HPS
marcus.merz@gwdg.de

Marcus Merz

Monitoring in High Performance Computing

Installing an example monitoring stack (TIG)

Table of contents

Learning Objectives

- Basic understanding of monitoring
- Basic installation of a monitoring stack

Definition

- 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:

- Monitoring takes place nearly in realtime
- Reporting is about analysing data/metrics over a long period: statistics

Motivation

- W/o monitoring the status of the system/software is unclear
- Admins want monitoring data to check:
  - Availability
  - Performance of the system
- 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

What can be monitored?

Availability of network infrastructure/components/services:

- PDUs, PSU, CDUs, . . .
- Switches, router, servers, compute nodes, . . .
- SSH, FTP, Apache web server, . . .

Performance/Load/Misc.:

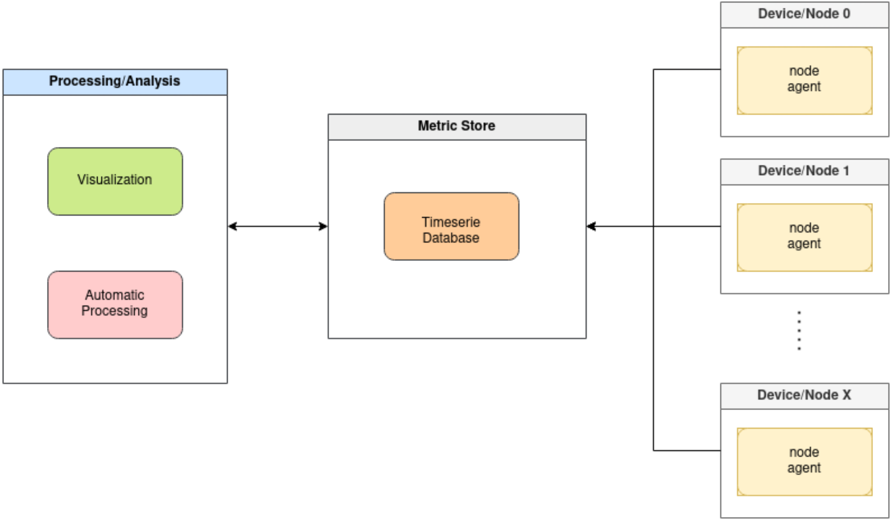
- Infrastructure: bandwidth used, node load, I/O load, storage used, . . .
- Job specific: CPU load, bandwidth used, I/O Load, storage/memory used . . .
- Energy consumption metrics to be associated later with jobs etc.

Monitoring infrastructure in HPC

Monitoring stacks usually consists of three components/stages:

- Metrics collector:
  - collect
  - aggregate/pre-process
  - forward/store/provide metrics to database
- Database: collects, stores and provides data (relational, time-series)
  - collect/stores metrics from collectors
  - provides the collected data via APIs for further processing
  - can have retention handling etc.
- Processing: tools accessing the data and the database to
  - perform automatic data analysis
  - visual display of data to allow human analysis of the performance/data

Monitoring infrastructure in HPC 2



Components: metrics collector

- task: collect, aggregate/preprocess and provide data to the database
- usually a local process run as service
- usually extendable/utilizing other tools to get metrics
- main approach for providing data:
  - **pull** - agent waits for the database to ask for the data
  - **push** - agent pushes collected data automatically to database
- examples: PCP, Telegraf, node-exporter, cc-exporter

Components: metrics database

- follow two main types
  - relational DB
  - time-series DB
- offer the typical DB functionality to work in enterprise (fleets, sharding)
- may offer functionality specifc to time series data, e.g. data retention, compaction
- examples: InfluxDB, Prometheus, OpenTSDB

Components: metrics database - TSDB

- developed specific for storing time series data
- time series data consists of
  - time-stamps
  - tags identifying the metrics/other metadata
  - data/values
- time series data for a specific measurement are
  - highly uniform
  - have no relation/are standalone
  - easily transferable using text
- uniformity/structure allow for specific optimimzation, e.g.
  - compression
  - transfer rate/ingestion

Components: TSDB vs. relational DB

Relational DB

- Defined table format/schema
- Develop for large amount of data with complex relations
- Fast to process SQL queries on huge sets
- Fast to operate on entries (e.g. modify/delete)
- Difficult to store large amounts of uniform data at the same time

Time-series DB

- Flexible format, no fixed schema
- Developed for storing large amounts of uniform data at the same time
- Fast to store the data
- Slow to operate on data in the DB (e.g., modify/delete entries)

Components: processing

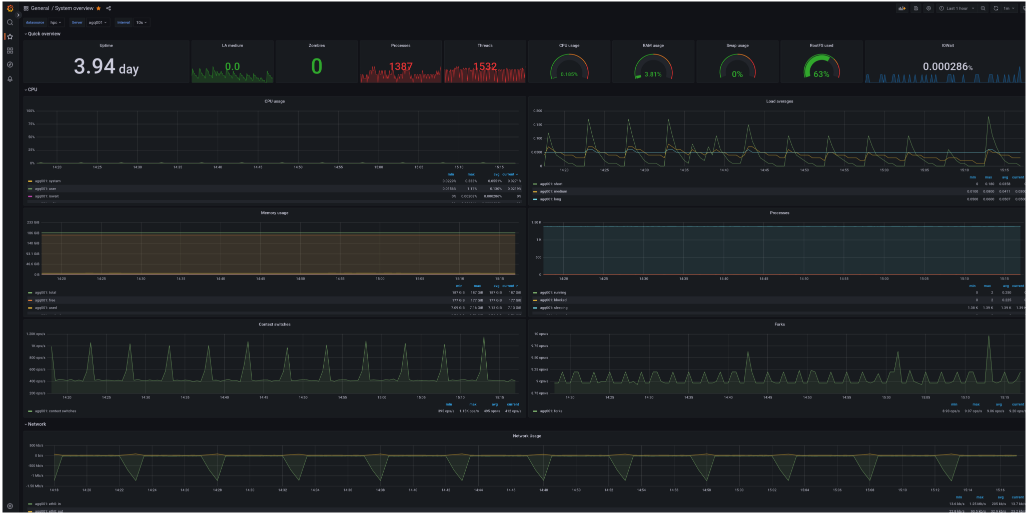
- the collected data can be processed by anything imaginable:
  - scripts
  - programms
  - visualization software
- use cases, e.g.:
  - regressions testing (compare system performance before/after changes using)
  - degeneration - loss of performance due to factors like age, overheating
  - report generation
  - analyse program performance
- visualization examples: Grafana, Kibana, Graphite, Cyclotron, matplotlib, . . .

TIG Stack

Abbreviation for

- **Telegraph** - the node agent collecting metrics on the systems
- **InfluxDB** - a time-series database used to store the data
- **Grafana** - a tool to visualize metrics and create dashboards

TIG Dashboard example



CPU usage

	min	max	avg	current
agq001: system	0.0229%	0.333%	0.0551%	0.0271%
agq001: user	0.0156%	1.17%	0.130%	0.0219%
agq001: iowait	0%	0.00208%	0.000286%	0%

Load averages

	min	max	avg	current
agq001: short	0	0.180	0.0358	
agq001: medium	0.0100	0.0800	0.0411	0.0300
agq001: long	0.0500	0.0600	0.0507	0.0500

Memory usage

	min	max	avg	current
agq001: total	187 GiB	187 GiB	187 GiB	187 GiB
agq001: free	177 GiB	177 GiB	177 GiB	177 GiB
agq001: used	7.09 GiB	7.16 GiB	7.13 GiB	7.13 GiB

Processes

	min	max	avg	current
agq001: running	0	2	0.250	
agq001: blocked	0	2	0.225	
agq001: sleeping	1.38 K	1.39 K	1.39 K	1.39 K

Context switches

	min	max	avg	current
agq001: context switches	395 ops/s	1.15K ops/s	495 ops/s	412 ops/s

Forks

	min	max	avg	current
agq001: forks	8.93 ops/s	9.97 ops/s	9.06 ops/s	9.20 ops/s

Network Usage

	min	max	avg	current
agq001: eth0: in	13.6 kb/s	1.25 Mb/s	205 kb/s	13.7 kb/s
agq001: eth0: out	22.8 kb/s	90.5 kb/s	32.9 kb/s	23.2 kb/s

InfluxDB

- Developed by InfluxData
- First released 2013
- Written in Go for performance
- Time-series based instead of relational
- Design specifically for time-series from all kind of devices
- Query language Flux (similar to SQL)
- push approach
- open-core

Telegraf

- Developed by InfluxData
- Node agent for InfluxDB
- Tasks: collecting, processing, aggregating, writing metrics
- Extendable architecture based on plugins
- As InfluxDB written in Go for max. performance
- Is run as service on the monitored systems

Telegraf - Plugins

- Telegraf uses plugins to collect and send data
- the plugins utilize different tools to collect metrics e.g.
  - Likwid to collect hardware counters from CPUs
  - Files in /procfs for process statistics
- 2 plugin types: input and output
- At least one input and one output plugin has to be configured e.g.
  - Input: CPU, diskio, procstat
  - Output: influxdbv2
- Huge variety of plugins can be found the Plugin documentation

Grafana

- Developed by GrafanaLabs
- Web based dashboard application to visualize data
- Written in Typescript, Go
- build in webserver based on Go http server
- Supports relational, time-series databases
- open-core
- Can display real-time data using specific plugins/config (MQTT+Telegraf)
- Goals: easy visual analysis of data from different sources, easy dashboard creation

Installation of TIG

- Tutorial 1
- Tutorial 2
- Tutorial 3
- Tutorial x

Plenary discussion

- issues
- impression/experience
- questions/open points?

References

- https://www.influxdata.com/
- https://grafana.com/
- https://docs.influxdata.com/telegraf/v1.20/plugins/