university of groningen / center for information technology

iRODS Monitoring for System Administration

iRODS monitoring from a system-admin's perspective

iRODS Monitoring for System Administration

iRODS monitoring from a system-admin's perspective

Presentation by:

Ger Strikwerda
Luca Le Preux
John Mc Farland

Research Data Management System
University of Groningen
rdms-support@rug.nl

university of groningen / center for information technology

Small intro

- **University of Groningen:**

- Old, very old: Founded in 1614
- 32,420-ish students, 9000+ internationals/127 countries
- Research in Energy, Healthy Ageing, Sustainable Society, Astronomy
- Home of Nobel Prize winners: Frits Zernike (1953), Ben Feringa (2016)

university of groningen / center for information technology

- **CIT:**

- Old, very old: Rekencentrum started in 1953
- First HPC-computer: Zebra (zeer eenvoudig binair reken apparaat)
- Central IT for the University (CIT 270+ employees)
- Domain Infra, Domain Research, Domain Education
- Research: HPC, datamanagement

university of groningen / center for information technology

- **RDMS:**

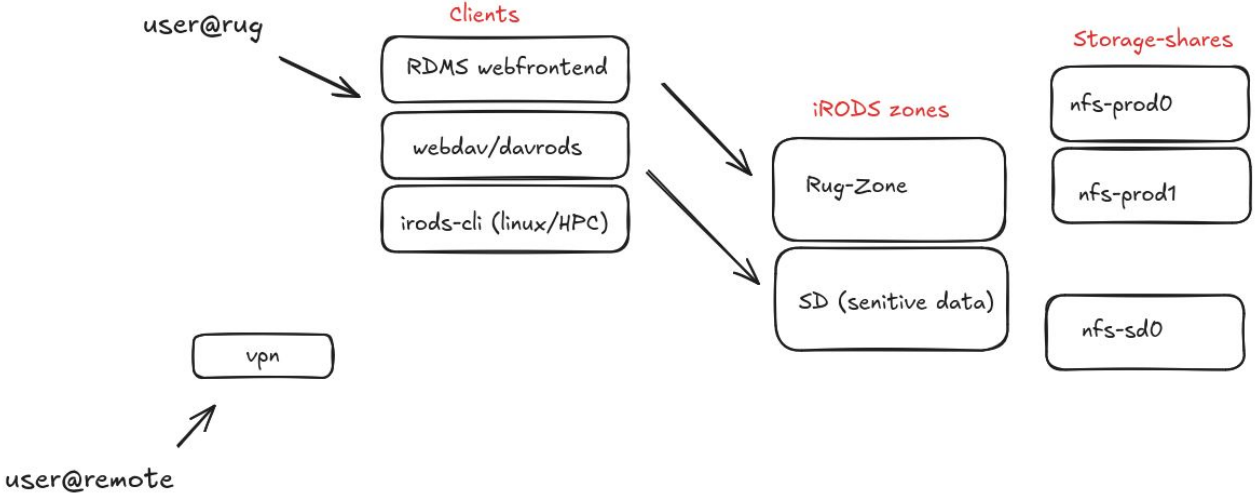
- Old, very old: iRODS community member since 2016
- We facilitate long term data storage (up to 10 years) for our researchers
- RDMS is a service: devops team 8 members
- data is replicated over 2 different sites across the town of Groningen (Zernike, Eemspoort)
- data-archive via Surf (2026, move old data disk -> tape)

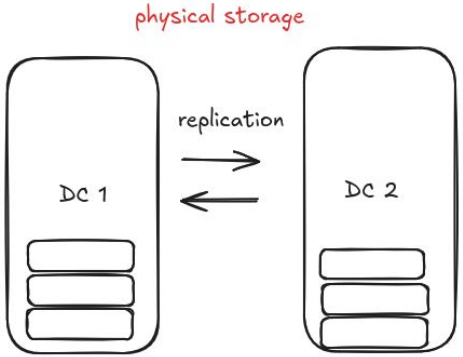
```
irods@store(00):~$ iadmin lu | wc -l
1388 users

irods@store(00):~$ df -h  | grep irods
129.125.xxx.yyy:/rdms-prod0  793T
129.125.xxx.yyy:/rdms-prod1  718T
```

university of groningen / center for information technology

iRODS setup@RUG





university of groningen / center for information technology

Ontopic: System monitoring

We have 2 options:

- monitoring via Nagios (system/and a bit application-monitoring
- irods-prometheus-exporter/grafana (combination system/application)

Nagios monitoring

'old-school':

- basically crontab-fired check-scripts (so not really realtime)
- good for sending alerts/warnings via email
- `check_irods` every 10 minutes:
  - iRODS version (imiscsvrinfo)
  - iRODS service (systemctl)
  - iRODS heartbeat (MsgHeader_PI)
  - stale delay server rules (iqstat)
  - possibly add other checks (e.g., ips)

example:

```
root@store-dev:/home/ger# /nagios/plugins/check_irods
IRODS OK: VERSION: rods4.3.4, ACTIVE: iRODS daemon active,
HEARTBEAT: HEARTBEAT, DELAY QUEUE: 0
```

university of groningen / center for information technology

Nagios versus Prometheus (or why we have both)

Nagios:

- good for warnings/triggers via email (mostly about system-related issues)
- no trends
- no graphs

Prometheus:

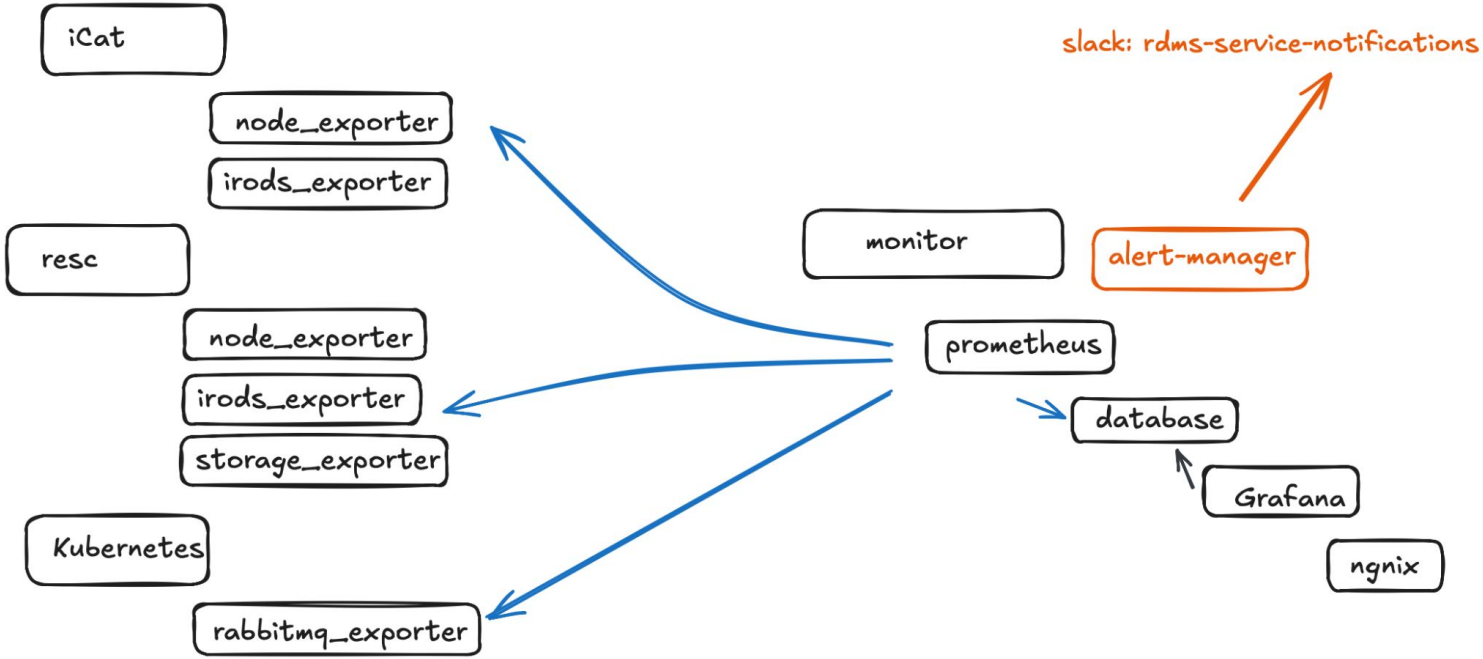
- cool dashboards/fancy graphs (reporting)
- longer time trends
- good way to correlate (make visible) the relation between application and system

Small crash-course/intro prometheus-grafana:

concepts:

- exporter (runs on a system) exports metrics via http(s)
- prometheus (runs on a monitor) scrapes the metrics and saves it in a database
- alert-manager (runs on a monitor) can set/create triggers to alert
- grafana (read the metrics in the database)
- grafana shows the metrics via web dashboard

1 picture says it all



iRODS Prometheus Exporter

- Lightweight Prometheus-compliant agent
- Python package `irods-prometheus-exporter`
- Runs as the iRODS service account
- Configurable metrics monitoring
- Configurable agent
- Deployment agnostic

university of groningen / center for information technology

iRODS application metrics

Basic server information

- `imiscsvrinfo` (iRODS version, uptime, zone name)

iRODS client processes

- `ips` (processid, user, zone, process, origin)

Delayed rules queue

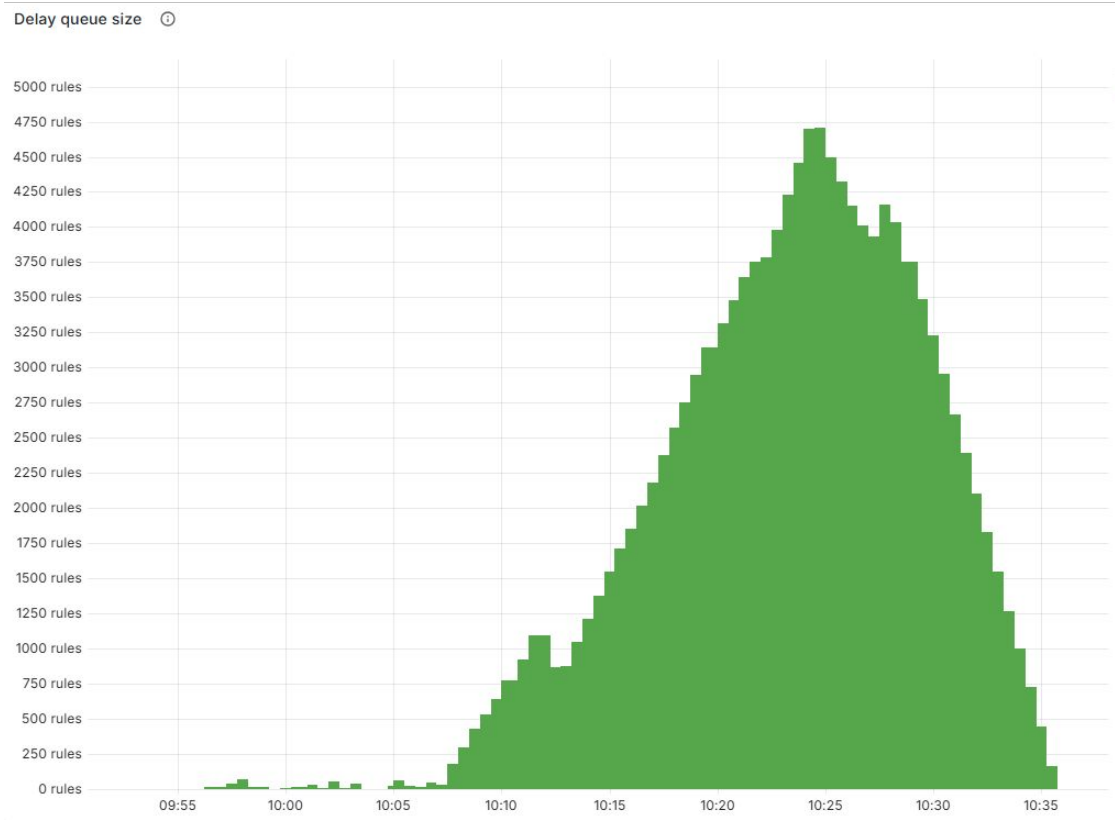
- `iquest "[\"%s\"]" "SELECT COUNT(RULE_EXEC_ID)"`
- `iquest "[\"%s\", \"%s\"]" "SELECT COUNT(RULE_EXEC_ID), RULE_EXEC_PRIORITY"`

Benchmarking (timing)

- `iquest "SELECT count(ZONE_NAME)"`
- `ils -d /rugZone/home/irods`

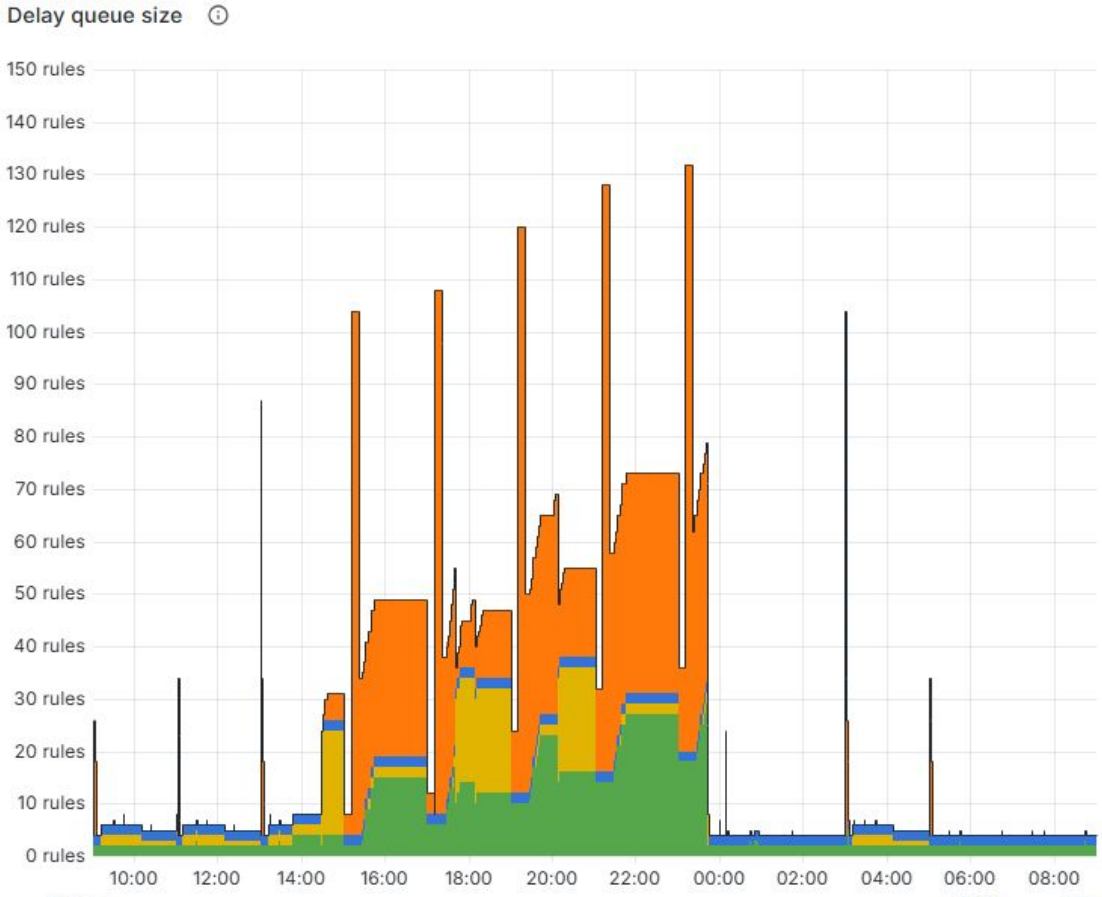
university of groningen / center for information technology

Automatic checksumming of ingested data



university of groningen / center for information technology

Delayed rules queue profile

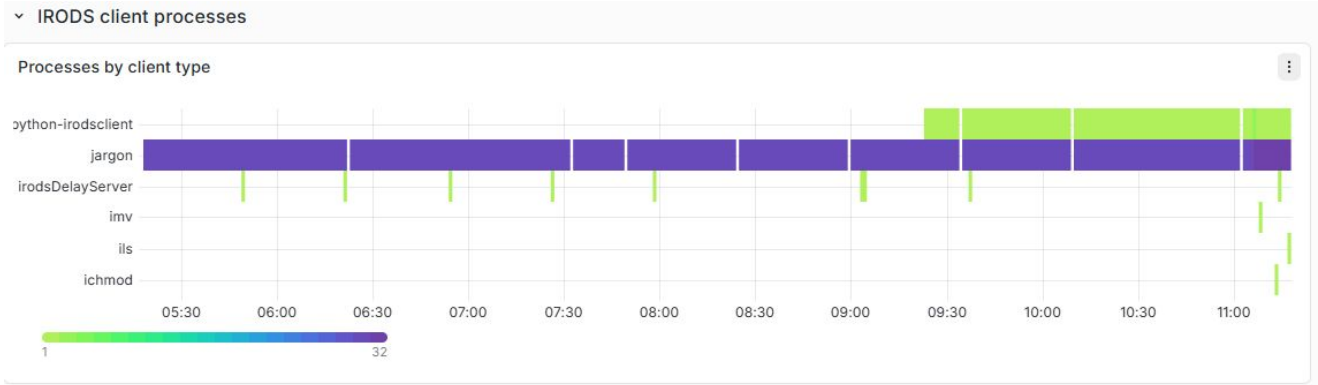


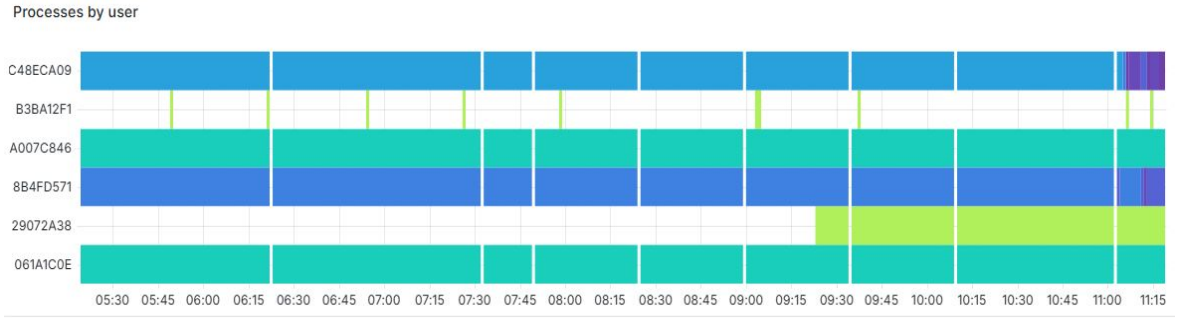
Name	Max	Min
Pr-1	29 rules	2 rules
Pr-5	20 rules	1 rules
Pr-7	2 rules	2 rules
Pr-9	112 rules	1 rules
Total	132 rules	4 rules

university of groningen / center for information technology

iRODS client processes

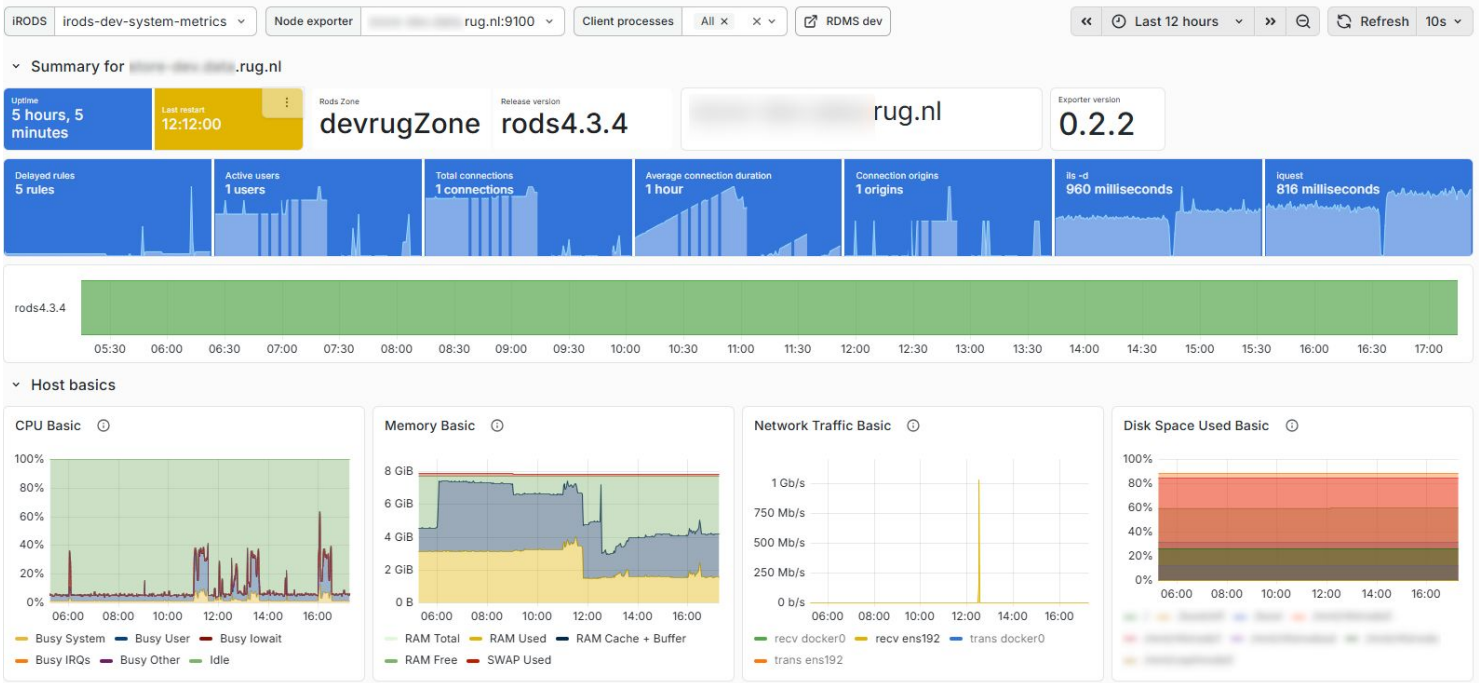
Based on `ips` iCommand





university of groningen / center for information technology

Integrating iRODS metrics with system metrics



Grafana dashboard

university of groningen / center for information technology

Future improvements

More metrics:

- Repeating rules in delayed queue
- Stale delayed rules metric (based on expected execution time)
- Simple `iput/iget` benchmarking
- Process origin resolution for known origins

Alerting:

- More alerts on iRODS metrics

Development:

- Use the python-irods-client in place of iCommands where applicable
- Open source and publish the irods-prometheus-exporter

Thank you.
Questions?

by Ger Strikwerda, Luca Le Preux, John Mc Farland
Research Data Management System - University of Groningen

university of groningen / center for information technology