Karlsruhe Institute of Technology

Future Plans

# LOGGING AND JOB MANAGEMENT

2019; AUgust 28th

Nils Braun | IETP - KIT

# 1.

# LOGGING AND MONITORING

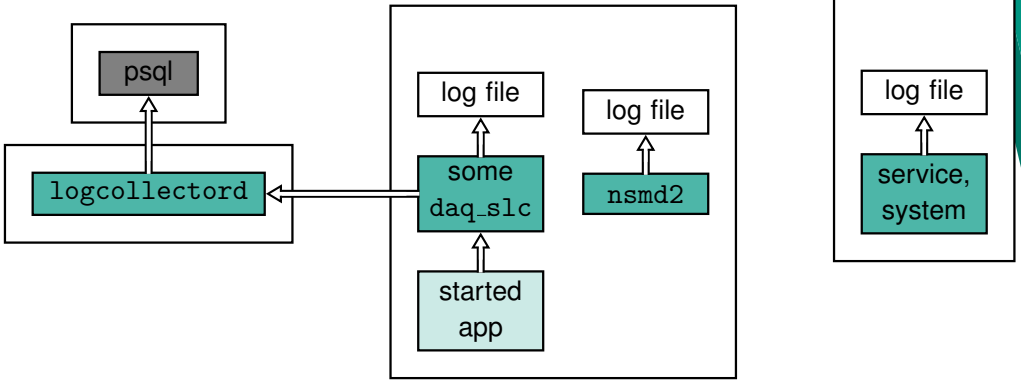
# STATUS & PROBLEMS

- Three different things to log
  - `daq_slc` applications (and applications started by them) log to file and send the log messages via `nsmd2` to the log collector, which writes them to the postgres DB
  - `nsmd2` itself logs to file
  - everything else (system services, system itself, `cvmfs` server, `zabbix`) also writes to files
- Problems:
  - Showing the logs is a heavy burden on the postgres DB (according to Yamagata-san)
    - Postgres is in principle not built for this workload
    - no possibility to do quick full text search
    - aggregations are possible but complicated
  - How do I see trends?
  - How can I check (system) services without logging in into each machine?
    - Example: does the `cvmfs` mirror only fail here or on all mirrors at the same time?

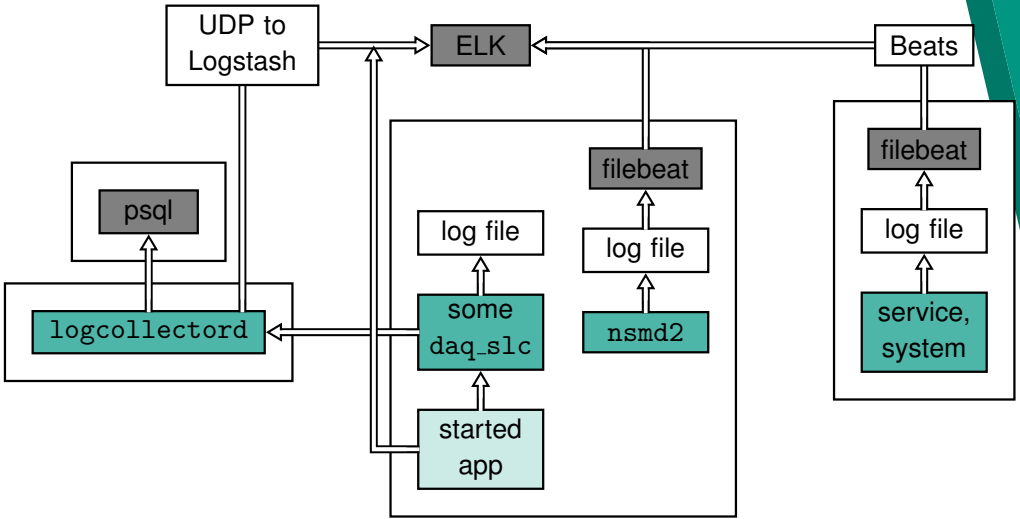
# LOGGING: ELK

- Many companies and data centers (also universities, grid) employ the ELK stack:
  - Elasticsearch and its lucene database for storing (it is basically a database service like postgres)
  - Logstash or filebeat for writing into the database (many many possibilities, but in principle a service accepting logs via pull or push)
  - Kibana for nice visualizations and dashboards
- Also used by PXD with good experience so far (as far as I know)
- Simple test setup:
  - Single VM with all three applications
  - Installed via `rpm` packages, basically no configuration needed
    - Enabled UDP input (see below)
    - Added users and passwords
  - Sent HLT unit `daq_slc` logs, `cvmfs` mirrors, `basf2` for testing

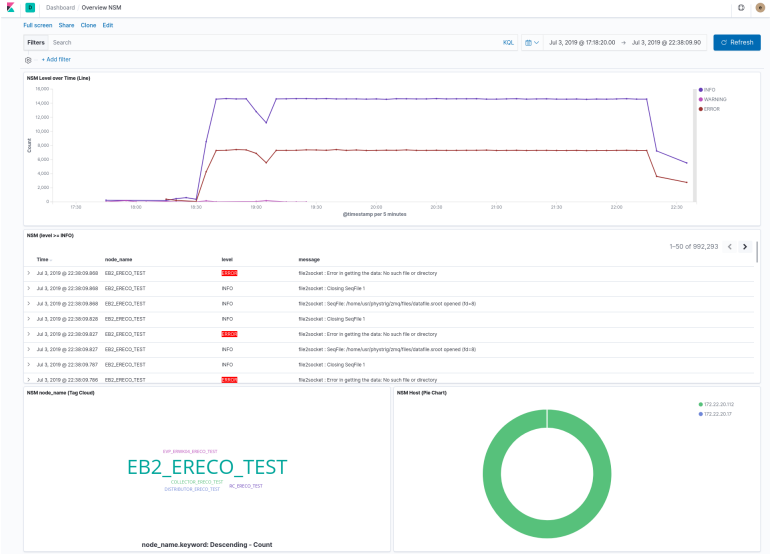
# CURRENT SCHEMA



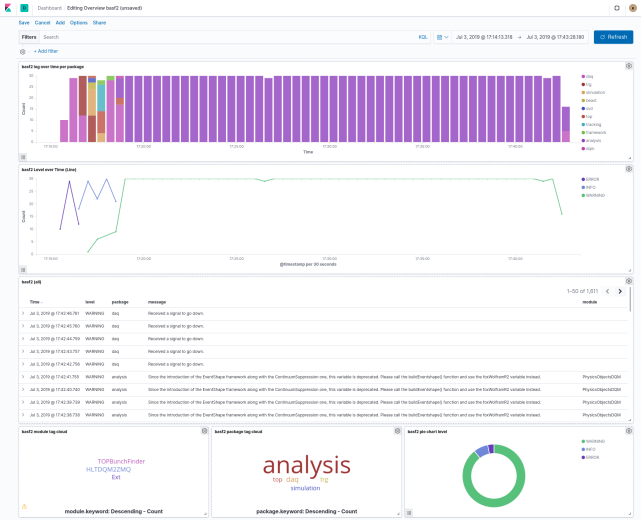
# PROPOSED SCHEMA



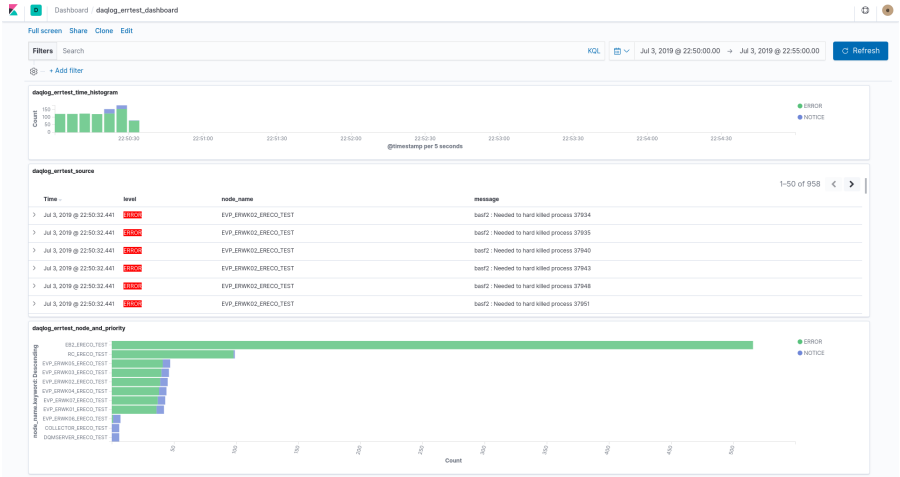
# SCREENSHOTS



# SCREENSHOTS



# SCREENSHOTS



# BENEFITS AND POSSIBLE NEXT STEPS

- Another industry standard
  - Lots of documentation, many tutorials, just works
- Log aggregation and beautiful user interface built in, also runs in kiosk mode
- Can handle logs properly in arbitrary sizes
- First implementation done, first experience also by Yamagata-san (the last screenshot is actually from him ;-) )
- If we want that:
  - proper VM setup
  - Do we want replication? Backup? How long do we want to store the logs?
- Check real world impact on VM (so far, very little computational power needed)

# 2.

# JOB MANAGEMENT

# JOB MANAGEMENT

In *principle* all our machines do the same:

- Run some core software (e.g. zabbix, nsmd2)
- Run some slow control applications (e.g. `rocontrold`)
- Run some additional user scripts etc.

Having to maintain each machine in its own has clearly some disadvantages, with `cvmfs` we did a first step to move most of the binaries and configuration to a common space.
How about the job management?

# HOW DO OTHERS DO THAT?

There exists a lot of experience for exactly this in industry. Companies such as Google, Amazon, Facebook normally do something like the following:

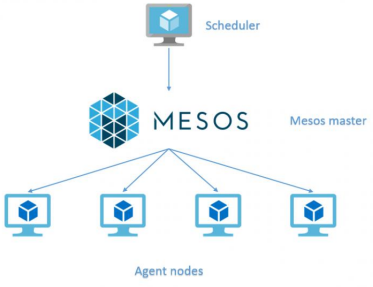
- Share binaries and configuration between all machines (basically have no local dependency) $\Rightarrow$ we already have that!
- Use a job scheduling system and define one a single source of truth which jobs should run where $\Rightarrow$ this talk
- Monitor and react on each state change $\Rightarrow$ the infrastructure is there!

By this, they can run literally thousands of workloads in their computing centers.

Do we also want this?

# A DREAM WORLD WITH APACHE MESOS/AURORA

- Start a mesos slave on each machine on system start, give additional attributes to each machine to identify it (basically: hostname)
- Share binaries via `cvmfs`
- Define in a single python configuration file which jobs should run on which machine
  - Also define things like `daq_slc` version in this file
- Let Apache Mesos handle the rest:
  - Automatic start and restart (configurable)
  - Configurable heartbeat and alive checking
  - Integration with any logging service we want
  - Nice monitoring built in, integration with zabbix etc.
  - Update procedures
- Reproducible, simple, less chance for mistakes



# MORE DETAILS: SCHEDULER

- Every Mesos cluster has one (actually: a few for fault-tolerance) main node, the scheduler
- Speaks with every machine and tells them to (re)start apps
  - Still not a large network bandwidth needed as only commands are sent (binaries etc. stay local)
- Does the monitoring etc.
- Also has nice web-based UIs
- Needs to have connection to every machine
  - Could be achieved simply by network forwarding rules
- Single source of truth which applications should run were
  - Just change one definition file
  - Also makes updated very simple

# MORE DETAILS: SLAVES

- On system start: start the Mesos executor service daemon
- Run everything defined by the scheduler:
  - Services (e.g. logging)
  - nsmd2
  - `daq_slc` apps
- Can of course still host ”normal” users and their jobs
- Some questions:
  - How to handle manual tests?
  - Who has access rights on the scheduler machine?
  - AUtomatic restarts or just notice?

# MORE DETAILS: WORKFLOW

- Cold Restart (or new machine):
  - Spin up Mesos executor and `cvmfs`
  - Start jobs on scheduler
  - Done!
- New `daq_slc`, `basf2` or nsmd2 version:
  - Just change single variable in job definition file
  - Let Mesos handle the update automatically
- Additional service (also during transition period)
  - Just add it to the single definition file
- Logging and Monitoring
  - Can be started as standalone ”job”
  - Most of the monitoring is not needed anymore

# IS IT WORTH IT?

- I do not know, but the DAQ upgrade could be a good opportunity
  - Building such a system is also a good way to get to know all components...
- I had a running Mesos setup ready in approx. 5 minutes (excluding time for package installation) on a virtual test system
- Mesos knows how to achieve high availability
- Only drawback: does not run on SL5 (as far as I know)
- Other experiments (e.g. CERN experiments) use similar techniques

In the end, it will probably not me to implement such a system but it is worth thinking about it