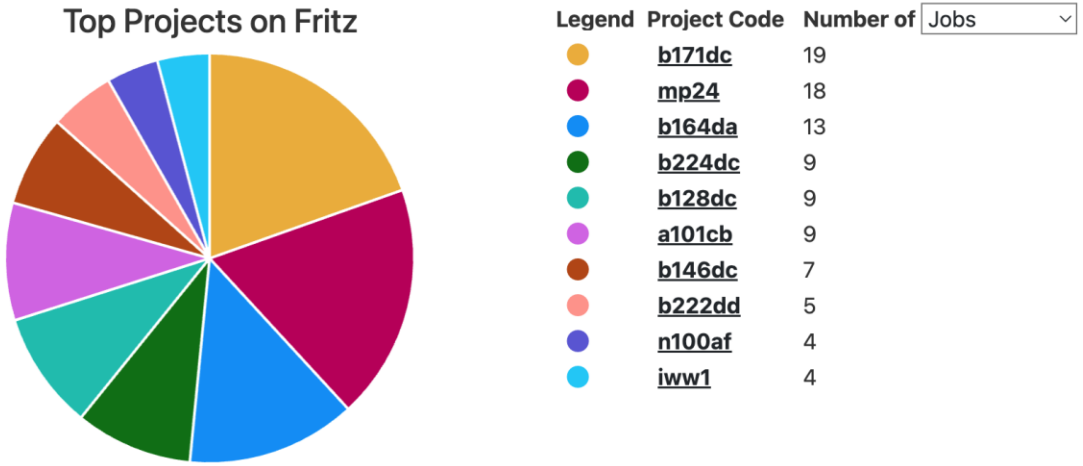
CusterCockpit: User list (a project list is also available)

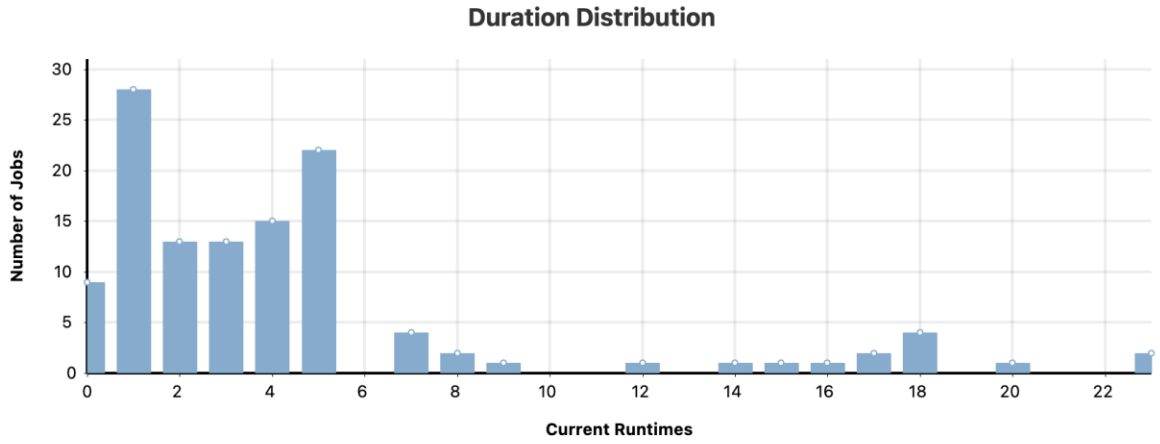
Search users | Filter by username | Filters | Last 30 Days

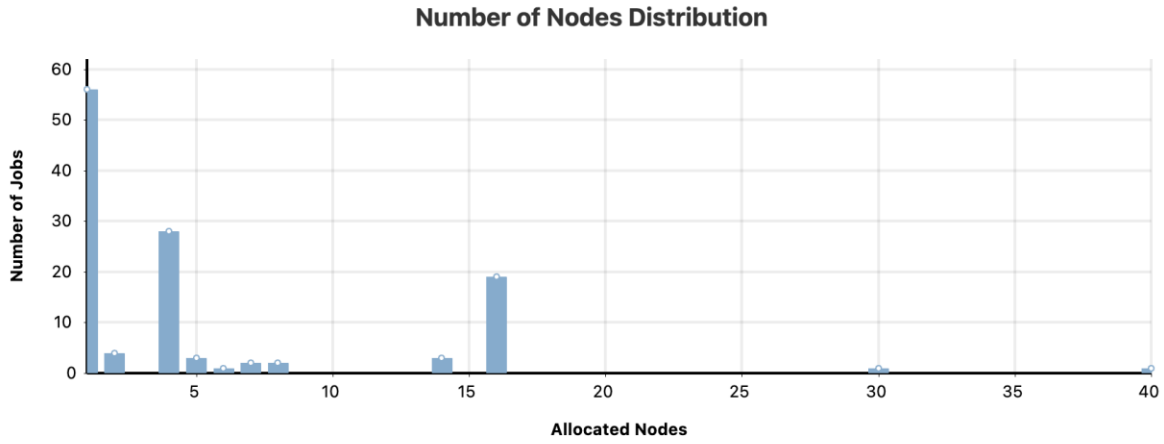
Username	Total Jobs	Total Walltime	Total Core Hours	Total Accelerator Hours
mppi102h	16664	28376	28376	0
mpwm018h	4159	25052	25052	0
bcpc001h	1838	23682	378909	23681
mppi044h	119916	19073	32914	0
b178bb11	11302	18481	295709	18481
b133ae20	716	13705	986793	0
k102de15	4234	13432	967142	0
b187cb13	1446	12146	194313	12146
bcml005h	3562	10032	10032	0
mfip100h	2826	9880	59281	0
mpet007h	533	9864	631300	0
bccc115h	763	9373	299900	0
b189da10	3573	8197	3541186	0
b179dc10	1023	8186	130986	8186
iwwm101h	3479	7811	141933	8870
b186dc14	5967	7431	118900	7431
mppi045h	761	7318	29274	0
b174dc13	314	6996	223874	13992

CusterCockpit: Cluster status view (aka management view)





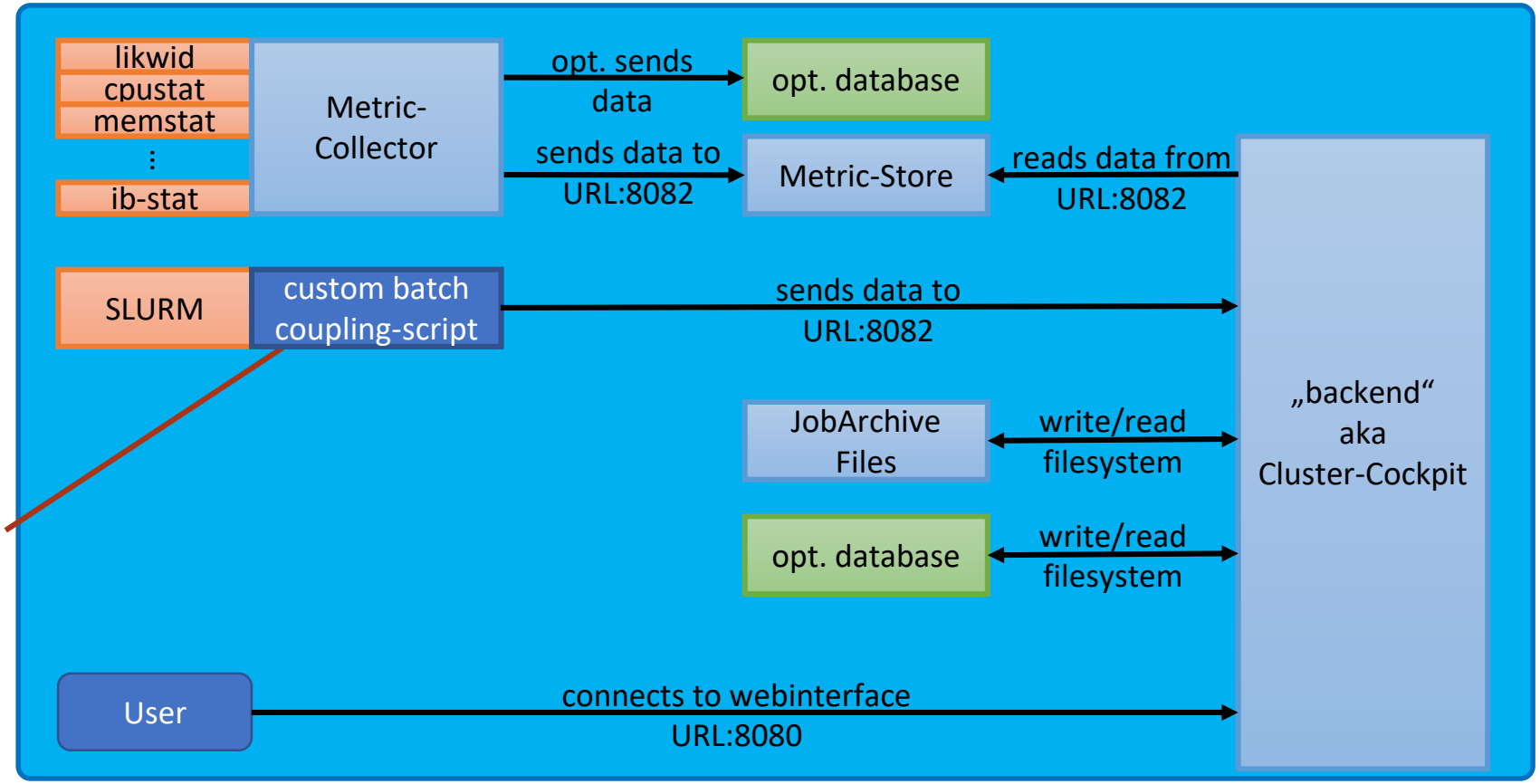




ClusterCockpit: Overview

- Job-specific Performance and Energy Monitoring Framework in HPC computing centers
- Components
  - cc-backend
  - cc-metric-collector
  - cc-metric-store
- Open-source MIT license

You must provide adapter, e.g.:
- Slurm commands
- Slurm REST API
- PrEp Plugin

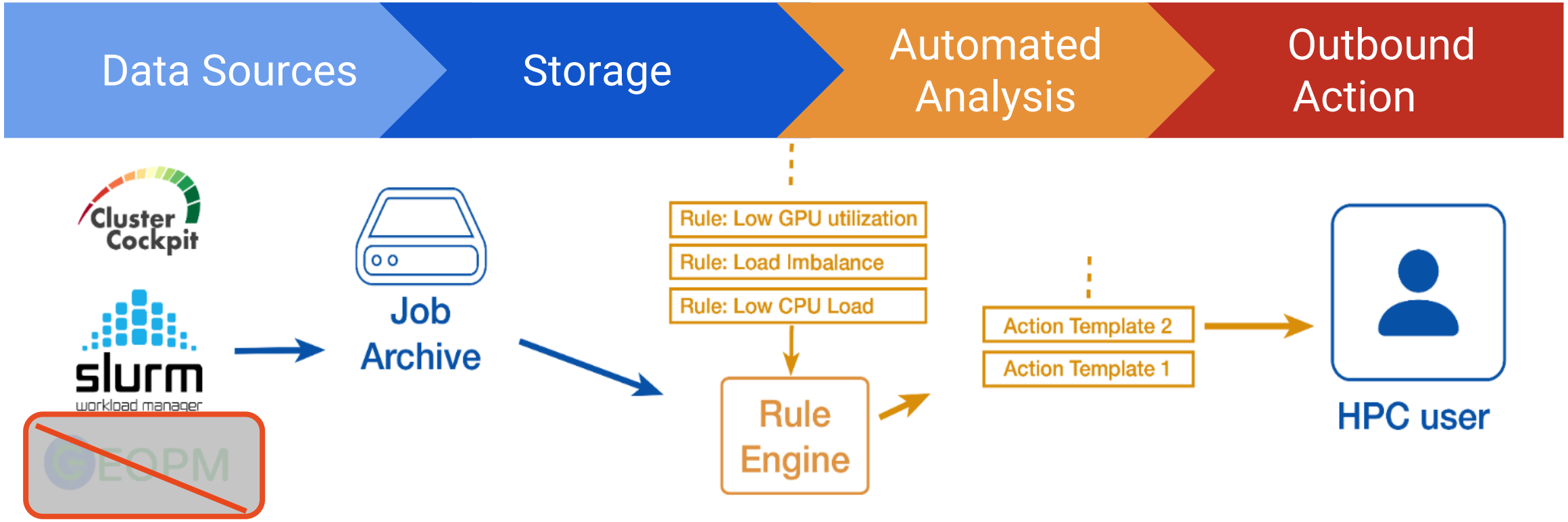


https://github.com/ClusterCockpit/

Patho-Jobs

- Goal:
  - automated rule-based detection system
  - for known pathological HPC jobs (jobs with previously encountered inefficiencies),
  - for cluster operators and users

https://git-ce.rwth-aachen.de/pathojobs/



- Cluster Cockpit generates a job-archive for each job;
  - alternative sources experimented with, but not ready to use
- Rule-Engine applies anti-pattern and triggers appropriate response for each job
- Detected anti-patterns trigger action templates (Email or job-archive tags)

Patho-Jobs: example pattern

```
{
 "name":"Low CPU load",
 "tag":"lowload",
 "parameters": ["threshold_factor"],
 "metrics": ["cpu_load, job"],
 "terms":[
 { "load_mean":    "cpu_load.mean('all')"},
 { "load_threshold":"job.numHwthreads * threshold_factor"},
 { "lowload_nodes": "load_mean < load_threshold"},
 { "lowload":      "lowload_nodes.any('all')"},
 { "load_perc":    "1.0 - (load_mean / load_threshold)"}],
  "output":"lowload",
  "output_scalar":"load_perc",
  "template":"Job ({{job.jobId}}) was detected as the mean cpu
  load {{load_mean}} falls below {{load_threshold}}."
}
```

Tag for use in ClusterCockpit

External configurable threshold

Which datapoints to pull from job-archive

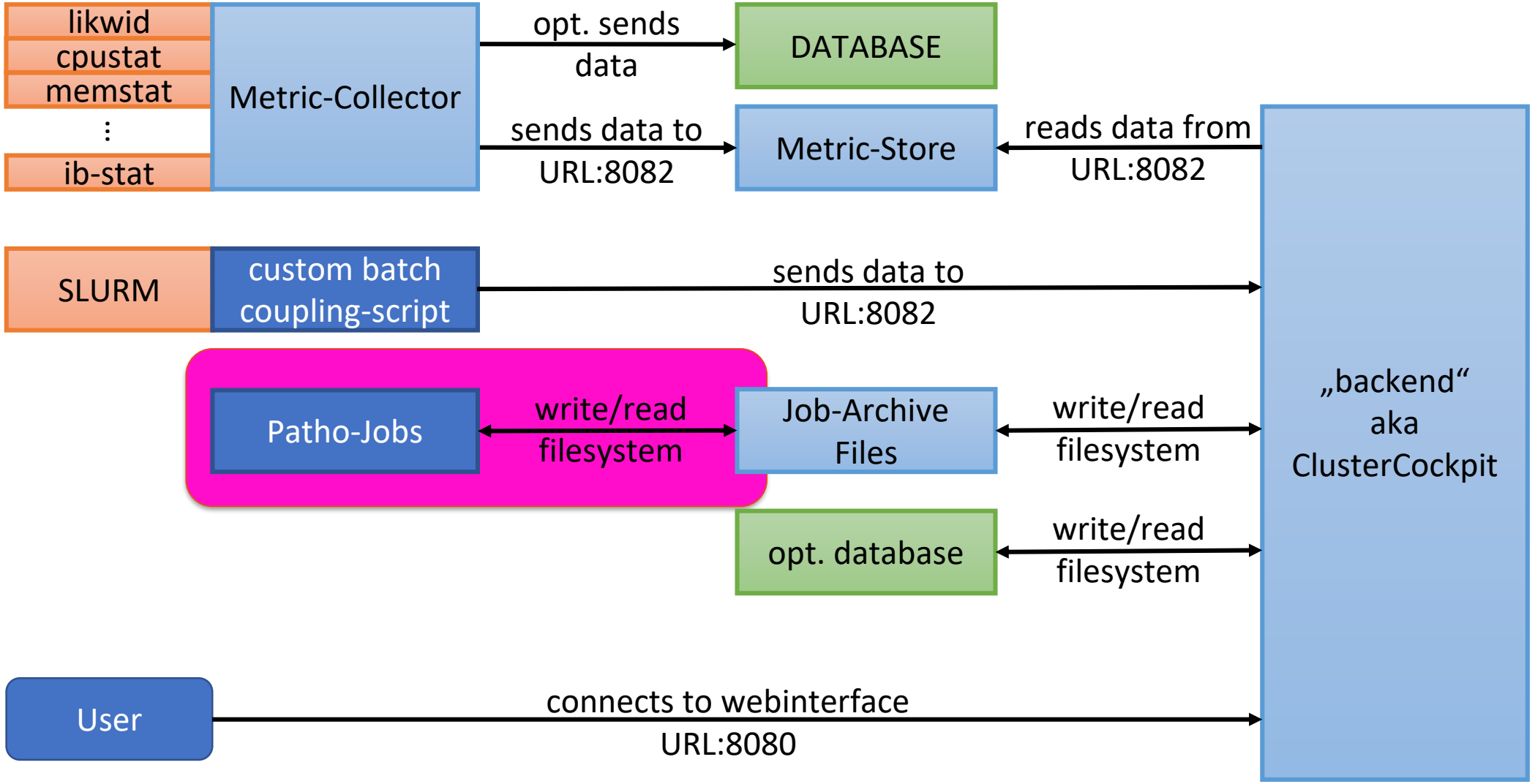
Variables initialized with in sequence evaluation of terms

Pattern output, variables can be used

PathoJobs: Prototype antipatterns

- Low CPU load
- Load-Imbalance
- Job Payload Overhead
- Excessive CPU load
- Problematic memory usage
- Uncoordinated multi-process GPU-usage
- Low GPU utilization
- Unnecessary job distribution
- Unnecessary PFS read

Patho-Jobs extension:



- Deployment:
  - The “PathoJobs” system is deployed by partners on production HPC systems.
  - Each processes job-archives from hundred-thousands to millions of jobs per month.
  - Current performance anti-patterns are evaluated for all HPC jobs.

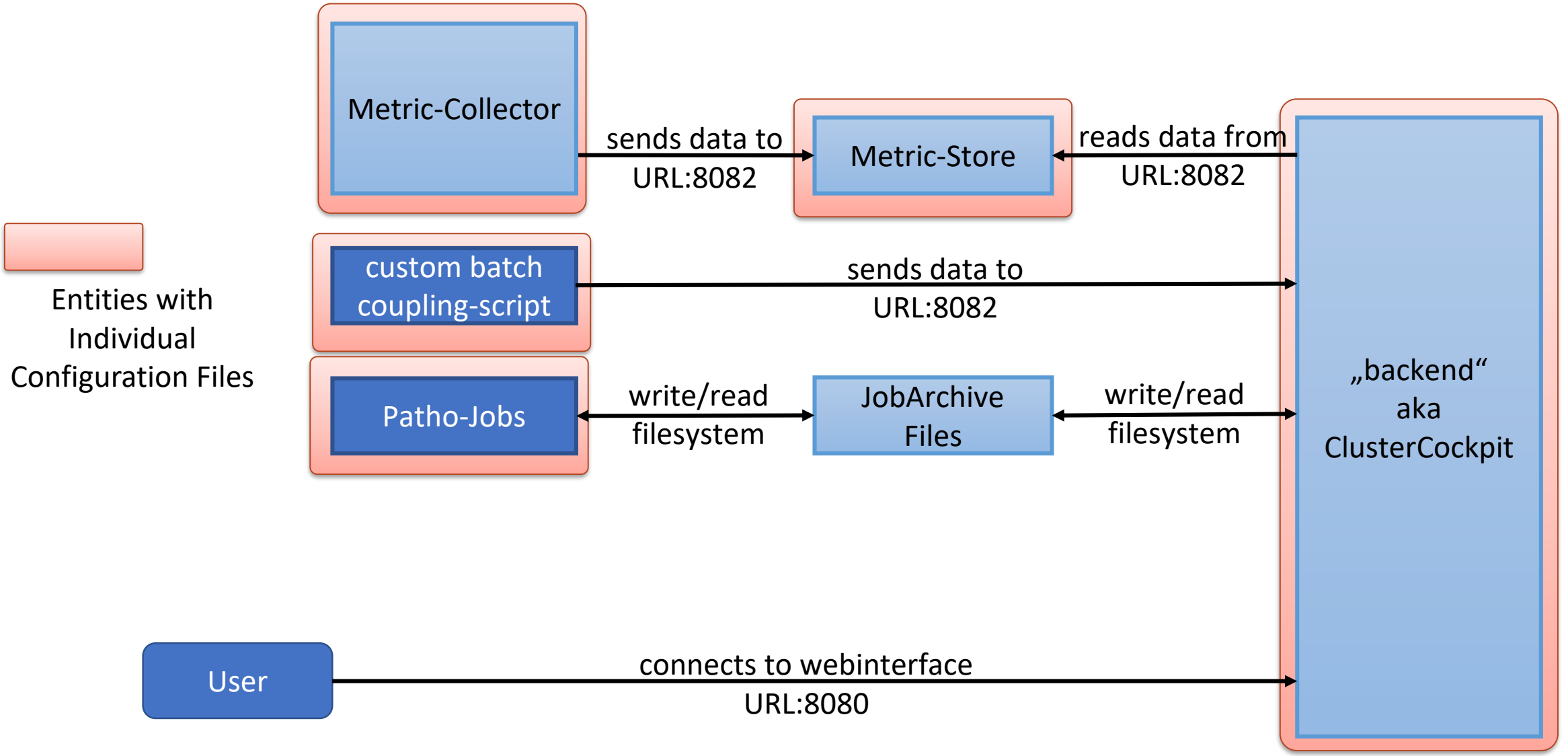
- First Experience & Insights:
  - Rule evaluation does not exhibit noticeable extra-load on the Cluster Cockpit hosts
  - Number of jobs with detected inefficiencies exceeds acceptable rates
  - System is able to detect explicitly malformed test-jobs

Location	Jobs / month	Jobs with issues	Rules with highest hit-rate	Analysis cost
TU Darmstadt	~ 250k	598 / 1000	Low CPU load Load-imbalance	230 ms / job
Paderborn	~ 190k	180 / 1000	Low CPU load CPU oversubscription	560 ms / job
FAU	~ 255k	326 / 1000	Low CPU load Low resource utilization	n.a.

Experience …

- Software can be directly deployed from binaries:
  - Did not attempt to compile on HPC system
  - 3 types of services:
    - metric-collector per compute node,
    - at least one metric store
    - backend/UI service
- Basic setup:
  - cluster configuration: specification of the cluster for the backend; **tools help**
  - metric collector: select which metrics are of interest, **challenging**
    - managing CPU registers, trade-off available metric sources, manage access priviliges
  - metric store: holds metrics; must match with metric collector and backend; assertaining peak capacity is not trivial

Individual configuration files:



Take-away

- Powerful tools:
  - ClusterCockpit:
    - Cluster-wide continuous monitoring
    - UI for job monitoring, early detection of performance or runtime issues, easy access to aggregate statistics for users and jobs
  - PathoJobs:
    - Automation of detecting specific jobs
    - Scriptable
- Initial setup is fairly easy,
- but …
  - metric selection requires expertise
  - one must maintain consistency across multiple configuration files

Take-away

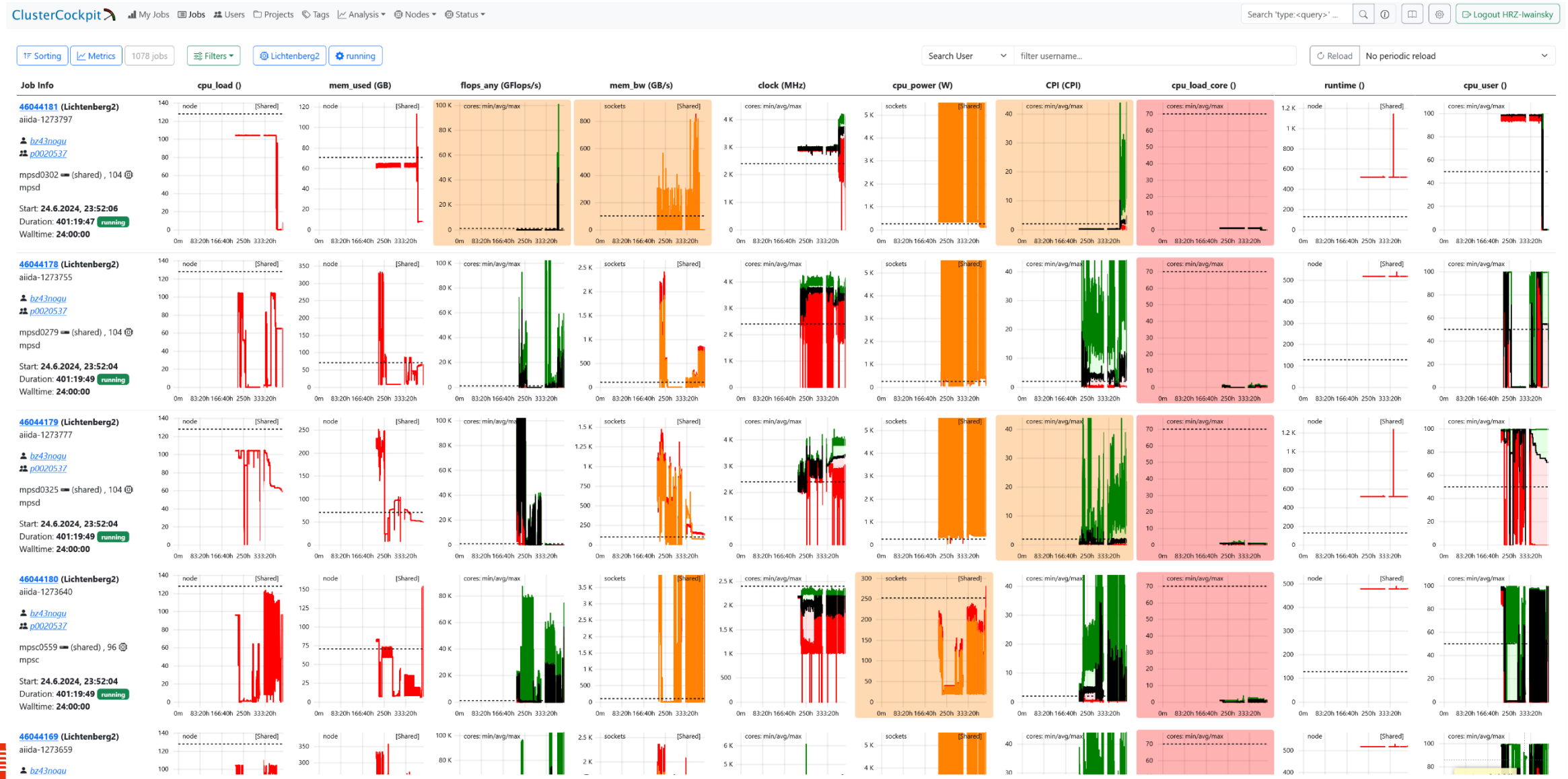
- Powerful tools:
  - Cluster Cockpit:
    - Cluster-wide continuous monitoring
  - PathoJobs:
    - Automation of detecting specific jobs
- Requires some expertise and effort to setup

Questions?

BACKUP SLIDES!

For your information

Joblist-View

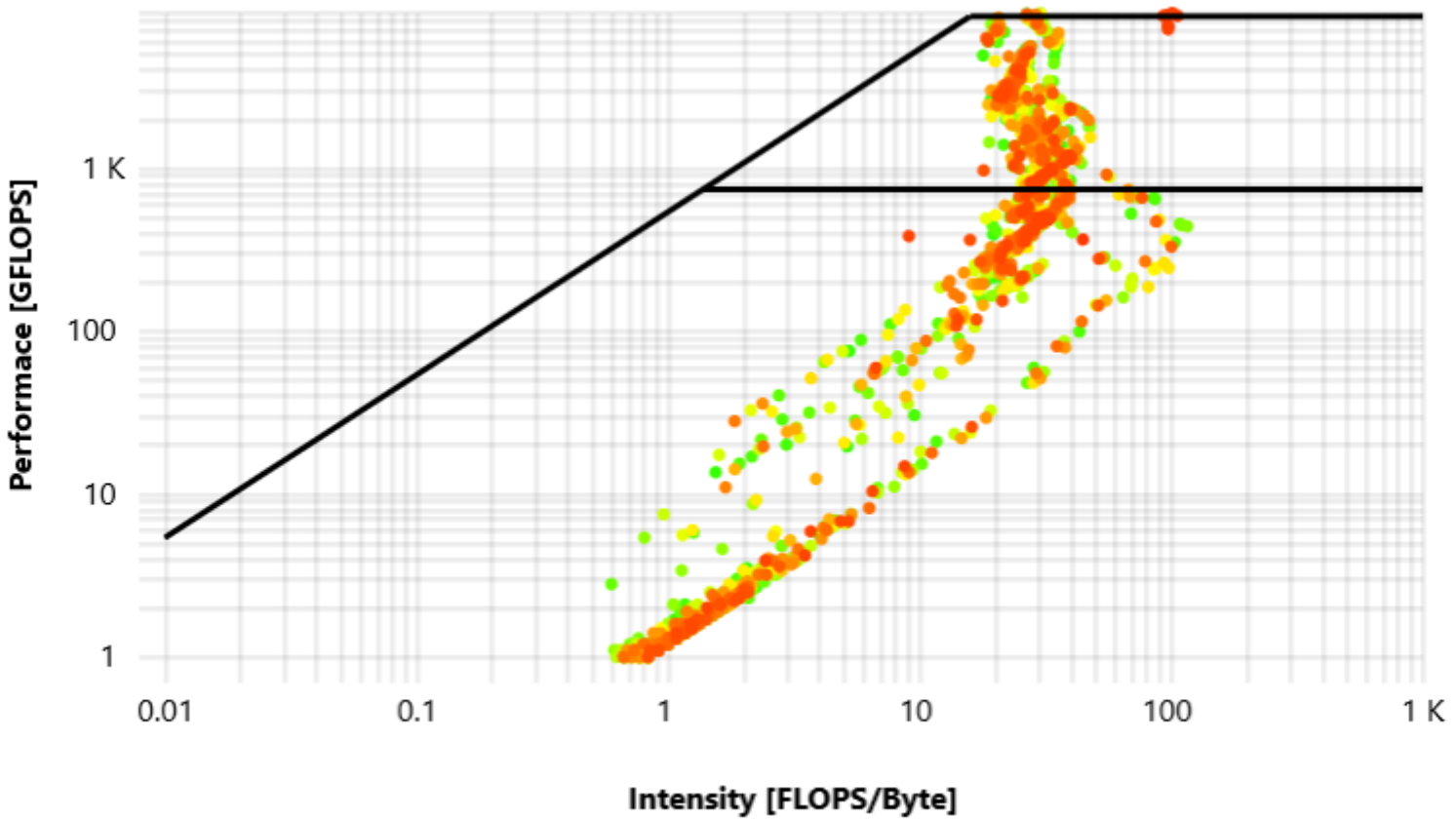


Job-View

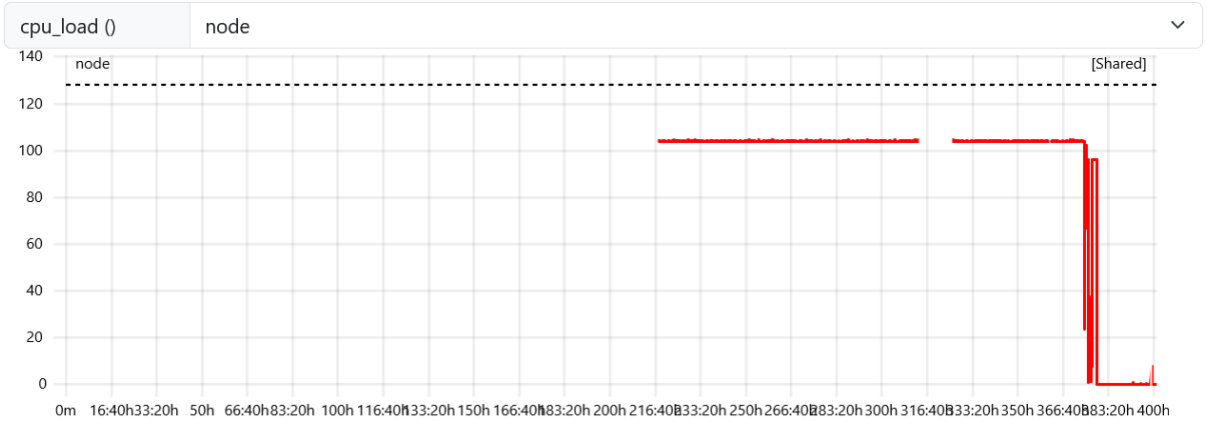
Roofline-Model,
Metrics &
Counters,
Assessment

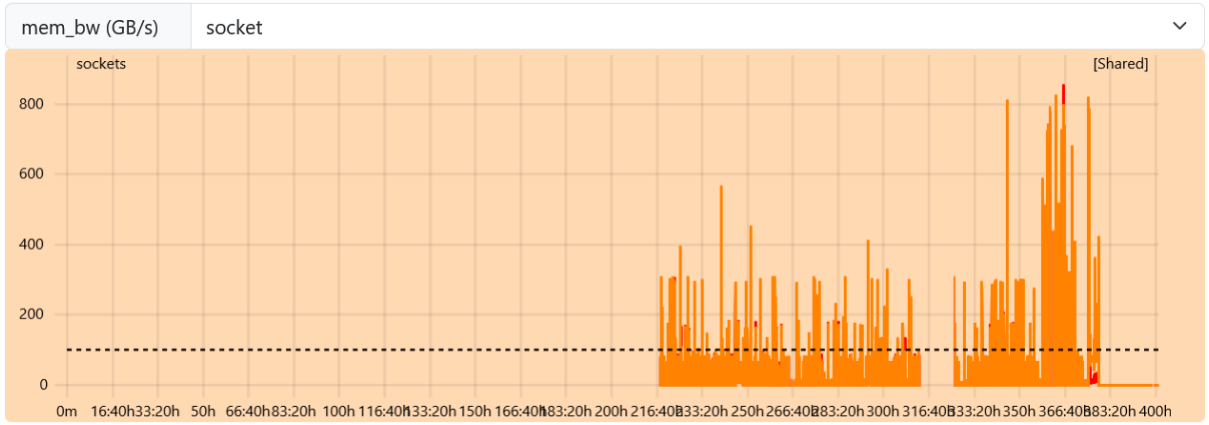
SLURM Info,
Core-overview,
SLURM Script

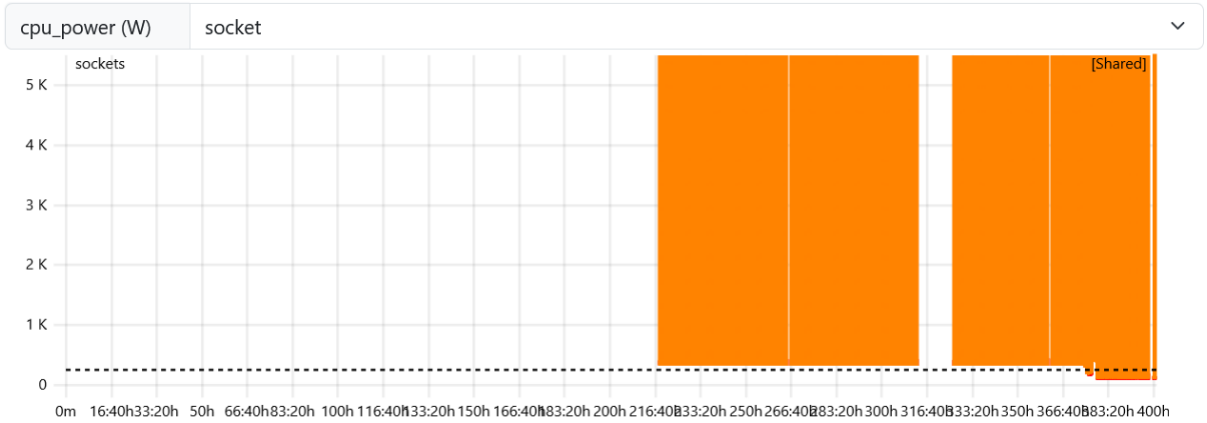
Roofline-model

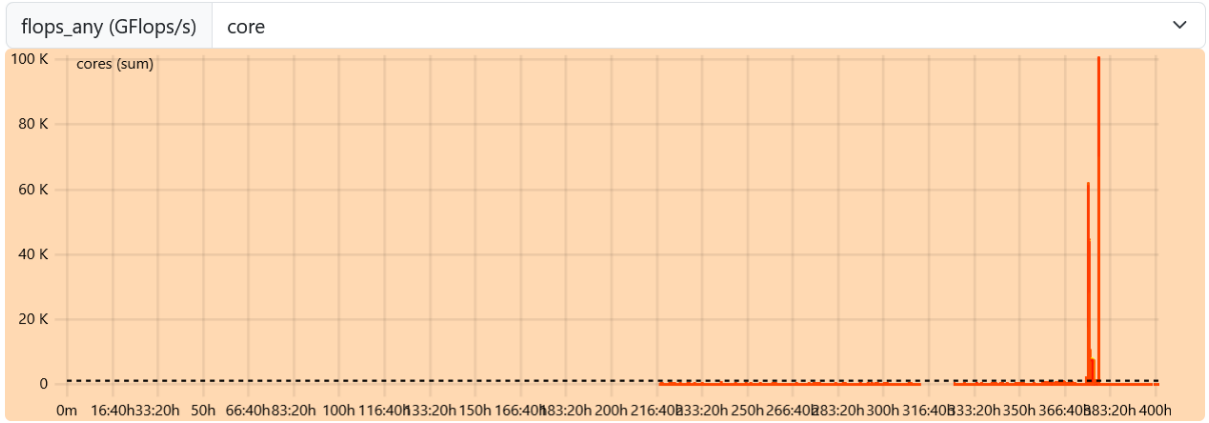


Ressource views









Job info: statistics table

Statistics Table | Job Script | Slurm Info

Metrics | flops_any | core

Node	Id	min	avg	max
mpsd0302	Sort ▼	Sort △	Sort △	Sort △
	0	0	61.67	72868.5
	1	0	57.49	41655.1
	2	0	59.1	43100.7
	3	0	63.93	72276.8
	4	0	59.38	43910.2
	5	0	61.34	72743.5
	6	0	62.84	72668.9
	7	0	63.18	71742.2
	8	0	59.24	51420.9
	9	0	63.13	72197.4
	10	0	60.88	71800.1
	11	0	53.14	31386.9
	12	0	59.64	63683.8
	13	0	56.34	46340.9
	14	0	62.37	58119.6
	15	0	63.84	70981.1
	16	0	62.41	70623.4
	17	0	63.28	70222.7

mem_bw | socket

Id	min	avg	max
Sort ▼	Sort △	Sort △	Sort △
0	0.4	19.7	854.3
1	0.4	22.39	824.3

Metric Collector

- Node measurement agent
- Collects metrics every N - seconds
- Measurements are transmitted to one (or multiple sinks) every M – seconds
- Available sinks:
  - stdout: Print all metrics to stdout, stderr or a file
  - http: Send metrics to an HTTP server as POST requests
  - influxdb: Send metrics to an InfluxDB database
  - influxasync: Send metrics to an InfluxDB database with non-blocking write API
  - nats: Publish metrics to the NATS network overlay system
  - ganglia: Publish metrics in the Ganglia using the gmetric CLI tool
  - libganglia: Publish metrics in the Ganglia directly using libganglia.so
  - prometeus: Publish metrics for the Prometheus Monitoring System
- Transformers can „update" data before transmitting

Available Metric Types

- cpustat
- memstat
- iostat
- diskstat
- loadavg
- netstat
- ibstat
- ibstat_perfquery
- tempstat
- lustrestat
- likwid
- nvidia
- customcmd
- ipmistat
- topprocs
- nfs3stat
- nfs4stat
- cpufreq
- cpufreq_cpuinfo
- numastats
- gpfs
- beegfs_meta
- beegfs_storage
- rocm_smi

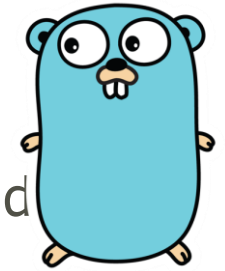
Metric-Store & Backend

- **Metric-Store:**
  - Ring-buffer for storing cluster metrics, for use in the backend
    - Size
    - Retention period
    - Backups every n-hours
  - Ressource intensive:
    - Each node * Time * Number-Of-Metrics
- **Backend:**
  - Webserver for interactive exploration
  - Aggregation of data to JOB-Archives

Technology stack

- Set of JSON schema specifications (config, cluster, job, metric naming) that serve as a blueprint for API payloads, data structures, file formats

- Golang programming language used for all components
  - Pragmatic and complete tooling environment (build system, LSP, formatting, package manager, testing, profiling, and debugging)
  - Mature and complete standard library
  - Powerful native support for threading
  - Static binary with file system embedding allows for easy configuration management and deployment



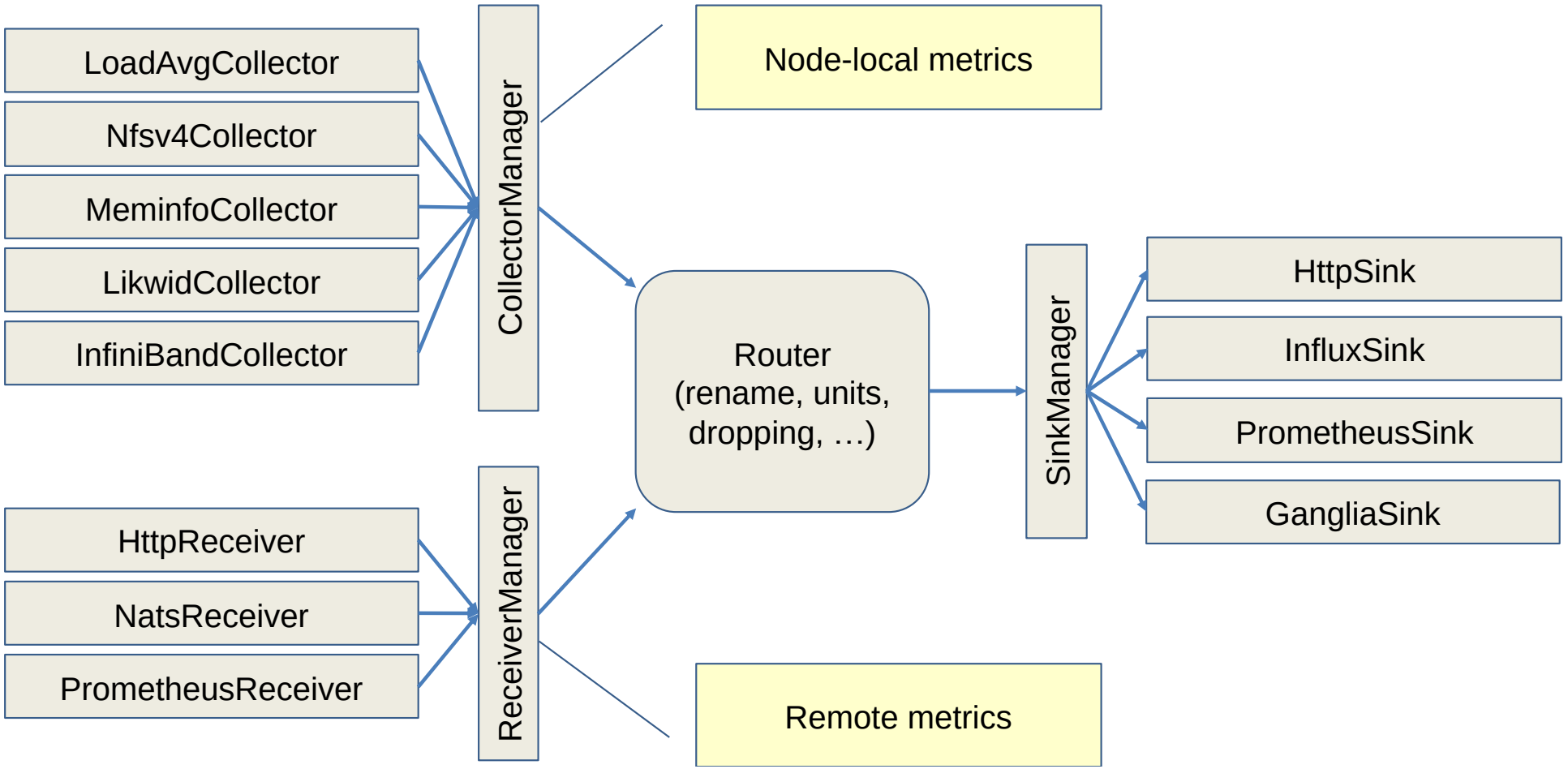
- Web frontend UI implemented using Svelte components

Details cc-metric-collector

- Daemon running on every compute node
- HPM metrics measured using LIKWID C library
  - Requires root rights at some point (either run daemon as root or use LIKWID access daemon)
  - Linux perf_event kernel interface untested but doable

- Potential interference with regular jobs
  - Test with Linpack (very demanding) benchmark with and without monitoring
  - If there is no measurable difference, then there is no influence
  - Most jobs are underutilizing the resources, it doesn't matter if something else is running
  - The benefit to identify faulty or badly performing jobs compensates any potential influence

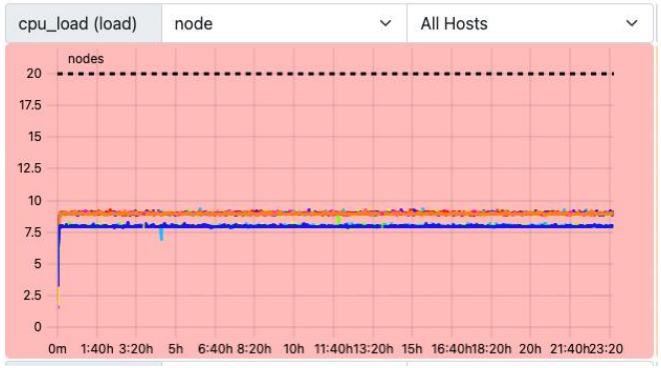
Software architecture cc-metric-collector



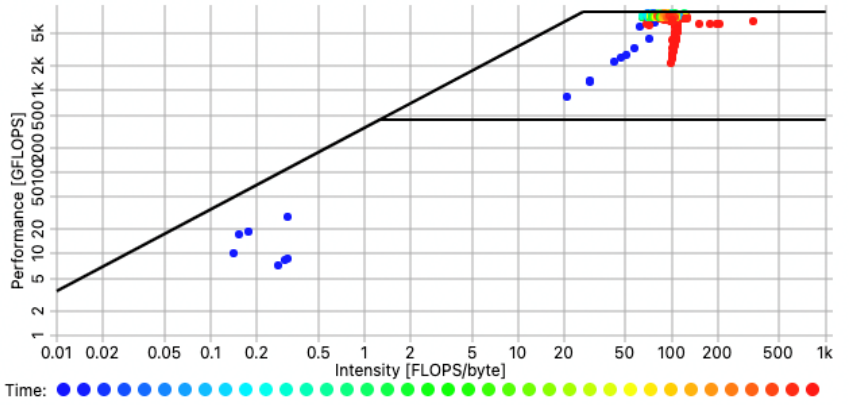
Uses a flavor of the InfluxDB line protocol as internal data format and external payload

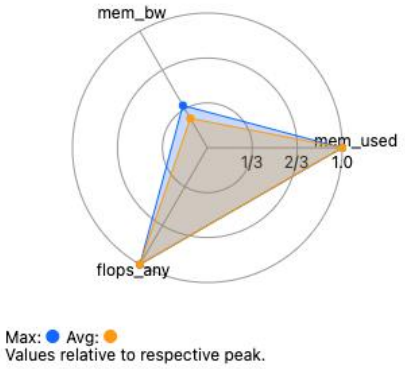
How to detect bad jobs

- Load metric indicates allocation or placement issues



- Basic resource utilization (arithmetic and memory bandwidth) can be seen in the roofline and polar plots





Technology stack

- Set of JSON schema specifications (config, cluster, job, metric naming) that serve as a blueprint for API payloads, data structures, file formats

- Golang programming language used for all components
  - Pragmatic and complete tooling environment (build system, LSP, formatting, package manager, testing, profiling, and debugging)
  - Mature and complete standard library
  - Powerful native support for threading
  - Static binary with file system embedding allows for easy configuration management and deployment

- Web frontend UI implemented using Svelte components

Software architecture

- Components can also be used stand-alone
- Agnostic regarding batch job scheduler (you need to provide adapter)
- Extendible architecture using interfaces
- In **`cc-backend`** for
  - Job Archive (file based, in work: S3 object store, elastic search DB)
  - Metric data backend (cc-metric-store, InfluxDB, Prometheus)
  - Authentication (Local DB account, LDAP, JWT token, Open ID Connect)

- In **`cc-metric-collector`** for
  - Metric collector (24 collectors including e.g. LIKWID, NVIDIA, GPFS)
  - Metric receiver (NATS, Prometheus, HTTP, IPMI, Redfish)
  - Metric sink (HTTP, InfluxDB, NATS, Ganglia, Prometheus)

Project governance and sustainability

- NHR@FAU is the development lead
  - Jan Eitzinger (permanent full-time position; project management, **`cc-backend`**, **`cc-metric-store`**)
  - Thomas Gruber (permanent full-time position; LIKWID, **`cc-metric-collector`**, **`cc-node-controller`**)
  - Christoph Kluge ((soon permanent) full-time position; **`cc-backend`**, web frontend)
  - Amritanshu Verma (BMBF EE-HPC full-time staff member; **`cc-backend`**)
  - Aditya Ujeniya (BMBF EE-HPC full-time staff member; **`cc-metric-store`**, **`cc-energy-manager`**)
- But independent GitHub Project to create joint development community
  - Contributions per pull-request: PC2 Paderborn (Authentication), DKRZ (Prometheus backend), KIT (cc-metric-collector)

Road map

- S3 backend for Job Archive
- Add energy metrics and job energy footprint
- Automatic job tagging (application detection and job classification)
- Generic support for message brokers (with NATS as default)
- New components: **`cc-node-controller`** and **`cc-energy-manager`**

What we can’t provide

- We are not a company that you can pay to implement things
  - But we can help and support you to do it yourself
  - If multiple sites are interested in a feature, you can do it together

- We cannot provide commercial support
  - But you can open issue tickets at GitHub
  - There are two Matrix chat rooms to get quick help

With ClusterCockpit you have full control, but you also must take responsibility yourself.