

# Monitoring at GWDG

---

Hendrik Nolte  
hendrik.nolte@gwdg.de

Marcus Merz  
marcus.merz@gwdg.de

Julian Kunkel  
julian.kunkel@gwdg.de

25. Mai 2023

hpc@gwdg.de

GWDG – Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen

- Continuous collection of data/metrics from a system
- Analysis of the collected data/metrics within a period of time
- Up-to realtime
- Storing of (selected) data for later analysis

Difference to reporting/data analysis:

- Monitoring takes place nearly in real-time
- Reporting is about analyzing data/metrics over a long period: statistics

- W/o monitoring the status of the system/software is unclear
- Admins want monitoring data to check:
  - Availability
  - Performance of the system
  - Status of different components (degraded modes etc.)
  - Status and compliance of the system
  - integrity of the system

Collected data can be used later offline

- Problems, e.g., regressions can be identified using the data/metrics
- Analyze system performance to plan future updates/systems (procurement)
- HPC users - profile/optimize jobs utilizing metrics

GWDG is operating and monitoring multiple systems

- SCC (local Super Computing Cluster) for UniGoe/Max Planck
- Emmy, Grete (HLRN/NHR)
- CIDBN (Max Planck)
- Caro (DLR)
- KISSKI (upcoming)
- misc. Customer Clusters (Hosting)



- Icinga/Nagios for network and service monitoring
- Different log/warning systems from the systems themselves, e.g.
  - Slurm
  - Storage controller
- Performance monitoring stacks
  - TIG (Telegraf, InfluxDB, Grafana)
  - Prometheus/Grafana
  - TICK (Telegraf, InfluxDB, Chronograph, Kapacitor)

→ We have a heterogeneous stack but we are currently integrating it

- Performance monitoring stack/TS-DB selection
  - One core part of the monitoring system: path-dependency
  - Decision between real open-source and open-core
    - e.g. Prometheus vs. Grafana
  - Avoidance of bait-and-switch scheme
  - High ingestion performance but low node performance impact
  - Community/supplier support
  - Scalability for the future

- Integration/adaptation of scripts/tools from different HPC monitoring systems
- Final selections of stacks may break some tools (e.g. ProfitHPC)
- Hot swap - monitoring system has to be switched fast for one cluster

- Login nodes (bcm, ssh, clock)
- Compute nodes (bcm, ssh, clock)
- Switches/router
- Slurm components (ctld, database, slurmd)
- Storage/filesystem status, incl. infiniband/omni-path switches
- Basically every node, every running service, every piece of infrastructure
- If something stops working, we want an immediate alert

- PDUs, e.g. Status,
- PSUs, e.g. rack, node (groups) specific power consumption
- CDUs, e.g. temperature (inflow, outflow), cooling demand, Heat-Capture Rate
- Inrow/Side cooler status, e.g the free-cooling or compression
- nodes temperature, fan speeds, etc.
- We try to gather as much information as possible

# Example: PSU Dashboard

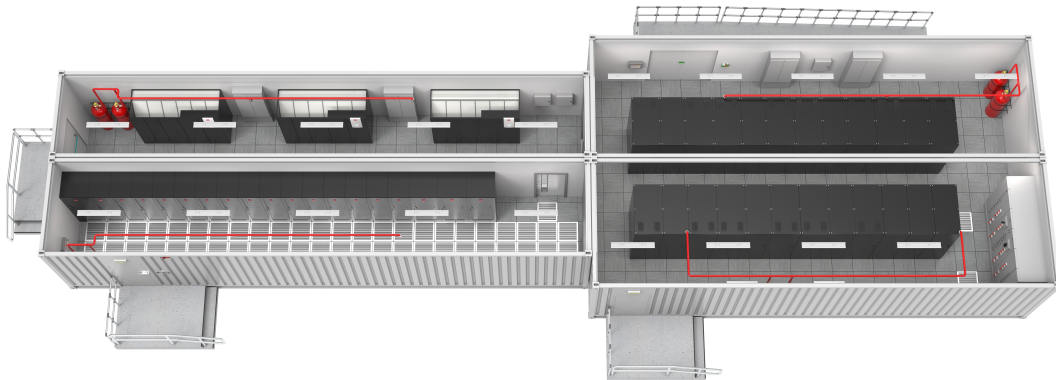




- Nodes performance data, e.g:
  - GPU/CPU(s) load
  - Memory bandwidth/consumption
  - IO loads/bandwidth consumption
- Job specific, e.g.
  - GPU/CPU loads,
  - Bandwidth used
  - I/O Load,
  - Storage/memory usage

# Example: GPU Specific Dashboard





- Why is gathering all these data at all useful?
- For example you can start to model your data center and do optimizations

## Defining Models

### Model HCR

lm(formula = HCR.Secondary ~ Real.S2n \* Real.Airn + I(Real.S2n<sup>2</sup>))

### Model Rack Power Consumption

lm(formula = Power.20 ~ Real.S2n + Real.Airn)

### Model Fanspeed

lm(formula = Fan.20 ~ Real.Airn + I(Real.Airn<sup>2</sup>))

→ Use these models to make predictions for new weather conditions

Former NHR project for job performance analysis.

Goals:

- Provide a tool to summarize job performance data and give hints to the customer.
- Easy to read performance reports (text, pdf)
- Complement existing performance tools

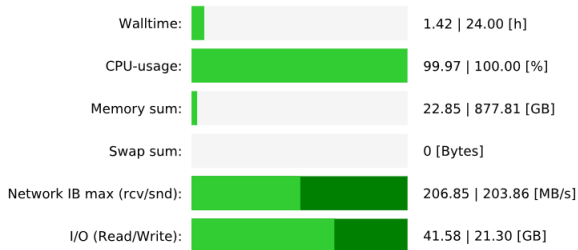
Motivation:

- Users require an easy way do a performance analysis of jobs
- Available, powerfull performance tools can be difficult to user or understand
  - Likwid, Scalexa, Vampyr
- Profit-HPC Webpage: <https://profit-hpc.de/>

- Batch job summary,
- CPU (mean values),
- Main memory (mean and maximum values),
- Swap space (mean and maximum values),
- IO (work, NFS, scratch; mean values),
- Network (infiniband, ethernet: mean values),
- GPU (if exists; mean values).



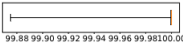





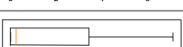
## Global Summary of Resource Usage



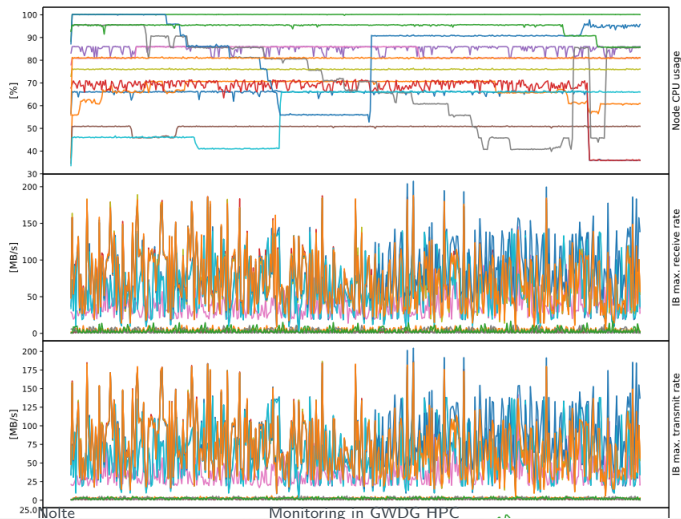
### Recommendations:

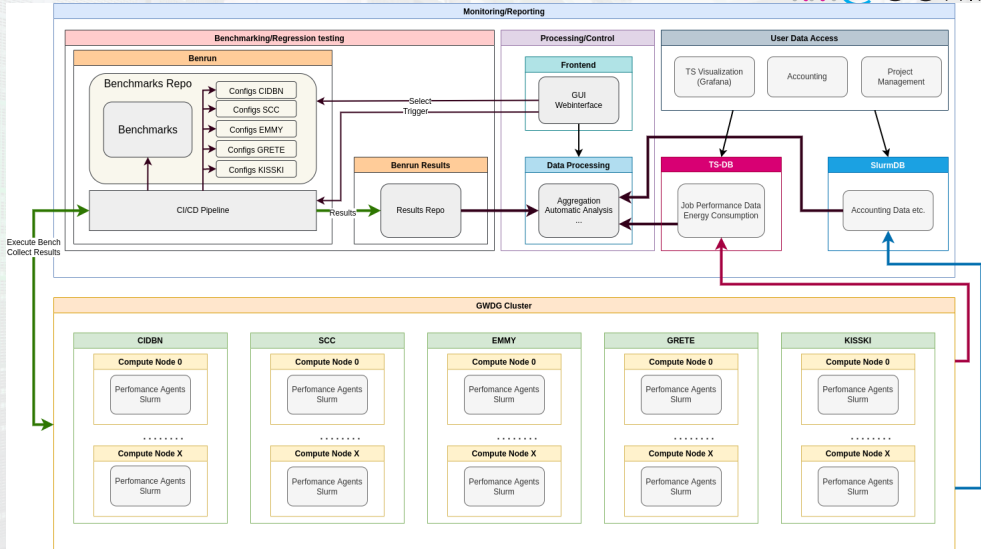
No problems detected!  
Good work!

## Node Distributions

Metric	Units	Min.	Max.	StdDev.	Avg.	Boxplot*
CPU usage	%	99.88	100.00	0.03	99.99	
Memory MaxRSS	GB	0.42	5.31	1.58	1.76	
Memory AveRSS	GB	0.41	5.02	1.47	1.67	
Memory Swap	Bytes	0.00	0.00	0.00	0.00	
I/O Read Size	GB	0.00	13.86	4.74	3.20	
I/O Write Size	GB	0.00	7.12	2.41	1.64	
I/O Read Count	*1.E-03	2.83	34.19	10.46	10.09	

## Node Timeseries Plots





# benrun (WIP)

## Goals:

- Central tool to prepare/run benchmarks and system probing
- Provide unified structure for benchmark/probe software
- Provide easy way to access results for further processing
- Use results for regression testing etc.


## Motivation:

- Enable engineers to just benchmarks/probes regression tests
  - Preparing/compiling/running benchmarks is usually done manually
  - Error prone as different settings may lead to non-comparable results
- Avoid duplication of work effort for porting/setup
  - Benchmark setup is stored for every system and is available
- Avoid scattering of results in different locations
- Offer an extendable toolbox for benchmarking/probing even for non-specialists
- Have a central point to sanity check your own metrics again


- Benchmarks/probes can be run:
  - Manually from command line
  - Via normal cron jobs
  - Via external trigger (requires a local script running)
  - Via the webpage and gitlab-runner
- Simple selection scheme: system → benchmark → config → nodes
- Simple security scheme
  - No trust in any external repository (could be hacked)
  - Repo with scripts/configs is only cloned manually
  - Parameters given are not executed/sourced into scripts
    - Avoid script injection
    - Required to allow script to be triggered from outside, e.g via a CI/CD runner
    - Only execute benchmark if all parameters are valid



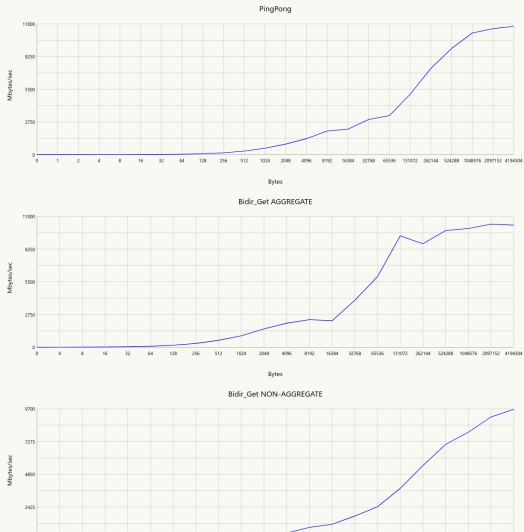
# benrun cont'd - simple results display

  
benrun

- Reports
- Submit Benchmark
- Schedule Benchmark
- Delete scheduled Benchmark
- List scheduled Benchmarks
- Settings

  
GWDG

Report STDOUT STDERR



- Reports
- Submit Benchmark
- Schedule Benchmark
- Delete scheduled Benchmark
- List scheduled Benchmarks
- Settings

## Schedule Benchmark

System  
SCC

Benchmark  
imb

Hourly Daily Weekly Monthly Yearly

Minute

\*

Hour

\*

Day of Month

\*

Month

\*

Day of Week

\*

Generate Cron String

\*\*\*\*\*

Description

The Description of the pipeline schedule.

Submit

- Automatic regression testing/analysis
  - Support optimization of software in regards to energy-consumption/performance
  - Optimize hardware settings for specific application to improve performance/power-consumption
  - Preventive maintenance by analyzing data for upcoming faults
  - Automatic supplier tickets for specific failures
  - Port ProfiT HPC to new infrastructure
- ← requires monitoring unification/central access of data

- In general there exists two different security-related monitoring purposes:
  - Compliance Checking / System Hardening
    - Monitoring that the system is in a well defined state
    - Enforcing that the System is deployed as described in a operational concept
    - Mainly focused on under-caffeinated admins
    - Should prevent a mistake/vulnerability from happening
  - Incident Recording / Intrusion Detection
    - Used to detect any suspicious activities on the system
    - Aiming to detect a potential attack
- Both will generate logs, instead of time series as in performance monitoring
- Both have to be considered as part of an overarching security concept

- Security Onion with 4 Layers
- Most sensitive systems at the top
- *ssh* access to admin nodes only
  - from an isolated admin-network
  - using sk keys on the JumpHost
- Only Movement downwards permitted
  - Enforced via node-local firewalls

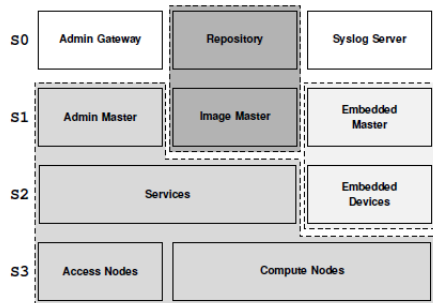


Abbildung LB.3: Sicherheitsschichten in der Systemkonfiguration des HLRN-IV.

- Layer 0
  - Most sensitive Systems
  - JumpHost, Syslog Server, etc.
  - Administration via special clients
- Layer 2
  - Management Nodes with daemons
  - e.g. slurmctld, licenses, filesystems
- Layer 1
  - Everything between Layer 0 and 2
  - e.g. Admin Master
- Layer 3
  - user nodes

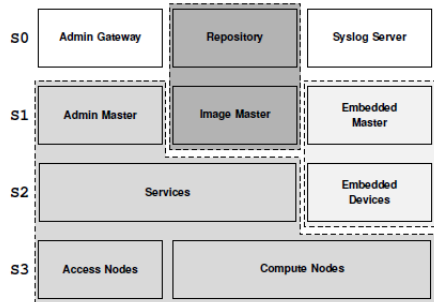


Abbildung LB.3: Sicherheitsschichten in der Systemkonfiguration des HLRN-IV.



# Compliance Checking

- Compliance checking on all nodes
  - Continuously during runtime
- Methodology
  - Create security concept for each node
  - Derive legitimate system state
  - Determine critical components
  - Check them via Icinga/Nagios
    - Simple checksum
- Components
  - iptables, ssh files, kernel modules
- Purpose
  - Not for intrusion detection
  - Proactively prevent vulnerability due to under-coffinated admin

**Tactical Overview: iptables**

**Host Summary**  
0 Hosts Down

**Service Summary**  
0 Services Critical

**Hosts: iptables**  
No hosts found matching the filter.

**Services: iptables**

OK	14 9h	check iptables rules on glogin7	[OK]: iptables rules are unmodified
OK	16 15h	check iptables rules on glogin8	[OK]: iptables rules are unmodified
OK	2d 2h	check iptables rules on glogin5	[OK]: iptables rules are unmodified
OK	May 4	check iptables rules on glogin6	[OK]: iptables rules are unmodified
OK	May 1	check iptables rules on glogin3	[OK]: iptables rules are unmodified
OK	Apr 27	check iptables rules on glogin9	[OK]: iptables rules are unmodified
OK	Apr 12	check iptables rules on glogin1	[OK]: iptables rules are unmodified
OK	Apr 12	check iptables rules on glogin2	[OK]: iptables rules are unmodified
OK	Apr 12	check iptables rules on glogin4	[OK]: iptables rules are unmodified
OK	2021-03	check iptables rules on gadmin1	diff: /var/spool/icinga2/data/check_iptables_files/gadmin1: no such file or directory

**UP** since Apr 28  
**OK** for 1d 9h  
Service: check iptables rules (sec:iptables)  
[Check now](#) [Comment](#) [Notification](#) [Downtime](#)

**Plugin Output**  
[OK] iptables rules are unmodified

**Problem handling**  
[Add comment](#)  
[Schedule downtime](#)  
Servicegroups: security checks

**Notifications**  
[Send notification](#)

**Check execution**  
Command: check-ip-table [Process check result](#)  
Check Source: gmon.novalocal  
Reachable: yes  
Last check: 4m 31s ago [Check now](#)  
Next check: in 5m 29s [Reschedule](#)  
Check attempts: 1/5 (hard state)  
Check execution time: 0.436s  
Check latency: 0.006291s

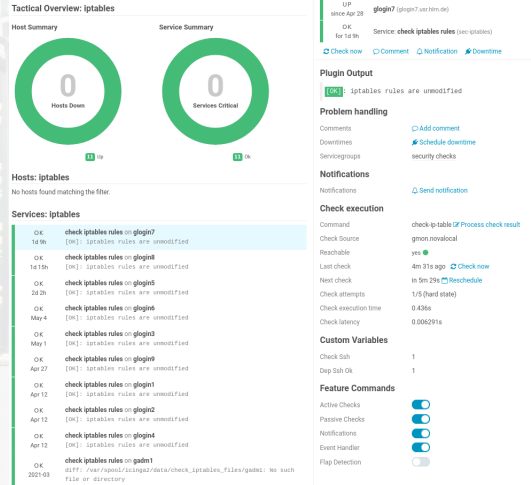
**Custom Variables**  
Check Ssh: 1  
Dep Ssh Ok: 1

**Feature Commands**  
Active Checks:   
Passive Checks:   
Notifications:   
Event Handler:   
Flap Detection:

- Nodes in S0 - S2 boot from local storage
  - Images are backed-up for disaster recovery
- Each node has individual, security-related settings, including
  - Allowed people to log in (ACL's)
  - (S-Bit) binaries
  - installed (whitelisted) software, particularly Daemons
  - configuration files
  - Systemd Unit Files
  - Kernel Modules
  - Vulnerabilities
  - ...and many more!
- These individual measures are usually described in an operational concept...
- ...and are made actionable by corresponding compliance checks

# Considerations of Compliance Testing

- Compliance Monitoring via Icinga/Nagios:
  - Can be implemented as a pull
  - From one central node e.g. a ssh connection is established
  - Only for outgoing connections to S3 nodes
- For S0-S2 nodes one should consider a client/server model:
  - Service is running on each node and pushes information to central server
    - e.g. via http
  - Data inflow needs to be supervised
  - No listening daemons,
    - only egress connections



- All S3-Nodes are deployed via a cluster manager and PXE boot
- The golden images are maintained on dedicated image servers
- These golden images are then synced to the cluster managers for deployment
- Already to this step a compliance check can be added to ensure that only compliant images to going to be deployed
- Still, compliance check at ensure continuous compliant operation

- On S3-Nodes: what security critical things can users (from userspace) do
  - This is something you have to ask yourself and answer for your individual setup
  - In our case we think that s-bit binaries are potentially dangerous
- Make those things observable (Basically the whole purpose of monitoring):
  - In our case we are using the auditd

## `/etc/audit/rules.d`

```
-a exit,always -F arch=b64 -F euid=0 -F uid!=0 -S execve -k audit-hpc  
-a exit,always -F arch=b32 -F euid=0 -F uid!=0 -S execve -k audit-hpc
```

- Logs every s-bit call on all S3-Nodes
- and sends summary every day per mail and gets checked by our security team

- IDS are specifically aimed to detect a potential break-in
- Often they also do compliance checking
- There are many tools available, FOSS and closed-source
- One should be aware of the functionality for an efficient use



- Base line scenario: An attacker successfully got a privilege escalation
- Compliance Checking / File Checks
  - Attacker changed some file, e.g. iptables/nftables, ssh,
  - Attacker switched out some binaries, e.g. ls
  - Compliance Checks often rely on a simple checksum to control legitimacy
    - If an attacker falls for this, great!
    - But after privilege escalation those mechanisms cannot be trusted anymore
  - Instead: Detect file changes by explicitly checking the inodes and check fingerprints
    - This relies on a database for matching,
    - DB needs protection from the attacker, e.g. NFS ro export
    - Requires an update of the DB whenever changes on the watched files are made
  - Probably only viable for S0-S2 Nodes



# Rootkit checking IDS

- Base line scenario:
  - An attacker successfully got a privilege escalation
  - To preserve further access a rootkit was hidden
  - This can be, for instance, a malicious LKM which provides a root shell

## Typical Rootkit Hiding

```
kill -50 1
```

- Usually an attacker will hide these things
- There are tools to (try) to detect those things, like *rkhunter* or *chkrootkit*
  - These are open source bash scripts
  - They use, e.g. `unhide` and `unhide-tcp` to detect
    - Hidden ports
    - Hidden processes
    - Processes running on deleted files
    - much more
- Main problem stays: You always have to second guess your data

- All these services before create log files
- Successful incident prevention/detection relies solely on the
  - integrity of the logged events
  - reliable analysis of these of these files
- The recommended way for log aggregation is an isolated central logging system
  - a central syslog server
  - Graylog, or similar
- From here further analysis is done and alerts are send out
- Currently we
  - check for events and send alerts
  - aggregate logs and send out summaries per E-Mail
    - Requires organization in a team that it is ensured that reports are checked
  - archive logs for post mortems

### What happened in 2022

- ISO270001 certification
- Established a secure HPC partition for GDPR protected patient data

### Activities in 2023

- Introduce 2FA for SSH login
  - Central logging of all authentication attempts for incident handling
- Deploy an IDS on system
- Update our security policy
- Combine all those sources into our monitoring infrastructure for an holistic analysis
- Utilize ML on this source to analyze prospect intrusion ...