Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
terabit

Advancements in Monitoring Infrastructure at INFN-TO Computing Centre

Workshop sul calcolo nell'INFN

La Biodola - Isola d'Elba

27/05/2025

Nicola Mosco
INFN TO

Outline

- Computing resources in Turin
- Monitoring infrastructure
- Dashboard design
- Machines database
- Future developments

Computing resources & project synergy

CTLab *(Computing Technology Lab)*:

- 4×Dell PowerEdge R7525:
  - 2×(AMD EPYC 7313 3.0 GHz 16 cores)
  - 1TB RDIMM, 3200 MT/s
  - 2×(AMD MI100, 32GB)
  - 1×(NVIDIA A16, 4×16GB)
  - 2×(NVIDIA A5000, 24GB)
- 3×Lenovo SR675v3:
  - 2×(AMD EPYC 9124 3.0 GHz 16 cores)
  - 1TB RAM (expandable)
  - 2×(NVIDIA L40S, 48GB)
- 1×NVIDIA GraceHopper superchip, evaluation system by E4:
  - CPU+GPU coherent memory model
  - 900 GB/s interface NVLink-C2C
- Connectivity: InfiniBand + NVIDIA ConnectX-7 400 Gb/s

TeRABIT:

- 6×Lenovo SR665v3:
  - 2×(AMD EPYC 9654 96-Core Processor)
  - 1,5 TB RAM
- 6×Lenovo SR675v3:
  - 2×(AMD EPYC 9654 96-Core Processor)
  - 1,5 TB RAM
  - 4×(NVIDIA H100 80GB)
  - NVIDIA ConnectX-7 400 Gb/s
- 1×InfiniBand switch 400 Gb/s
- Connectivity: NVIDIA ConnectX-7 400 Gb/s

Synergic and consistent management computing resources

Monitoring infrastructure design

Machines and resources catalog:

- PostgreSQL (*the database*)

Metrics collection:

- Redfish
- SNMP

Time series database:

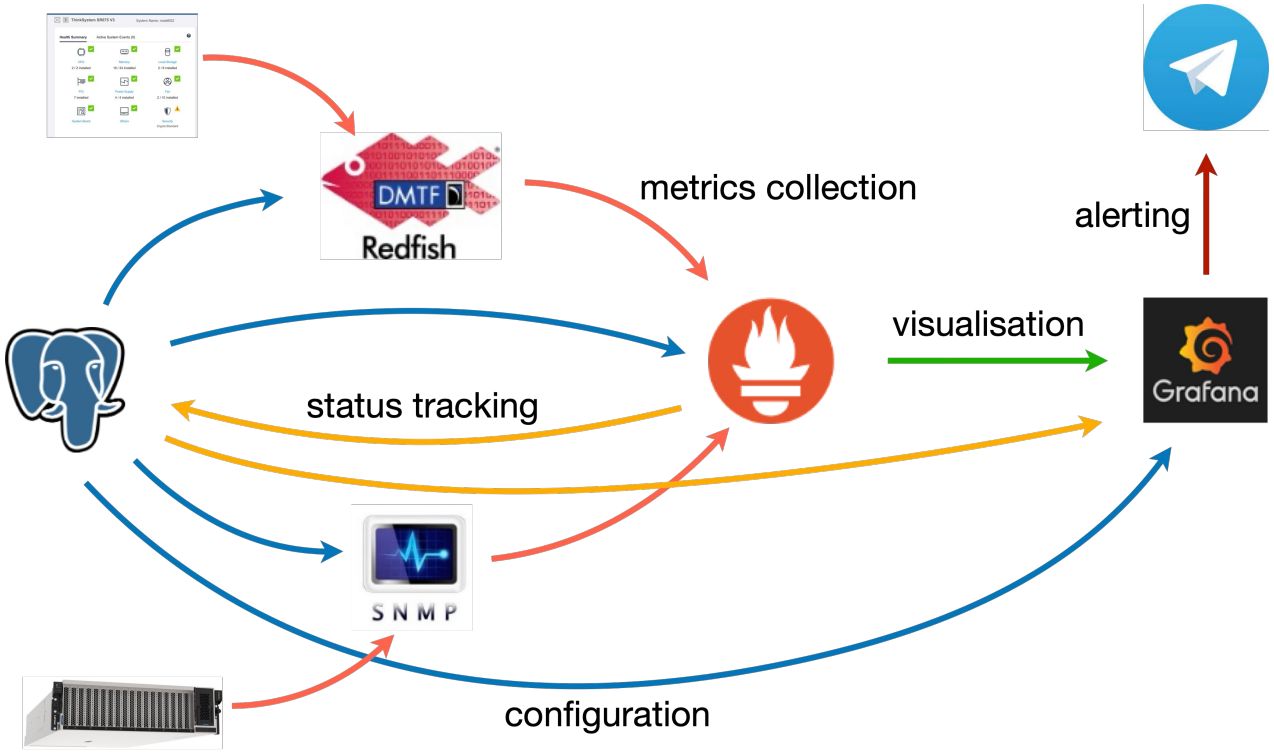
- Prometheus

Data visualization:

- Grafana

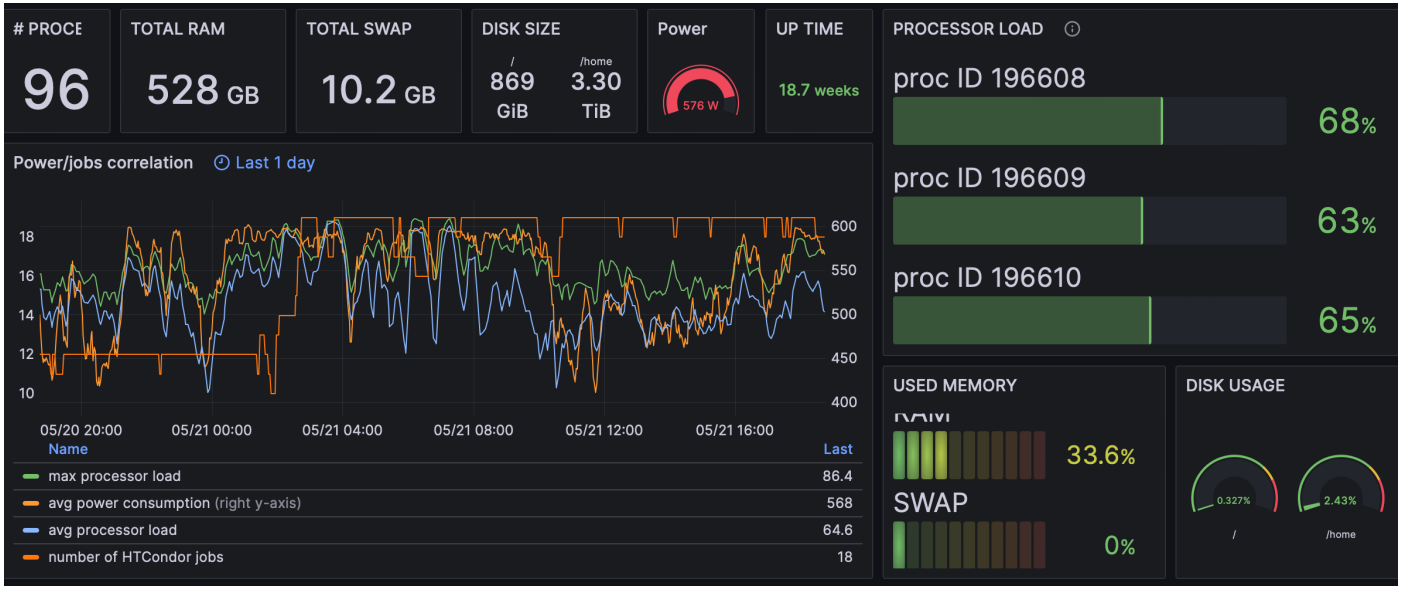
Alerting:

- Currently, Grafana + Telegram



What we had last year

- Basic metrics of machines: CPU, RAM, disk usage and power metrics.
- For HTCondor nodes, correlation between power consumption and number of jobs.



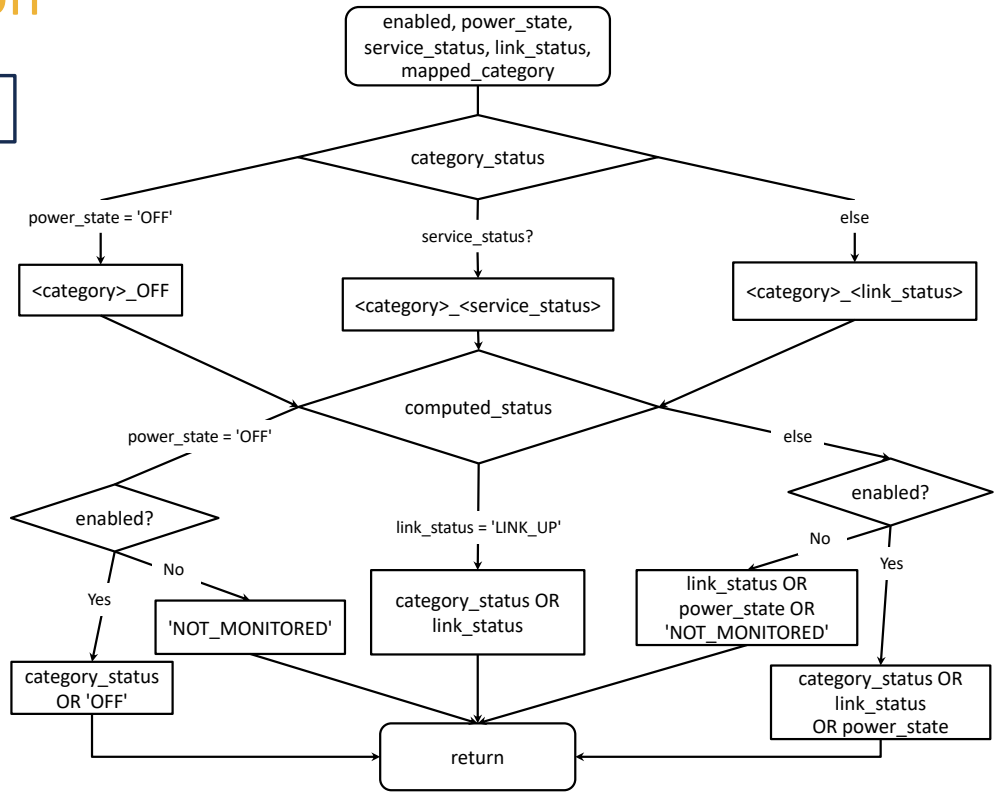
What we have now

- Visualise machines in their location.
- Node identification.
- Color codes for synthetic status display: role & status.
- States, categories, and codes are configured in the database.
- Monitoring status: ping, power state, HTCondor

Dashboard design: Status determination

```
category_status = mapped_category + status suffix
```

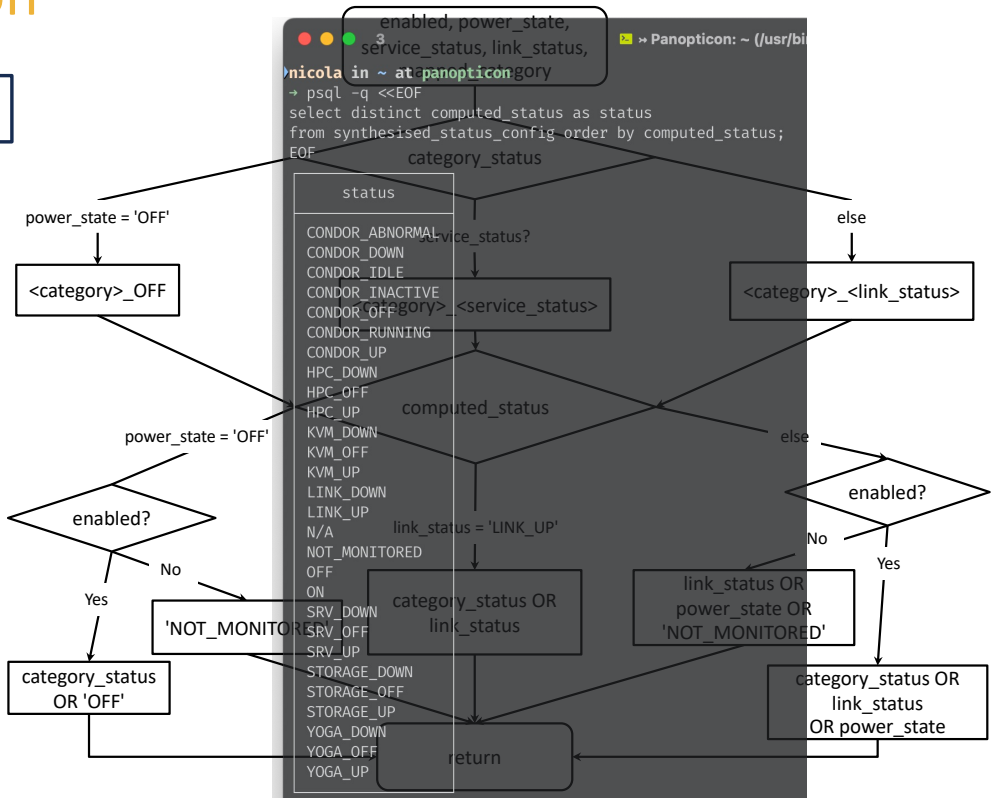
```
computed_status
```



Dashboard design: Status determination

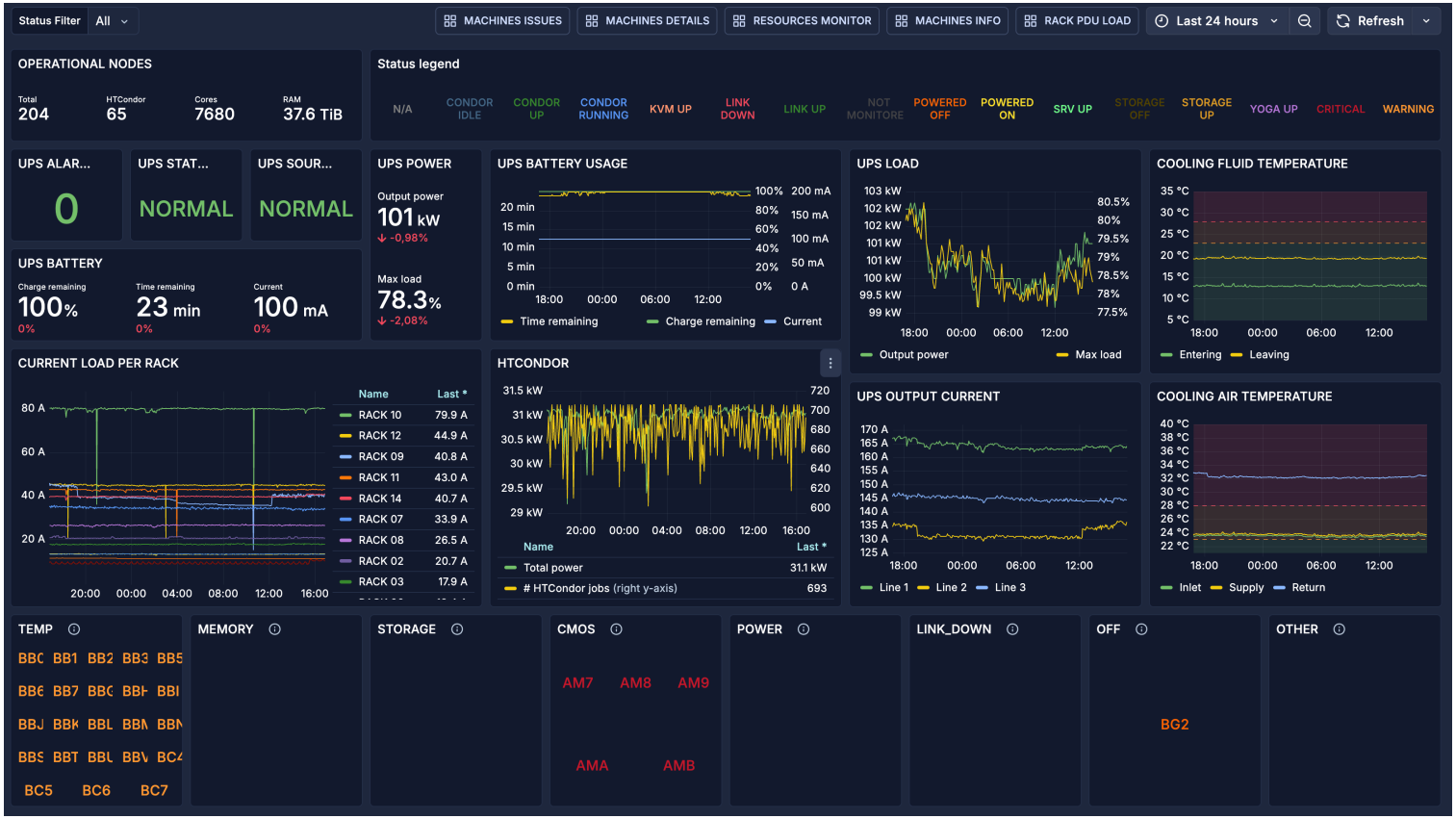
`category_status = mapped_category + status suffix`

`computed_status`



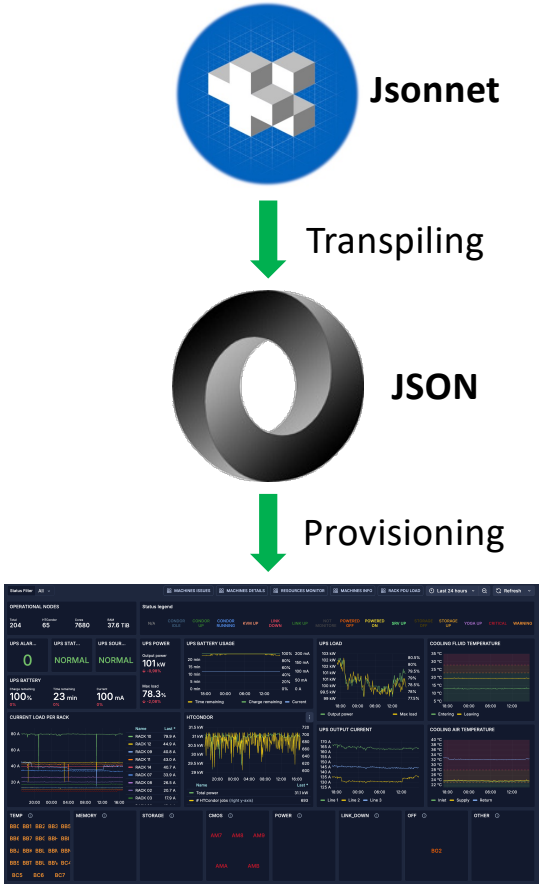
What we have now

- General status overview
- Resource usage
- Health issues detection



Dashboard design with Grafonnet

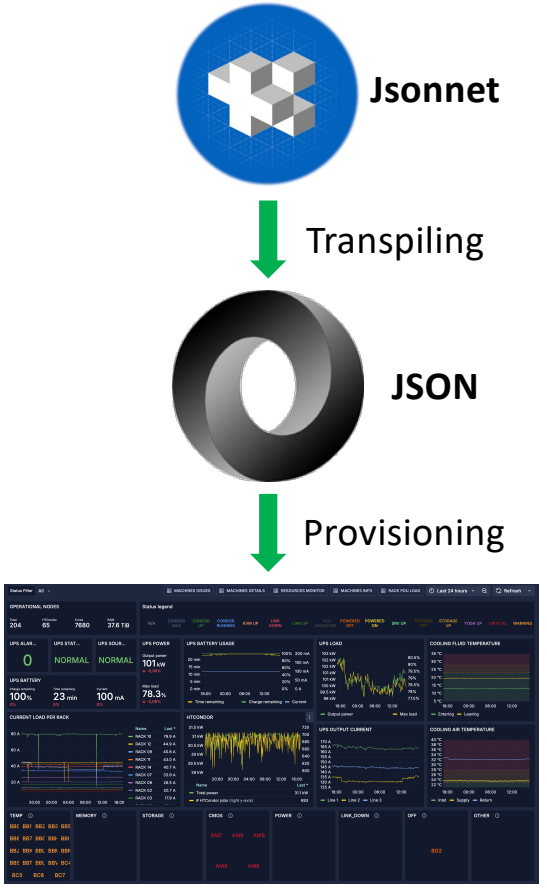
- Grafonnet: Jsonnet library for generating Grafana dashboards.
- Jsonnet:
  - Configuration language that *transpiles* to JSON (source to source translation, same abstraction level).
  - Functional style (no side-effects).
- Fit into pipeline for provisioning of dashboards and easier (wrt. JSON) versioning.
- All well and good, but... **double the effort!**
  - Grafonnet lags behind GUI features.
  - Need to rewrite GUI-generated dashboard to Jsonnet code.



Dashboard design with Grafonnet

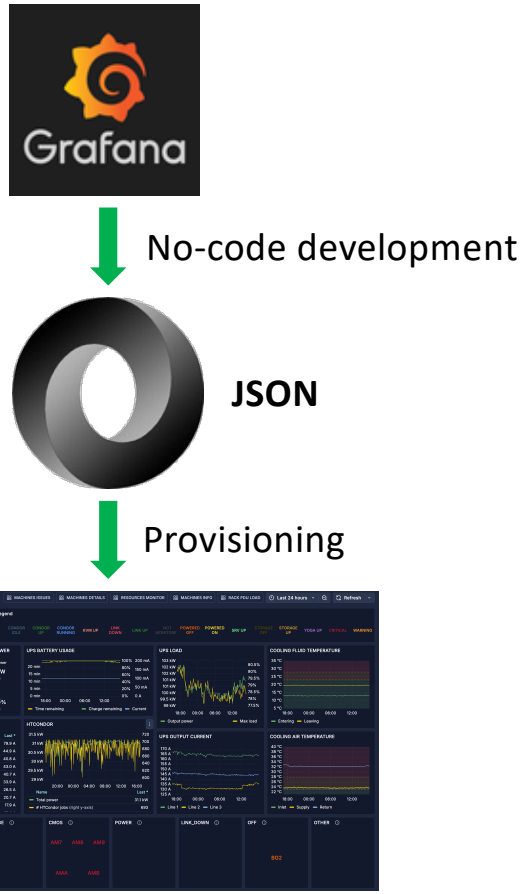
- Grafonnet: Jsonnet library for generating Grafana dashboards.
- Jsonnet:
  - Configuration language that transpiles to JSON.
  - Functional style (no side-effects).
- Fit into pipeline for provisioning of dashboards.
- All well and good, but... **double the effort!**
  - Grafonnet lags behind of GUI features.
  - Rewriting GUI with Jsonnet code.

Failed attempt!



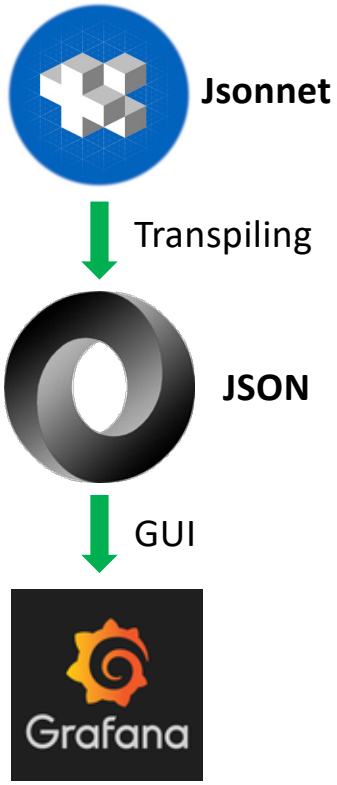
Dashboard design ~~with Grafonnet~~ *with GUI*

- Develop dashboards with GUI.
- Take JSON source.
- Provisioning pipeline.



Dashboard design *with Grafonnet*: Initial development

- Develop complex logic with Jsonnet.
- Take JSON source to define new dashboard in Grafana.
- Continue development with GUI.



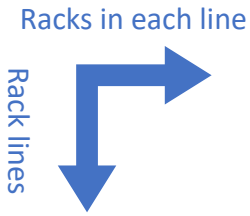
Jsonnet machinery useful for initial development!

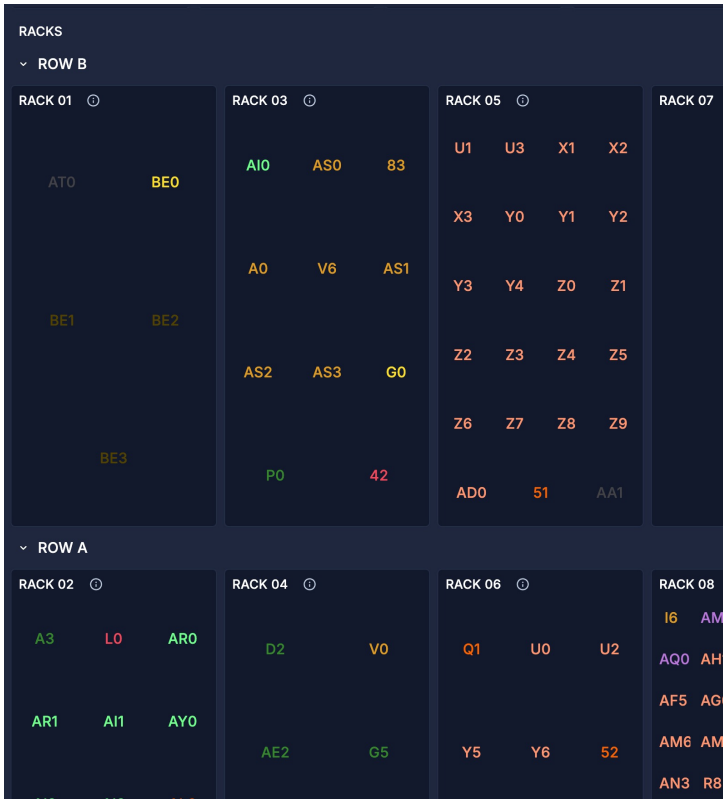
→ Static generation of complex layouts

Dashboard design *with Grafonnet*: Initial development

- Hard to deal with simultaneously-varying, correlated variables in Grafana GUI.
- There is no support (yet) for variables defined at row level.

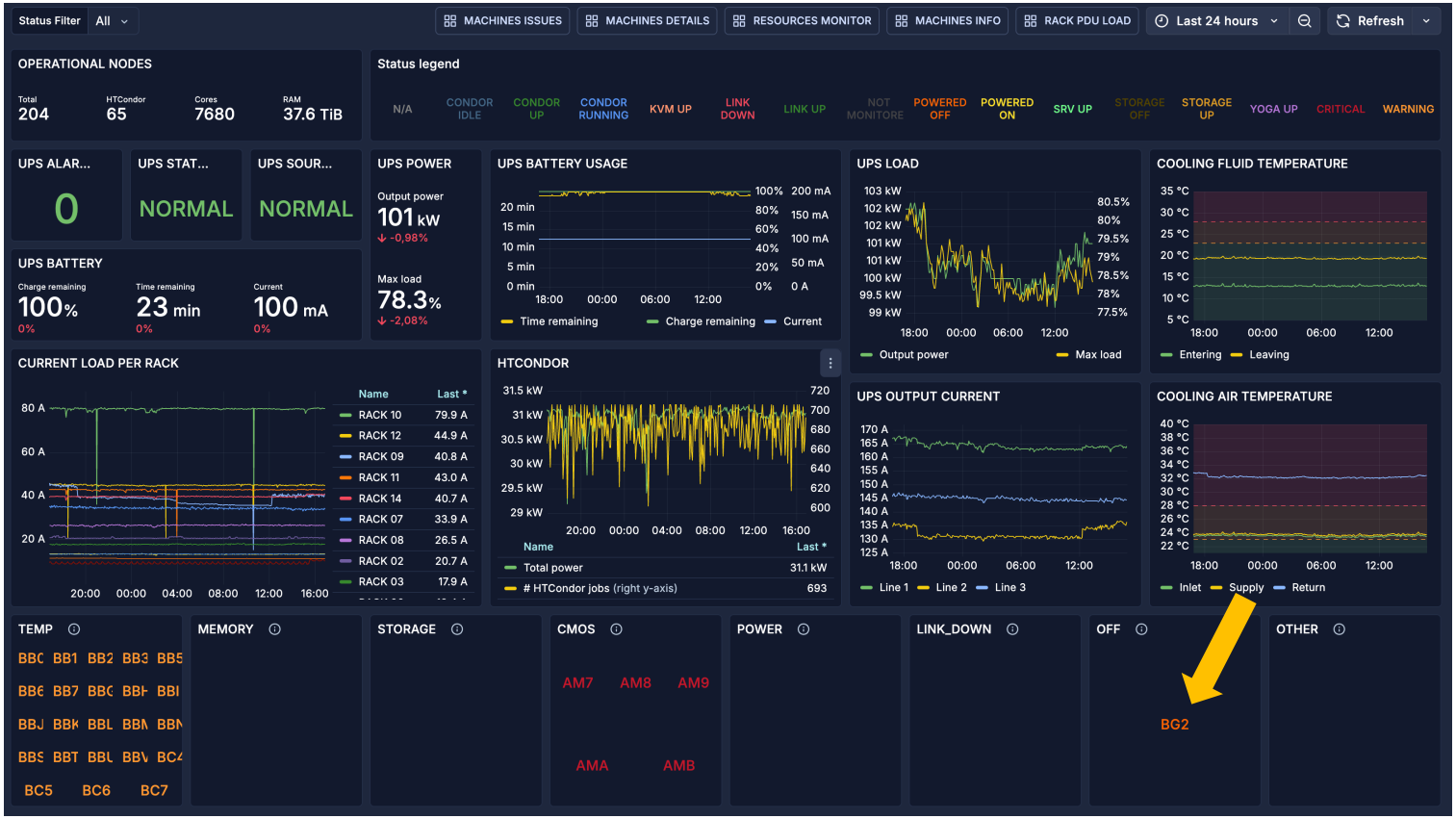
Use Jsonnet to create multiple variables depending on configuration, defined in the database.





What we have now

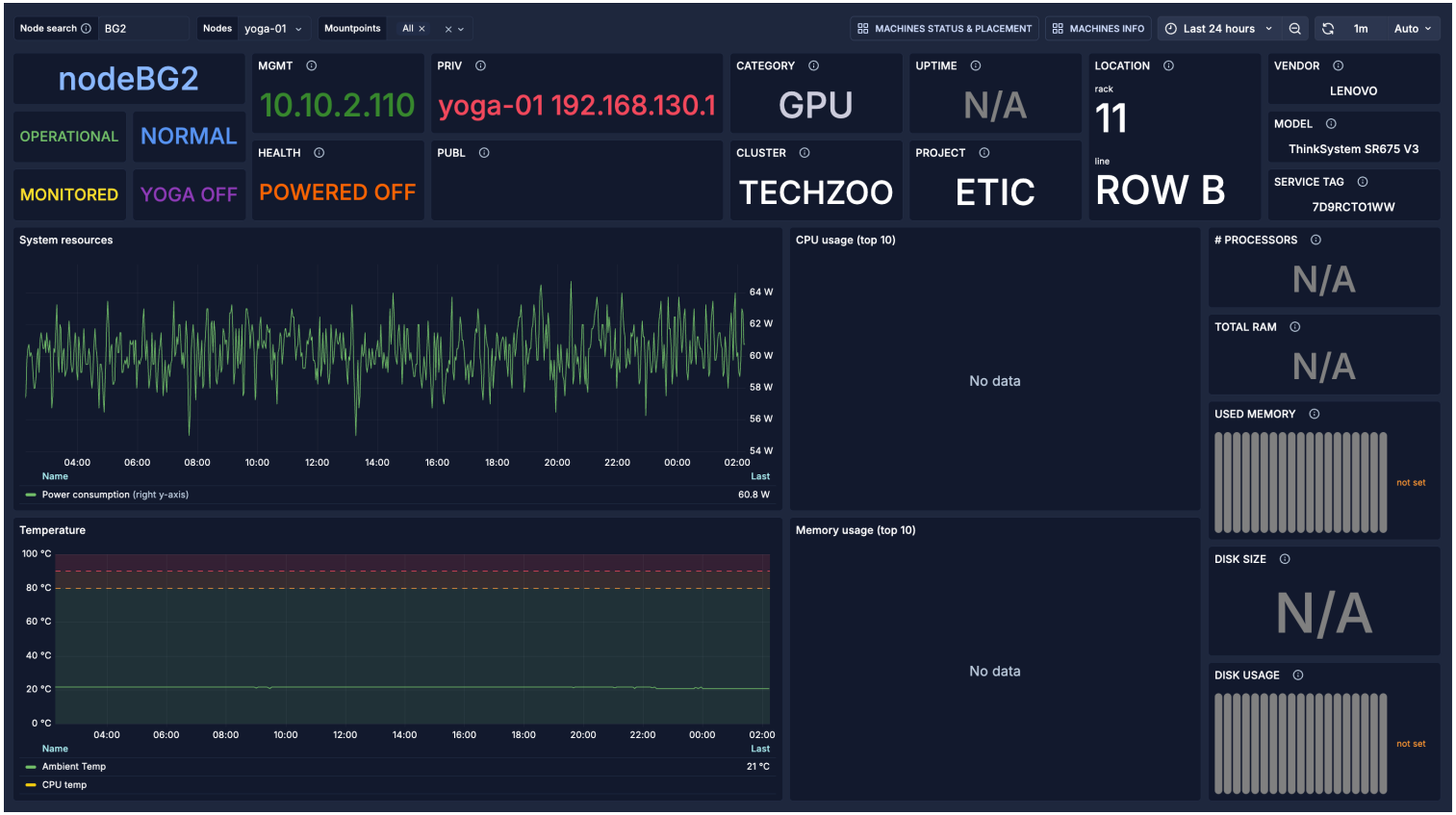
Machine reported as powered-off:
click to open the detailed view…



What we have now

Detailed machine view

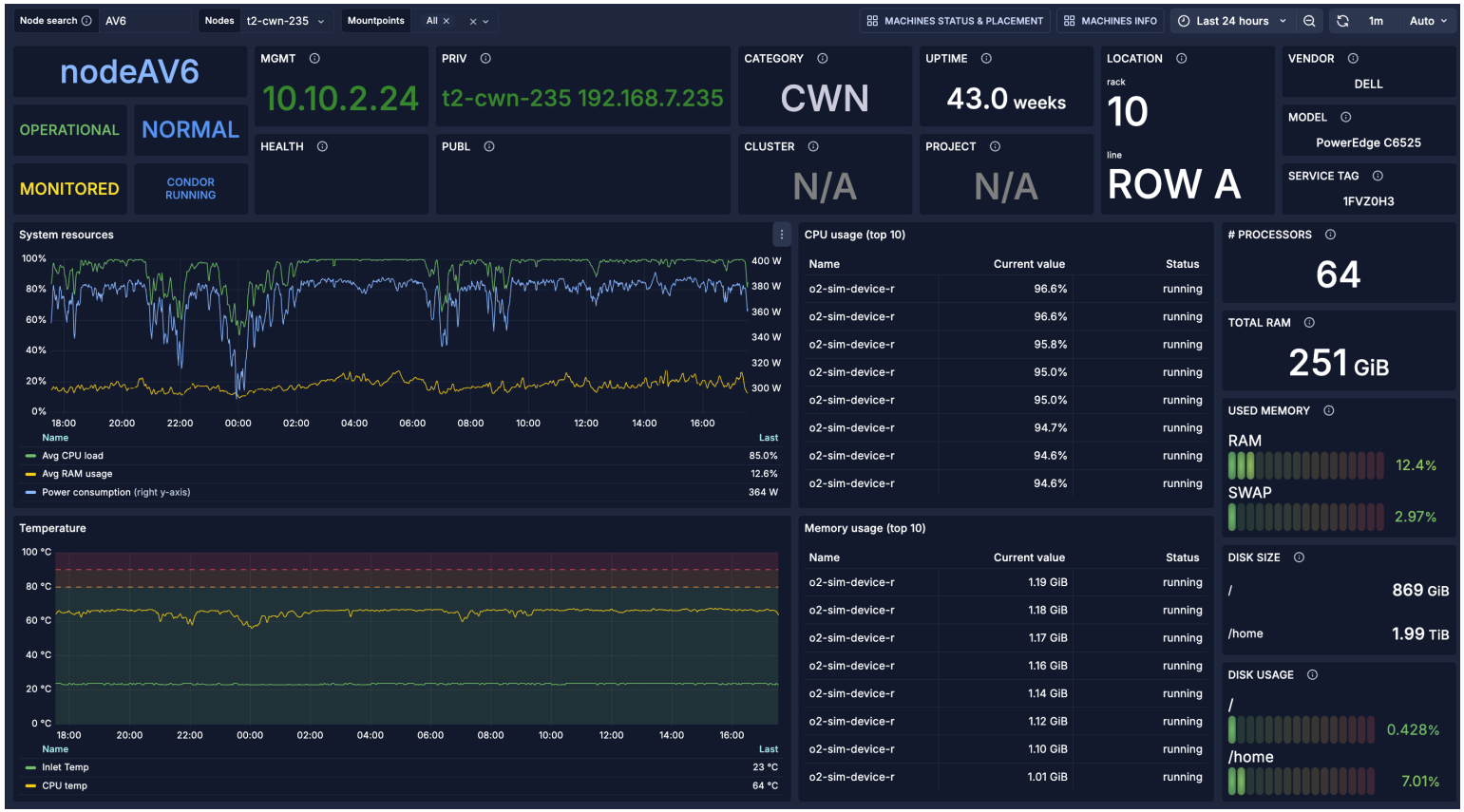
- Resource usage.
- Power and temperature metrics.
- Service state.
- Link to BMC page.



What we have now

Detailed machine view

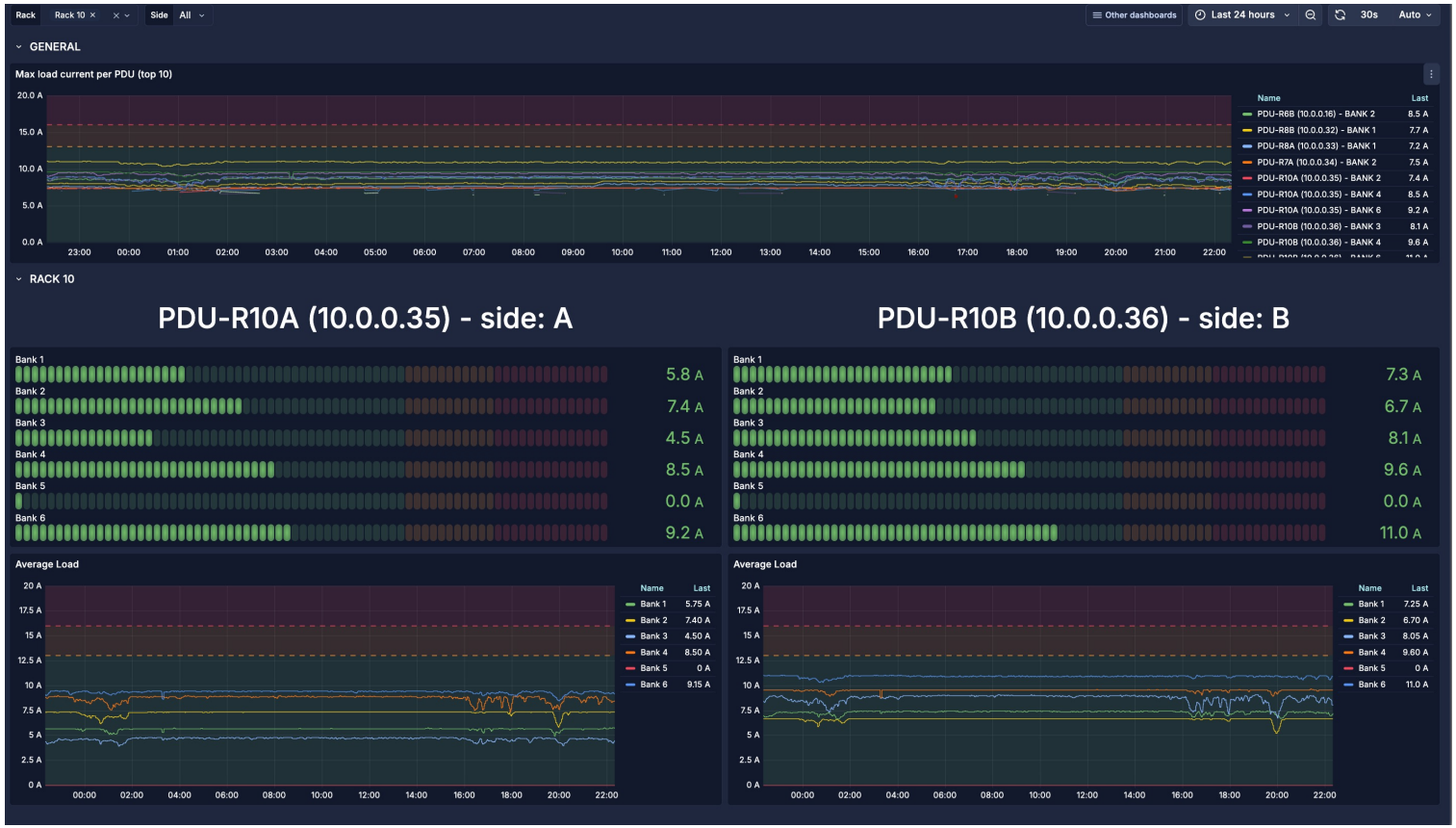
- Resource usage.
- Power and temperature metrics.
- Service state.
- Link to BMC page.



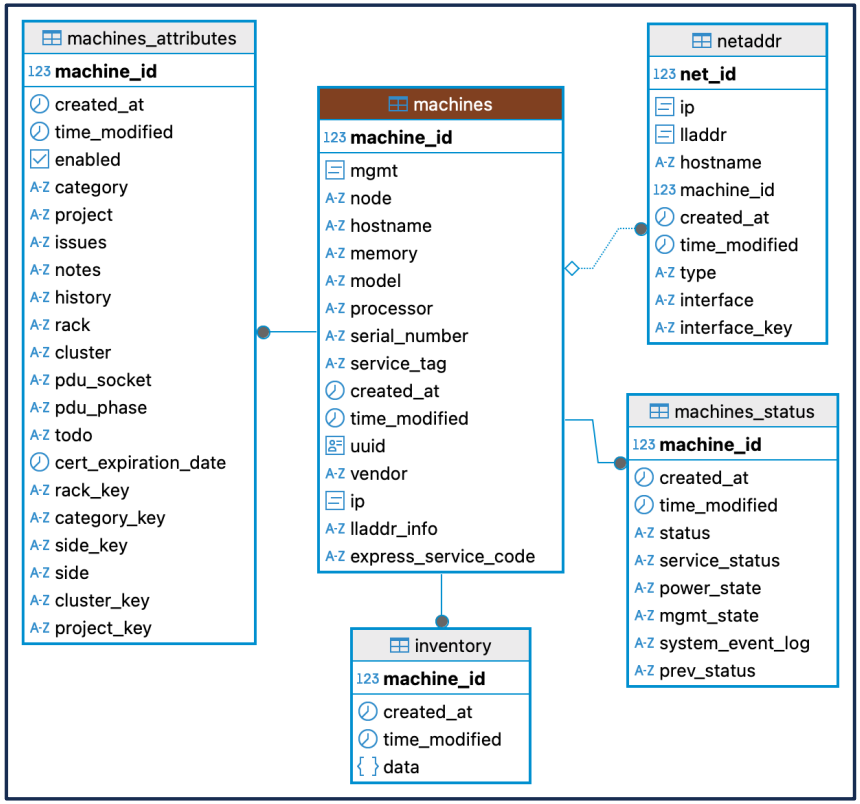
What we have now

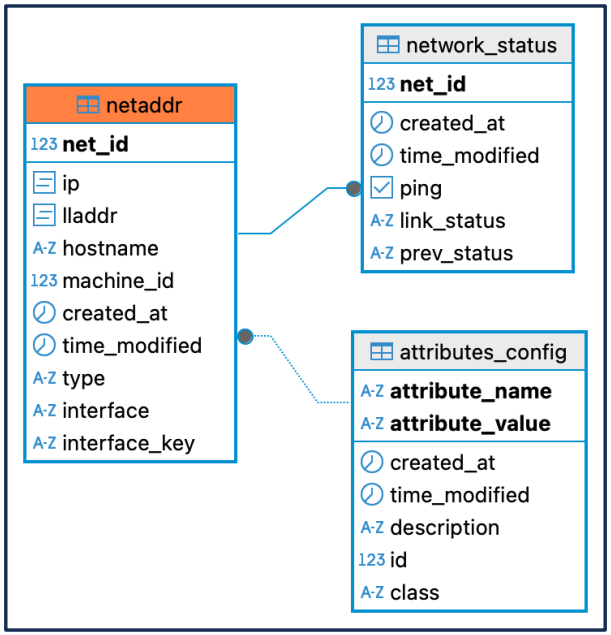
Power supply load per-rack

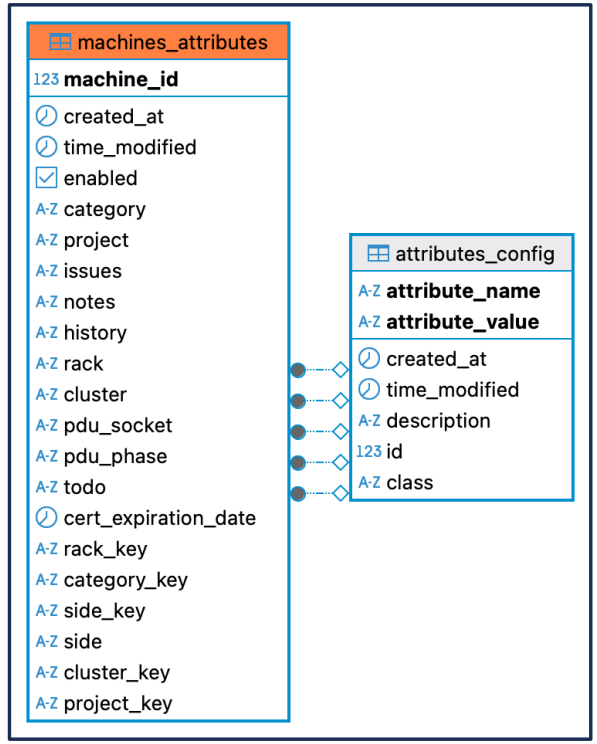
- Power supply monitoring.
- Load distribution per rack.
- Alerting based on parameters configured in the database.



Machines database

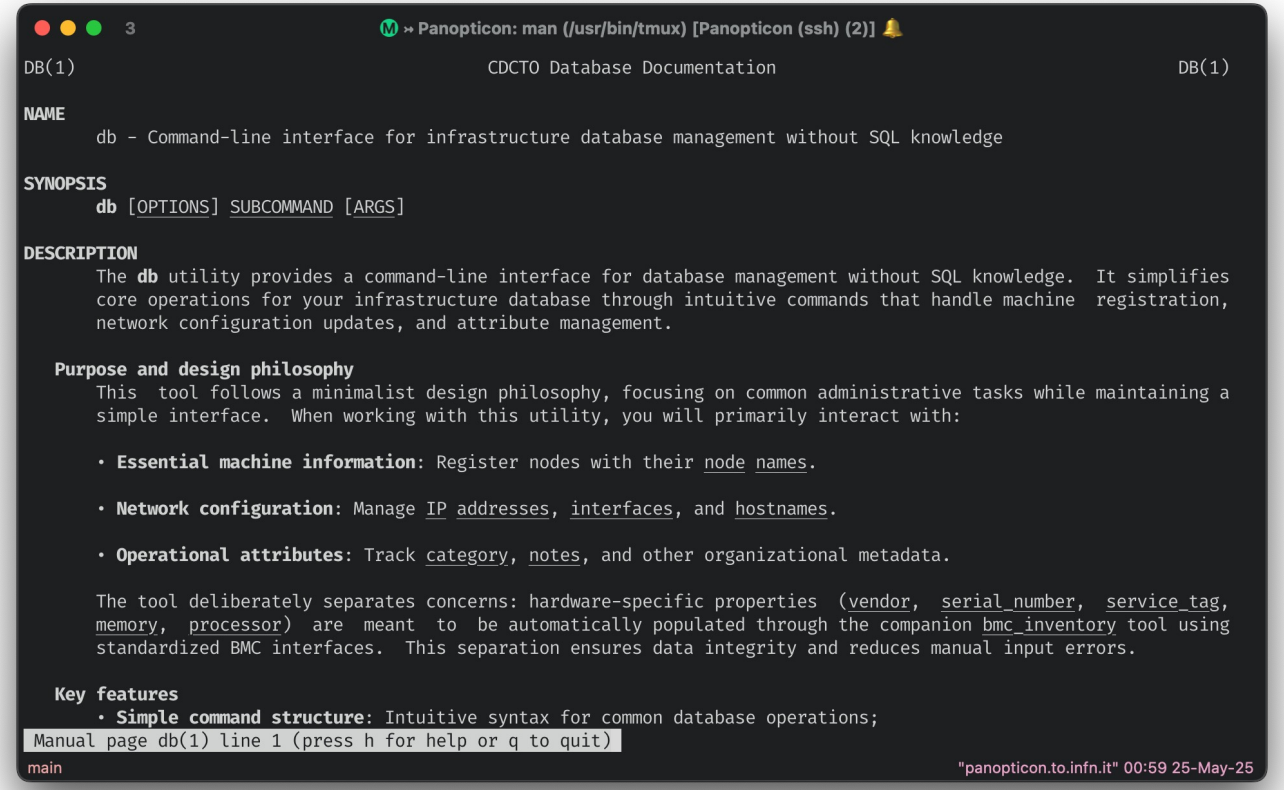






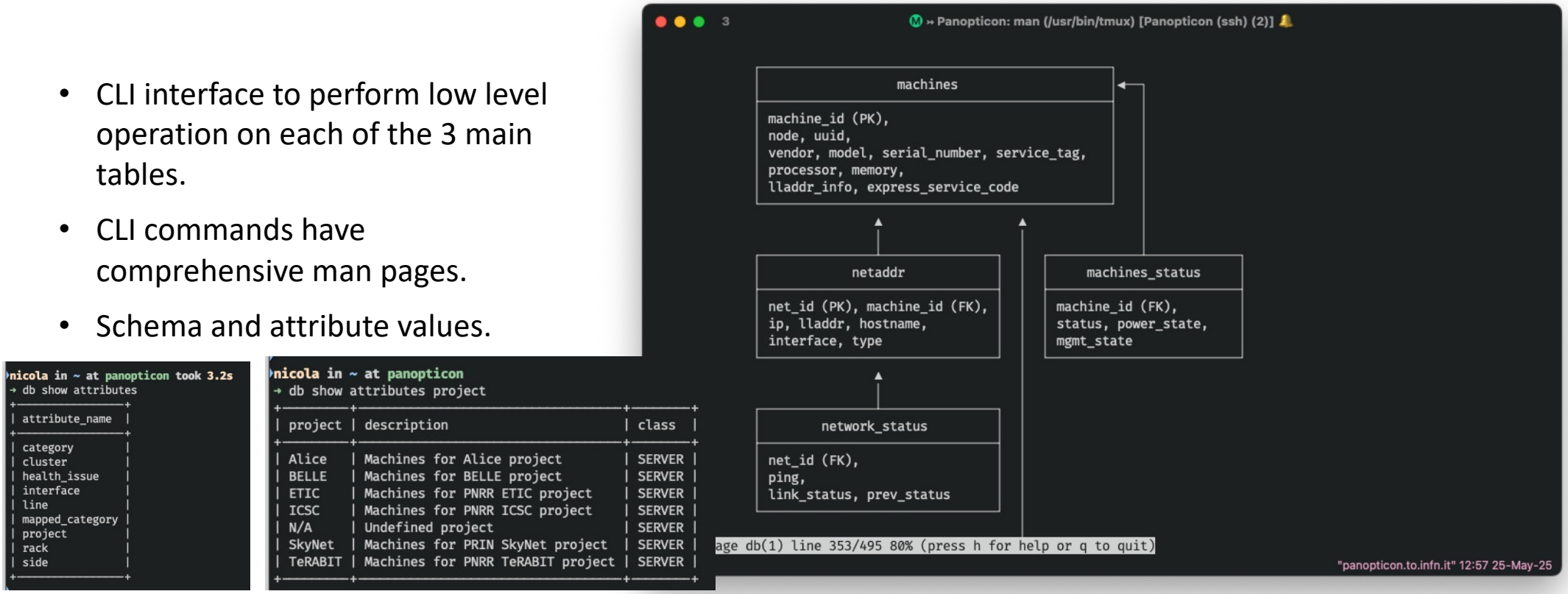
Machines database: CLI

- CLI interface to perform low level operation on each of the 3 main tables.
- CLI commands have comprehensive man pages.



Machines database: Schema

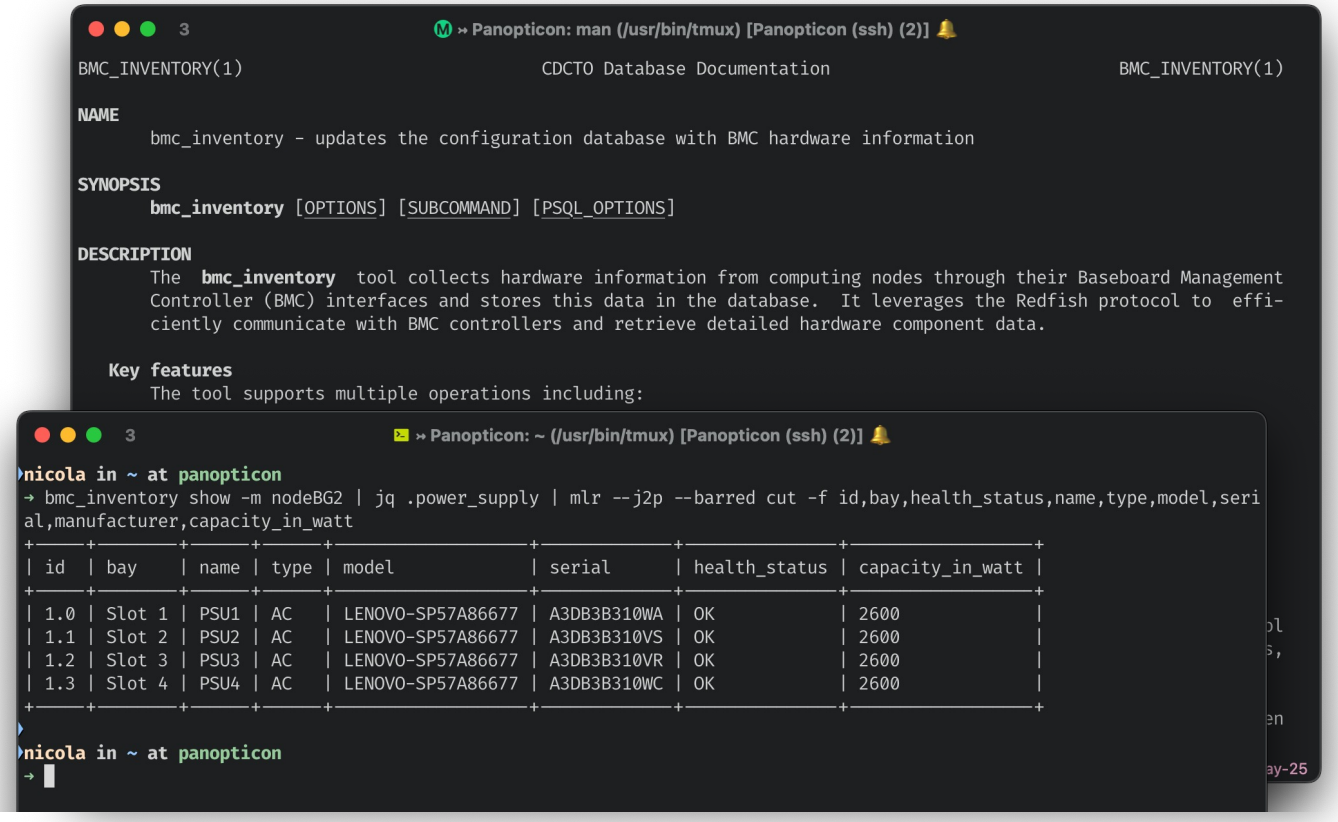
- CLI interface to perform low level operation on each of the 3 main tables.
- CLI commands have comprehensive man pages.
- Schema and attribute values.



Inventory retrieval

- CLI interface to get inventory from BMC.
- PostgreSQL views for easy access to relevant information (processors, memory, disks, etc…).
- Thanks to check_redfish.py script: plugin for Nagios, NetBox, Icinga2.

https://github.com/bb-Ricardo/check_redfish



Future developments

- Analysis of aggregated data over a long period of time.
- Long term storage of relevant metrics.

Aggregate metrics and statistics:

- Thanos
- Cortex
- Mimir
- VictoriaMetrics

Native Prometheus support

- TimescaleDB + PostgreSQL

Long term goals for the monitoring:

- Additional infrastructural components: Cooling systems and incident response.
- Higher abstraction layers: Integrate monitoring of cloud and orchestration layers (e.g. OpenNebula).
- Monitoring of the application layer (e.g. JupyterHub/JupyterLab).

Conclusions

Thank you!

- Monitoring infrastructure: PostgreSQL for machines catalog, Redfish and SNMP for metrics collection, Prometheus for metrics ingestion, Grafana for visualization.
- CLI interface for simplified database management.
- Provisioning pipeline of datasources and dashboards in Grafana.
- Future development: Storage devices are not yet in the database, long term storage of relevant metrics; enhance monitoring coverage to infrastructural components (like cooling appliances), cloud and orchestration layer (OpenNebula), and applications layer.

BACKUP SLIDES

Computing infrastructure

HTC Cluster (WLCG Tier 2)

ALICE

BELLE

Yoga cluster

jupyterhub

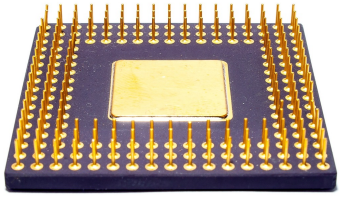
Local cluster for ML

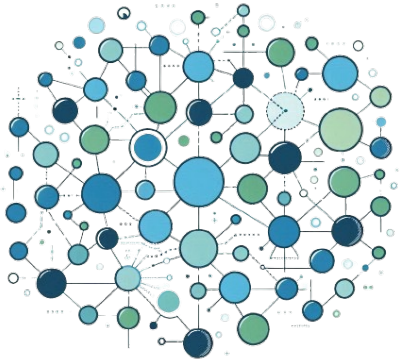
CTlab4ET

ET ITALY Einstein Telescope

GraceHopper

Monitoring a computing infrastructure





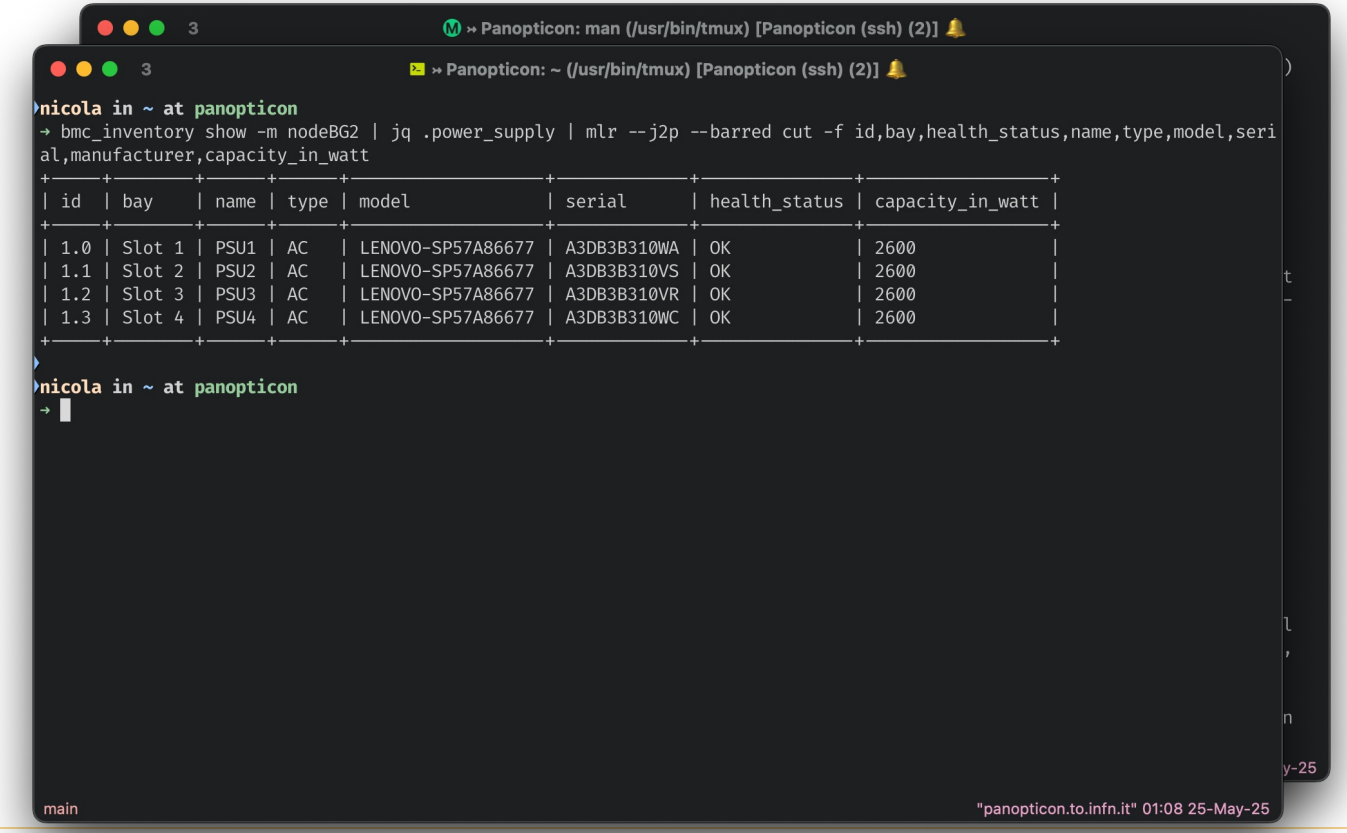
Resources:

- Computing nodes
- Network devices
- Storage
- Appliances

- Availability
- Resource optimization
- Issue detection

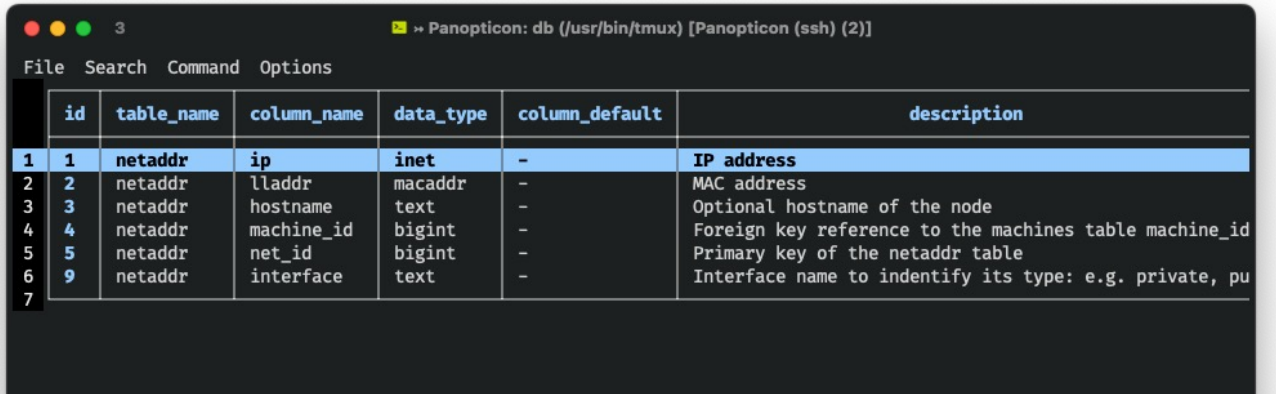
Inventory retrieval

- CLI interface to get inventory from BMC.
- PostgreSQL views for easy access to relevant information (processors, memory, disks, etc…).
- Thanks to check_redfish.py script: plugin for Nagios, NetBox, Icinga2.
- Easy pipeline on the command line with Miller and jq to process JSON data.



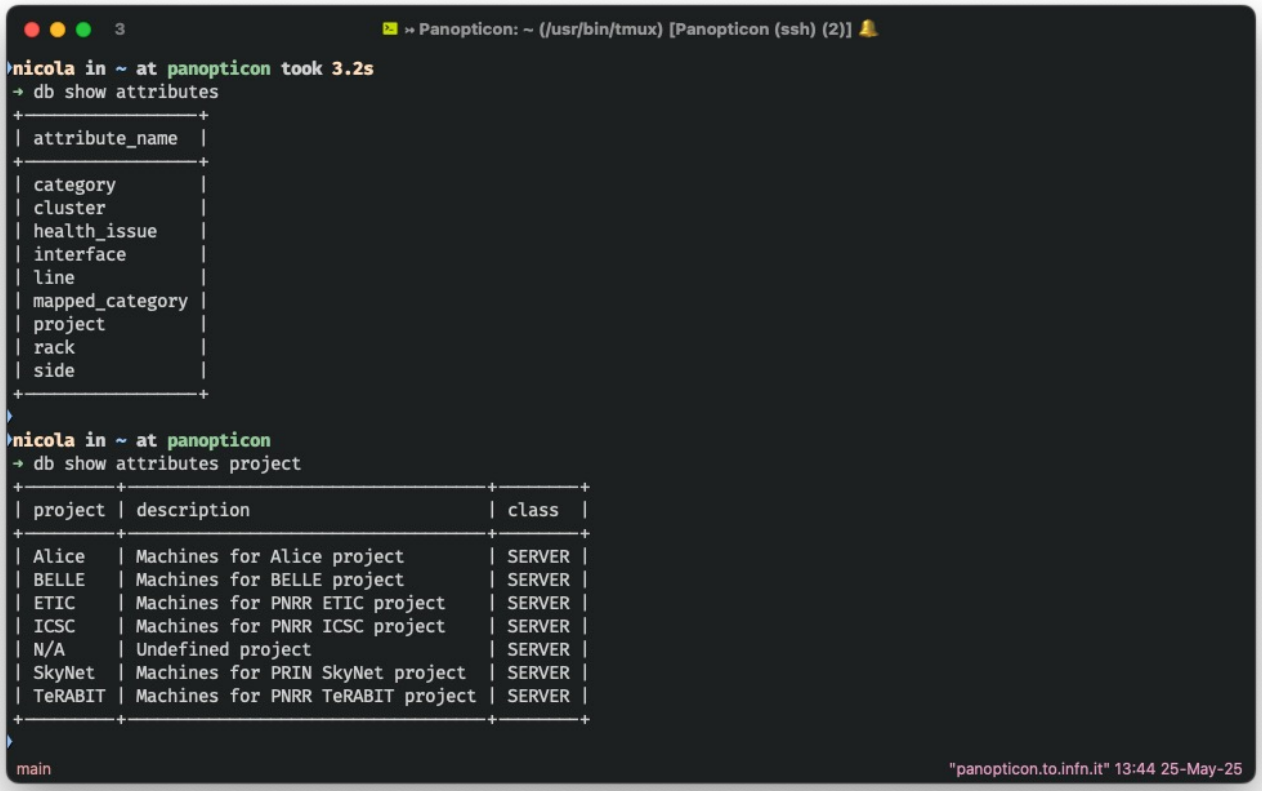
Machines database: Schema

- CLI interface to perform low level operation on each of the 3 main tables.
- CLI commands have comprehensive man pages.
- Schema and attribute values.
- pspg: a pager for PostgreSQL.



Machines database schema

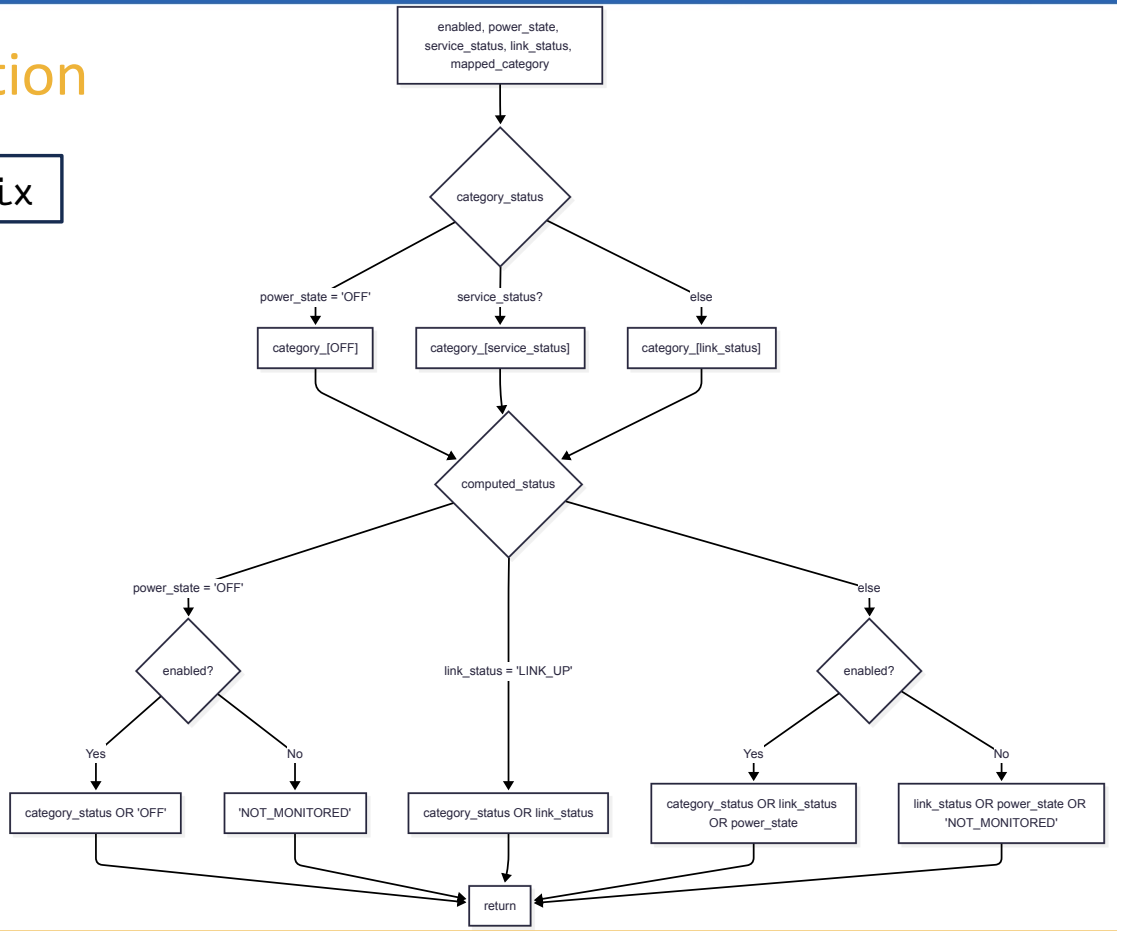
- CLI interface to perform low level operation on each of the 3 main tables.
- CLI commands have comprehensive man pages.
- Schema and attribute values.



Dashboard design: Status determination

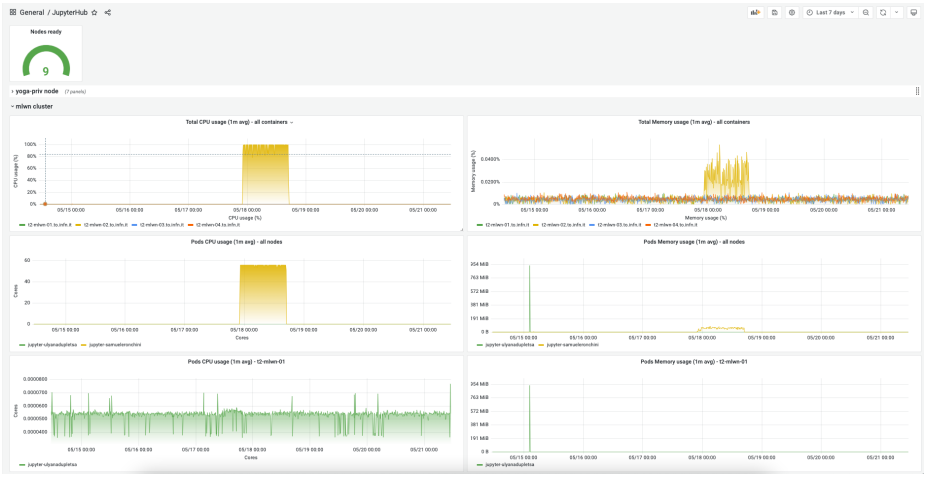
```
category_status = mapped_category + status suffix
```

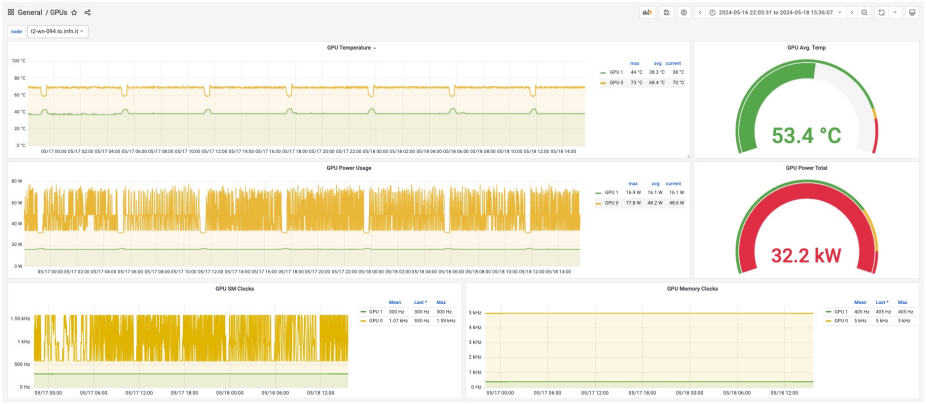
```
computed_status
```

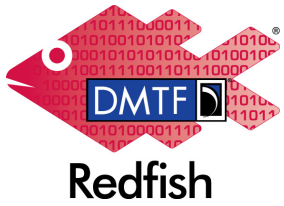


Yoga cluster

- Prototype for the resources that we intend to provide to the local community.
- It should be combined with INFN Cloud.
- Kubernetes cluster for ML applications.
- JupyterHub interface.
- CPU and GPU resources.
- Monitoring based on Prometheus and Grafana running in kubernetes nodes.







- Open standard specification
- Agent-less
- RESTful API
- JSON-based schema
- Features: management, provisioning, monitoring
- Monitoring hardware health: temperature, power, disk I/O
- Logs collection

- Industry standard protocol
- Manager-agent architecture
- Based on MIBs (Management Information Bases)
- Monitoring network devices, CPU/RAM usage, storage
- Real-time alerts for network faults and errors
- Mature and widely adopted in the industry

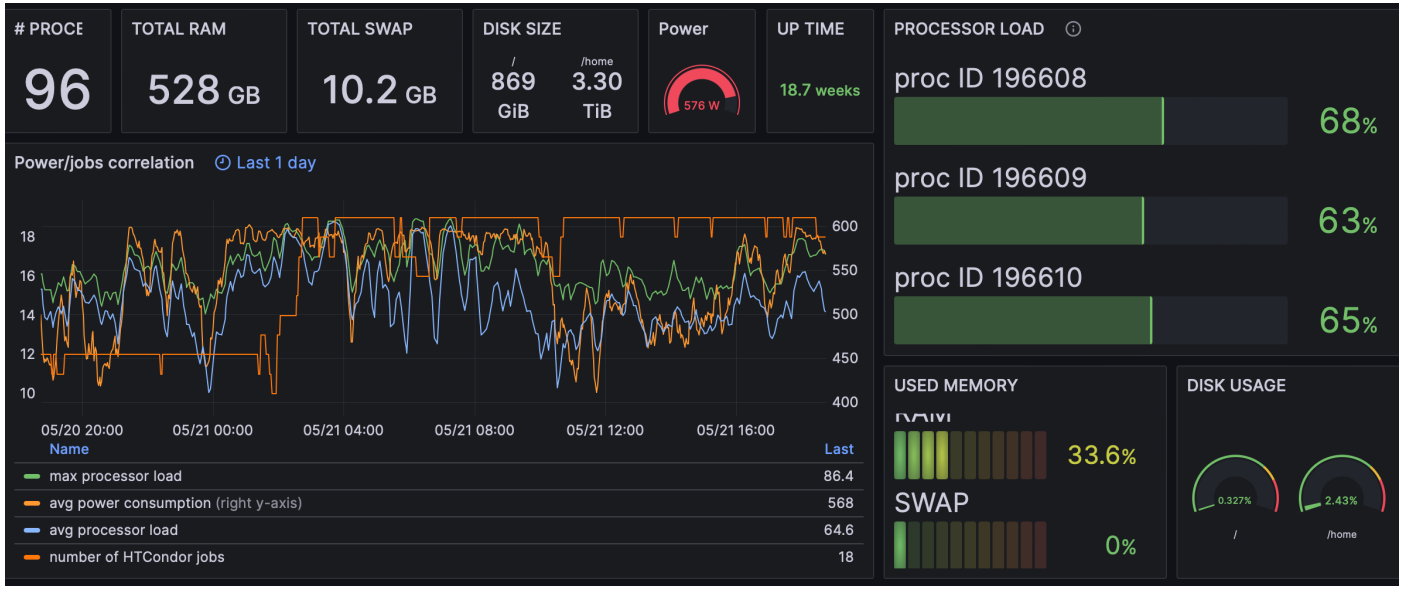
Prometheus

- Open source tool with alerting capabilities
- Time-series database
- Collection of data over HTTP (through exporters)
- Widely adopted for reliability and scalability

- Open source platform for creating interactive dashboards
- Supports several data sources
- Notification when alerts are raised in Prometheus

Use case: Monitoring HTCondor jobs, CPU load, and power consumption

- SNMP tool installed on nodes to collect hardware metrics: CPU, disk, RAM; collect metrics about HTCondor jobs.
- Custom Redfish exporter to collect metrics about about power consumption and temperature.
- Data collected by Prometheus and displayed in Grafana: show correlation between power, CPU load, and number of jobs.



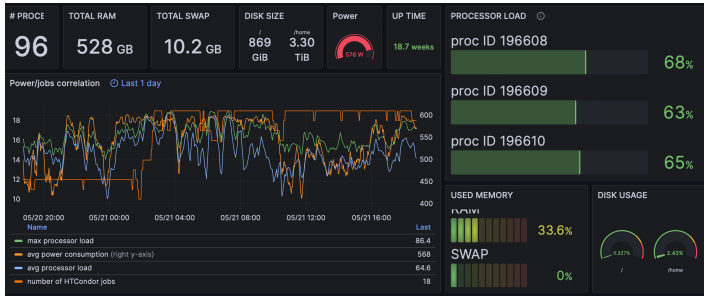
HTCondor Software Suite
jupyterhub
OpenNebula

Application
Middleware
Hardware
Base infrastructure

Redfish
SNMP
Telegraf

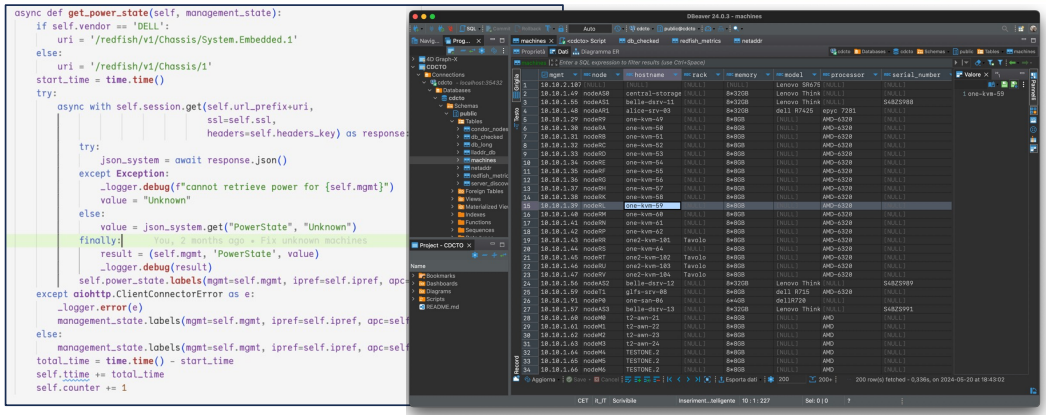
Prometheus

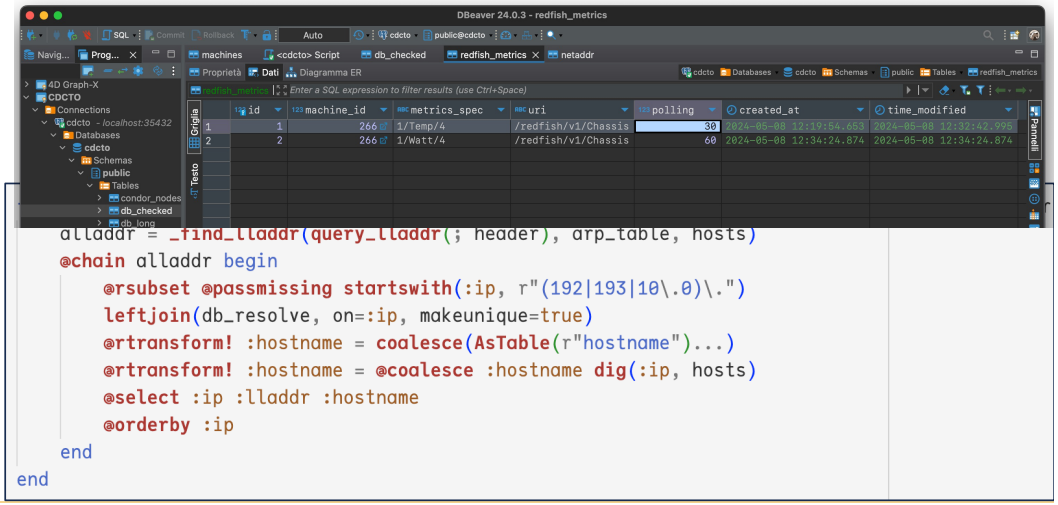
Grafana



Future goals and development:

- Under development Python script for (semi)automatic discovery of metrics paths in Redfish.
- Metrics configuration stored in (Postgres) database.
- Configuration information processed by Julia scripts and loaded into the database.
- Redfish used to retrieve logs and collected by the ELK stack.






```
    alladdr = _find_lladdr(query_lladdr(; header), arp_table, hosts)
    @chain alladdr begin
        @rsubset @passmissing startswith(:ip, r"(192|193|10\.0)\.")
        leftjoin(db_resolve, on=:ip, makeunique=true)
        @rtransform! :hostname = coalesce(AsTable(r"hostname")...)
        @rtransform! :hostname = @coalesce :hostname dig(:ip, hosts)
        @select :ip :lladdr :hostname
        @orderby :ip
    end
end
```

Computing infrastructure

CTLab *(Computing Technology Lab)*:

- New cluster dedicated to ET (*TechZoo*).
- Evaluate new technologies.
- Dedicated heterogeneous computing power on demand.

TeRABIT:

- Network for academic research.
- Upgrade current network to Tb/s.
- Infiniband connection high-bandwidth/low-latency.
- HPC bubbles: HPC resources available close to the user.

synergy

What do we have in Turin?

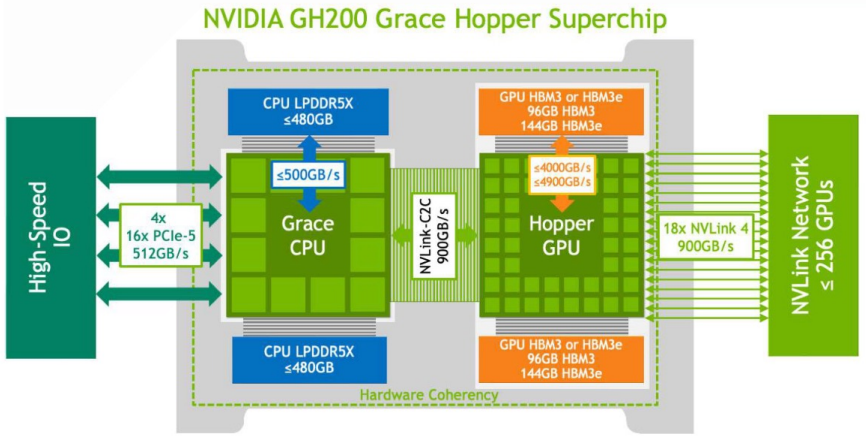
TeRABIT:

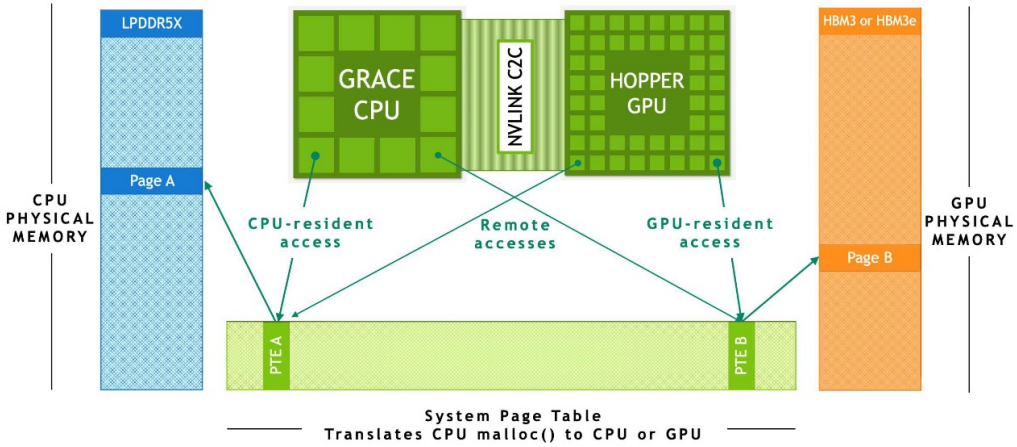
- 6xCPU nodes
  - 112-192 cores
  - RAM 8GB(DDR5) minimum/core
  - InfiniBand NDR 400Gbps
- 6xGPU nodes
  - 80GB minimum
  - HBM2e memory per node
- 1xInfiniBand switch 400 Gb/s

What do we have in Turin?

Evaluation system, funded by ETIC grant:

- 1×NVIDIA Grace Hopper Superchip
  - CPU+GPU coherent memory model
  - 900 GB/s coherent interface NVLink-C2C
  - Adopted also in Alps Supercomputer @ CSCS
  - 7x higher than x16 PCIe Gen5 bandwidth lanes



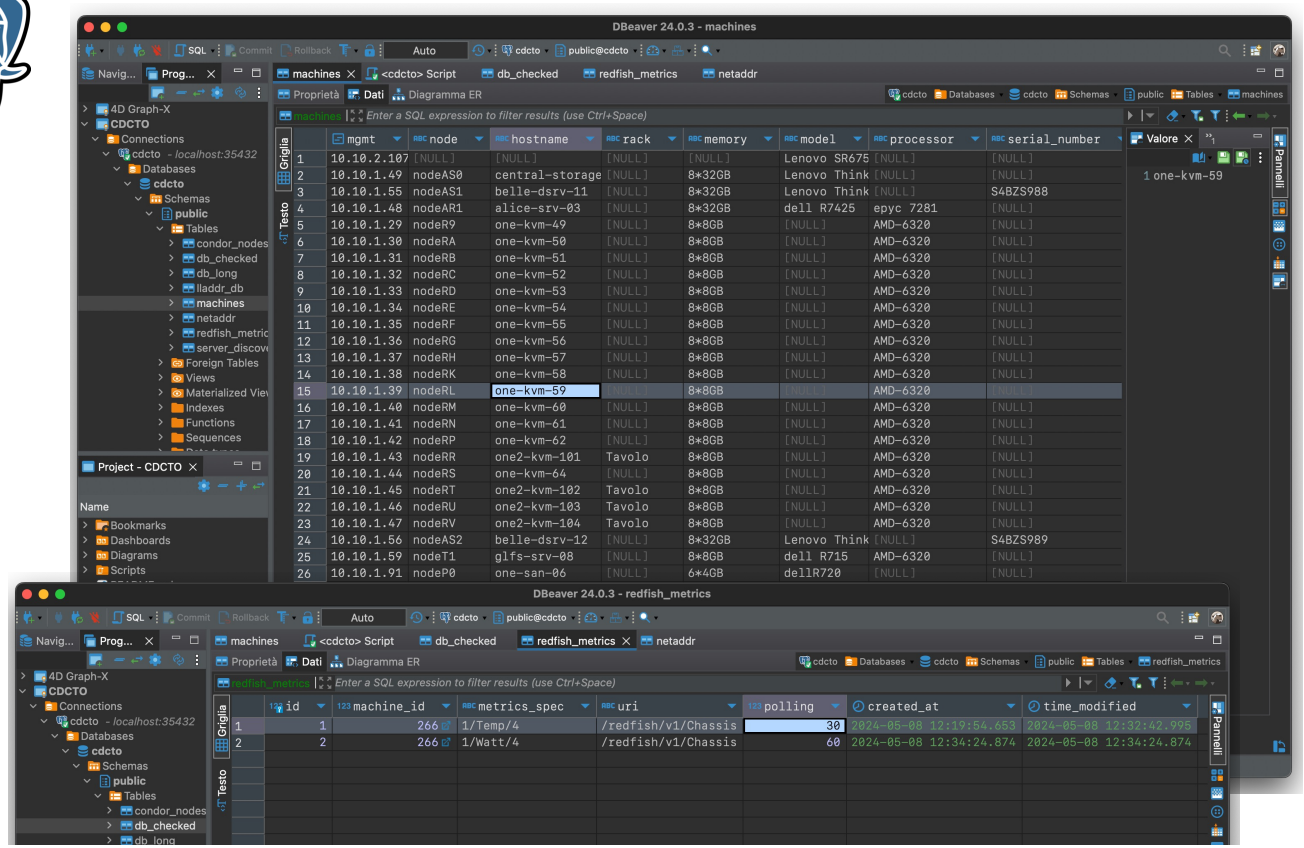


Redfish and Snmp

- **Benefits of Redfish and SNMP**
- **Scalability**: Redfish and SNMP support monitoring of large-scale infrastructures with ease.
- **Interoperability**: Both protocols are widely supported by hardware vendors, ensuring compatibility across different devices and platforms.
- **Real-time Monitoring**: Redfish and SNMP provide real-time insights into hardware and network performance, enabling proactive issue resolution.
- **Standardization**: Redfish and SNMP adhere to industry standards, simplifying integration and automation in infrastructure monitoring workflows.

Redfish configuration development

- Database initially built from preexisting sources, processing the information with Julia scripts.
- Julia script to verify the initial information.
- At this stage data is imported manually into the database.
- Prometheus configured by exporting the configuration *via* a script.



Future goals and development:

- Database of machines for configuration of Redfish exporter.
- Python script for (semi)automatic discovery of metrics paths.
- Redfish used to retrieve logs and collected by the ELK stack.

```
async def get_power_state(self, management_state):
    if self.vendor == 'DELL':
        uri = '/redfish/v1/Chassis/System.Embedded.1'
    else:
        uri = '/redfish/v1/Chassis/1'
    start_time = time.time()
    try:
        async with self.session.get(self.url_prefix+uri,
                                    ssl=self.ssl,
                                    headers=self.headers_key) as response:
            try:
                json_system = await response.json()
            except Exception:
                _logger.debug(f"cannot retrieve power for {self.mgmt}")
                value = "Unknown"
            else:
                value = json_system.get("PowerState", "Unknown")
            finally:      You, 2 months ago • Fix unknown machines
                result = (self.mgmt, 'PowerState', value)
                _logger.debug(result)
            self.power_state.labels(mgmt=self.mgmt, ipref=self.ipref, apc=self.apc).state(result[2])
    except aiohttp.ClientConnectorError as e:
        _logger.error(e)
        management_state.labels(mgmt=self.mgmt, ipref=self.ipref, apc=self.apc).state("DOWN")
    else:
        management_state.labels(mgmt=self.mgmt, ipref=self.ipref, apc=self.apc).state("UP")
    total_time = time.time() - start_time
    self.ttime += total_time
    self.counter += 1
```

```
function find_multiple_addr!(df::AbstractDataFrame)
    addr_types(n) = map(("ip_$i" for i in 1:n)) do x
        replace(x, r"^ip_1" => "primary", r"^ip_2" => "seco
    end
    @chain df begin
        @rsubset begin
            :source in ["left_only", "both"] && startswith(
            @kwarg view=true
        end
        @groupby :lladdr
        transform!(nrow => :naddr)
        @rsubset(:naddr > 1; view=true)
        @aside @debug "naddr: $(length(_.naddr))"
        @transform! :type = addr_types(length(:naddr))
        @orderby :mgmt :ip
    end
end
```

Monitoring a Computing Infrastructure

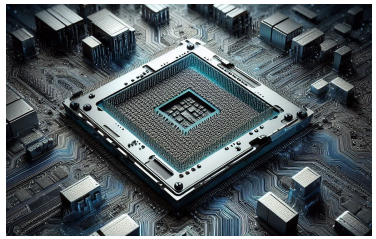
- Ensure availability
- Performance optimization
- Detect issues early
- Security

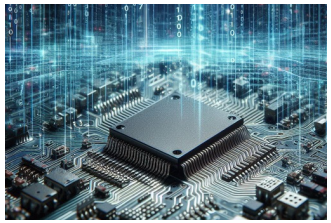
Monitoring a Computing Infrastructure

- Ensure availability
- Performance optimization
- Detect issues early
- Security

https://home.web.cern.ch/science/computing

Monitoring a Computing Infrastructure





Availability & Issue detection

Resource optimization