
Real-time monitoring Slurm jobs with InfluxDB

September 2016

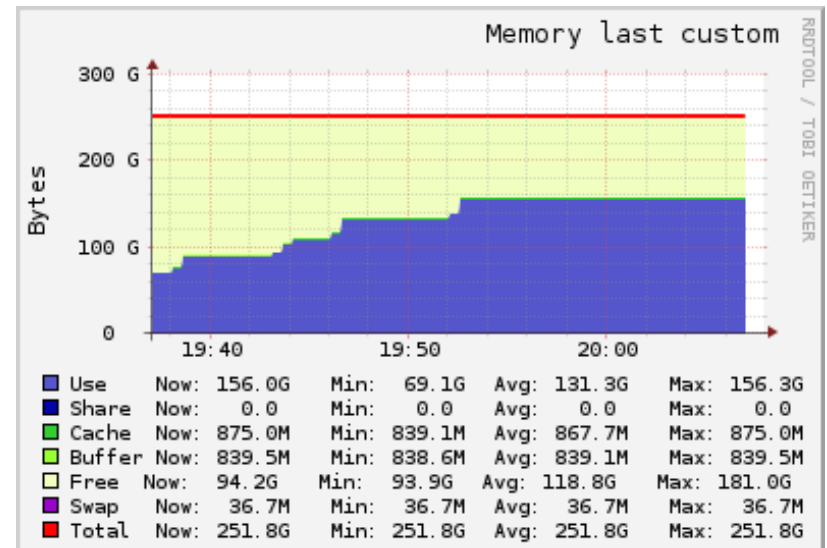
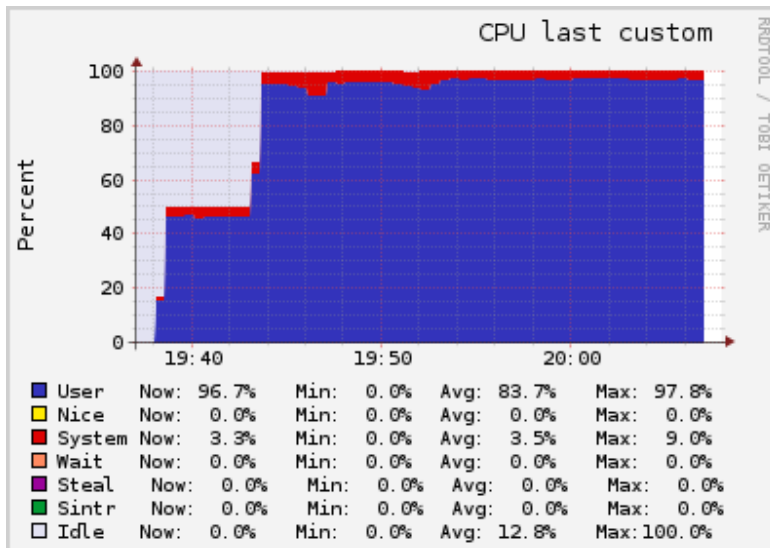
Carlos Fenoy García

Agenda

- Problem description
- Current Slurm profiling
- Our solution
- Conclusions

Problem description

- Monitoring of jobs is becoming more difficult with new systems with higher amount of resources as jobs tend to share compute nodes.
- “Standard” monitoring tools hide the individual job usage in the compute host resource monitoring



Current Slurm profiling

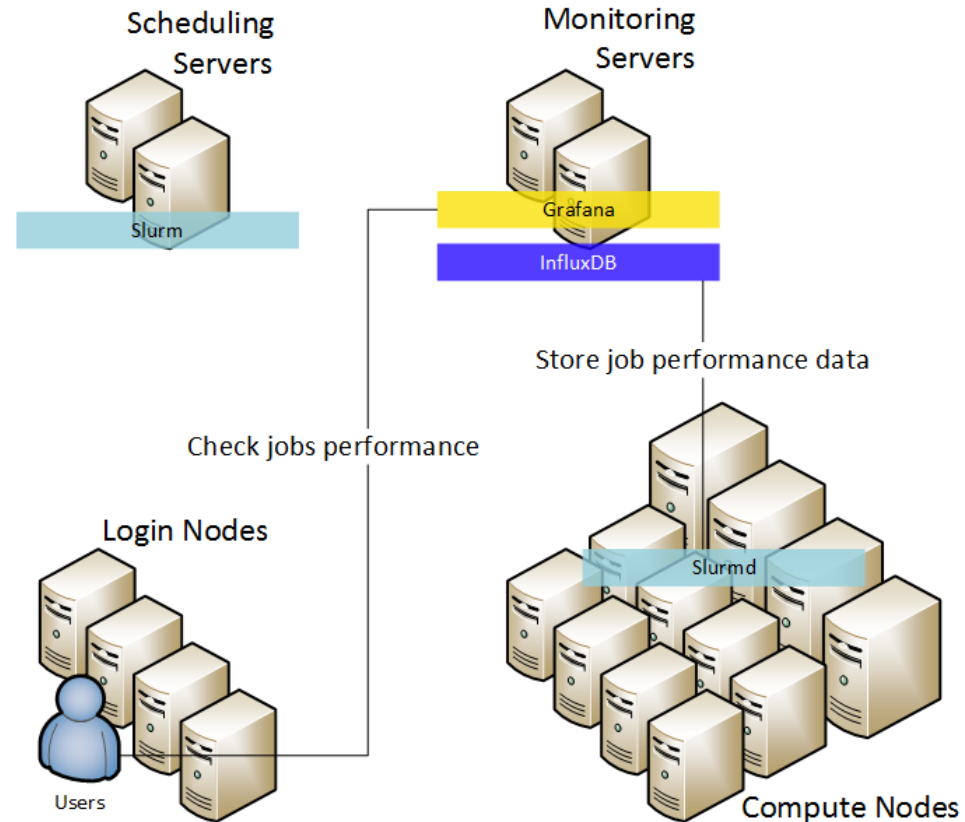
- Slurm support profiling of applications using HDF5 as storage
 - It gets resource usage every few seconds
 - Stores the information in an HDF5 file per host
 - Once the job is finished the users have to merge all the .hd5 files to create a single per job file

Current Slurm profiling (II)

- Pros
 - No need for a central monitoring storage or to send data through network
 - Uses the existing shared filesystem
 - Light-weight collection and storage of data
- Cons
 - If one node dies, the HDF5 file may be corrupt and irrecoverable
 - No data can be retrieved until the job finishes
 - Filesystem can not be mounted with root squash

Our solution

- Using the same base as the HDF5 profiling plugin, export the information to an InfluxDB server
- Collects exactly the same information as the HDF5 plugin
- A small buffer is used to avoid sending data for every sample collected
- Information is sent to the central server using libcurl



InfluxDB and Grafana

- “InfluxDB is an open source database written in Go specifically to handle time series data with high availability and high performance requirements.”
influxdata.com
- InfluxDB has a REST API to insert and query data
- Integrated with Grafana for nice dashboards



Metrics collected

Default metrics:

CPUFrequency	RSS
CPUTime	ReadMB
CPUUtilization	WriteMB
Pages	

Additional profiling plugins it is possible to collect information from Infiniband, Lustre and Energy

Configuration

- 3 new parameters added to the `acct_gather.conf` file
 - `ProfileInfluxDBHost`: the host where to send the data to
 - `ProfileInfluxDBDatabase`: the database in influx where to store the data
 - `ProfileInfluxDBDefault`: Default profiling level
- Default profiling level set to `ALL` if nothing else specified to be able to also collect information from the job script

Sending data to InfluxDB

- A small 16KB buffer is used to aggregate some data before sending
- The influx line protocol is used to send the data
 - METRIC,(TAGS) value=VALUE (TIMESTAMP)
 - CPUTime job=24,step=1,task=2,host=node001 value=99 1460713153
- Floating point data is sent with 2 decimals precision

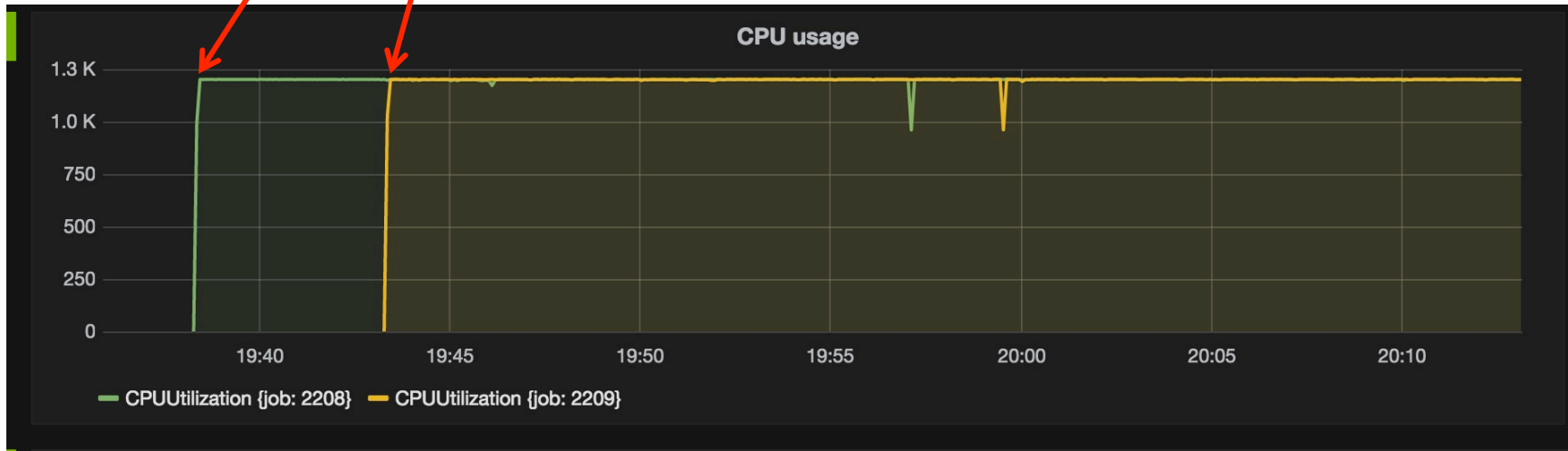
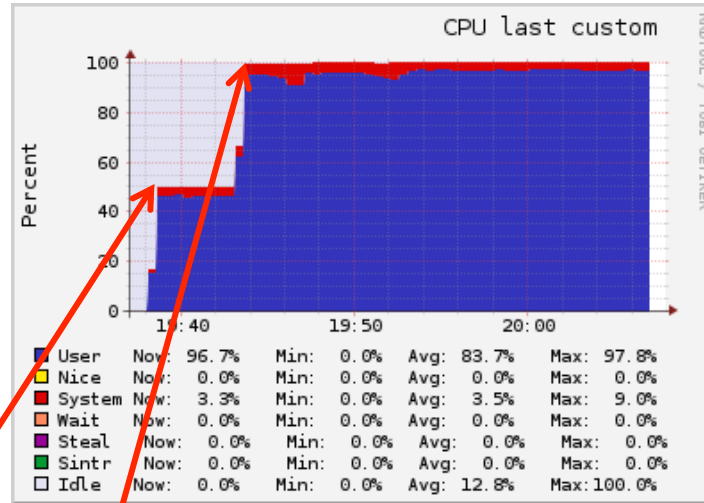
Sending data (II)

- Information is sent through curl to the database server
 - `INFLUXDB_SERVER/write?db=slurm&rp=default&precision=s`
 - If an error is returned by the server the data is dropped
 - Some profiling data may be lost
- You can also send the data to a Logstash server to store it in a different DB.

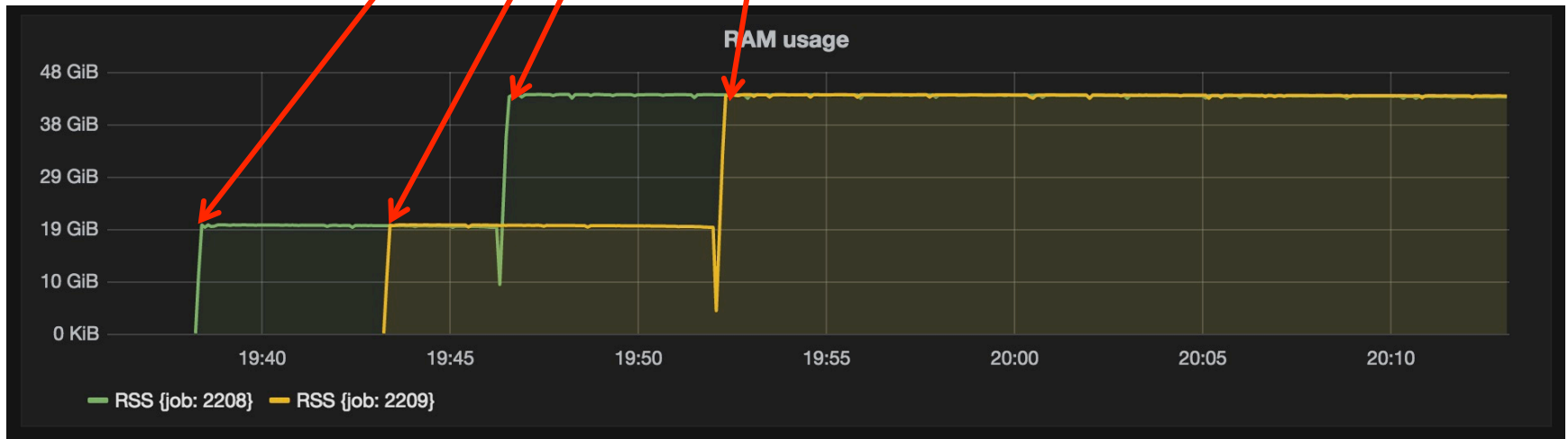
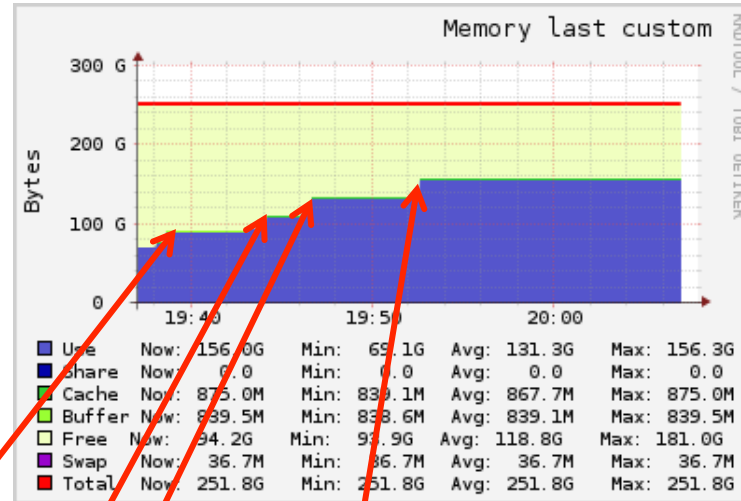
Our solution (II)

- Pros
 - Light-weight collection and storage of data
 - All the information is available almost in real-time
 - No information stored locally on the nodes, and no possibility of data corruption due to a node crash
 - Information available per job/task enhances understanding of the usage
- Cons
 - Needs a central server where to send all the collected data.

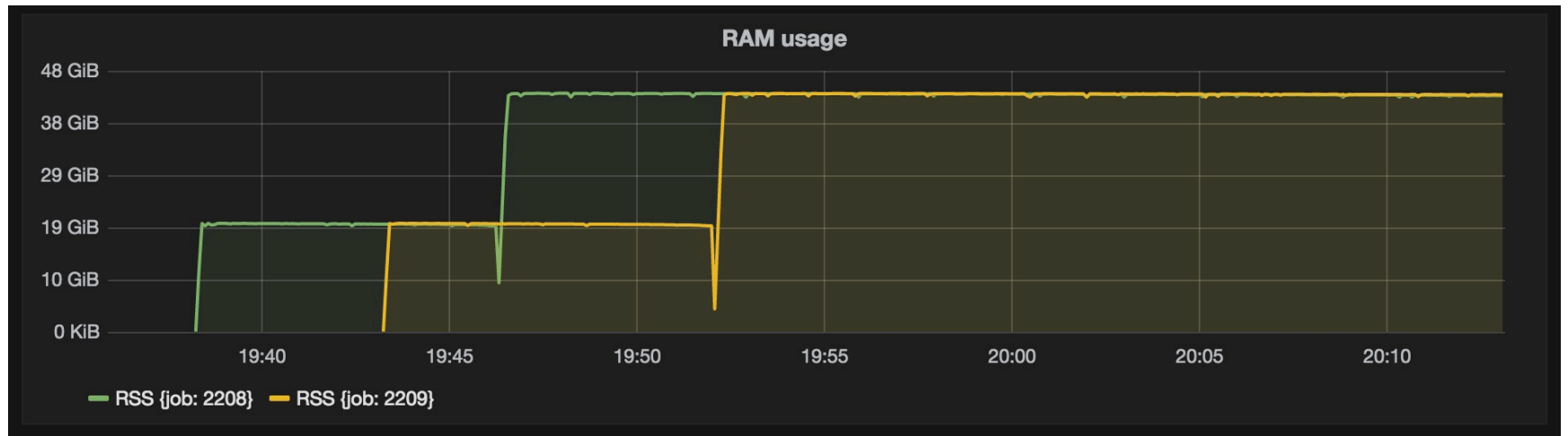
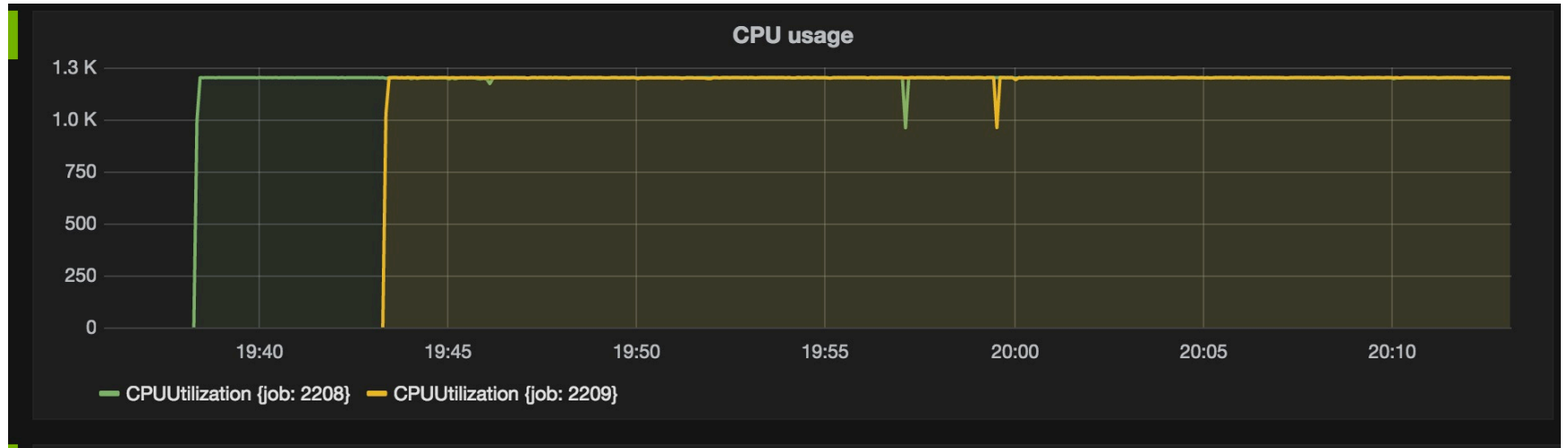
Examples



Examples



Examples



Conclusions

- Easy to setup monitoring system
 - 1 daemon
 - 1 config file in the compute nodes
- Real-time monitoring => faster reactions to issues
- Better monitoring => better understanding of the usage of the cluster
- Monitoring information related to jobs and not only nodes

GITHUB

<https://github.com/cfenoy/influxdb-slurm-monitoring>

References

- InfluxDB: <http://www.influxdata.com>
- Grafana: <http://www.grafana.org>
- Slurm: <http://slurm.schedmd.com>
- Slurm profiling: http://slurm.schedmd.com/hdf5_profile_user_guide.html

Doing now what patients need next