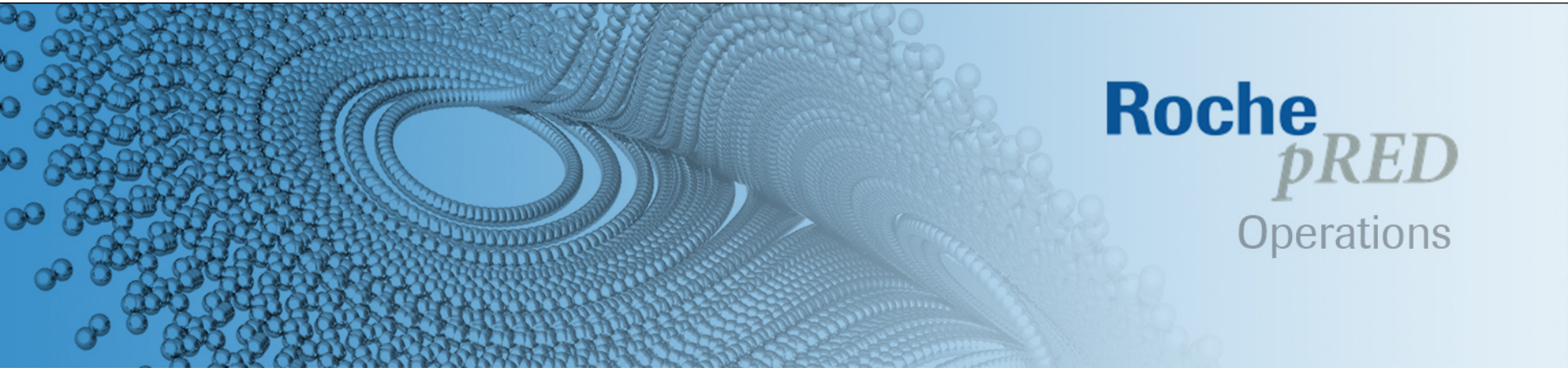
Real-time monitoring Slurm jobs with InfluxDB
September 2016

Carlos Fenoy García

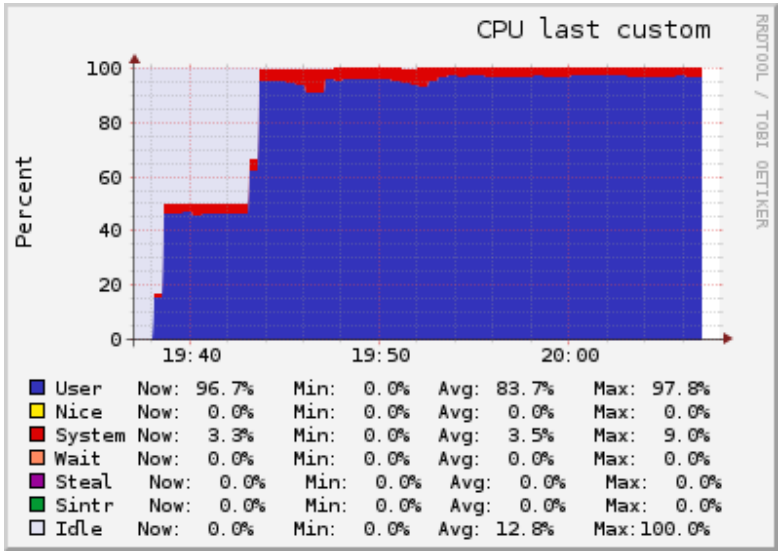


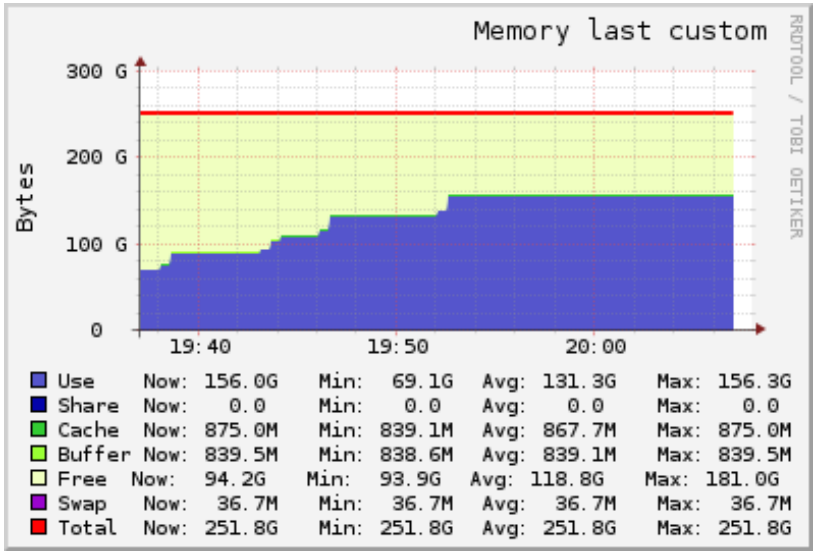
Agenda

- 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

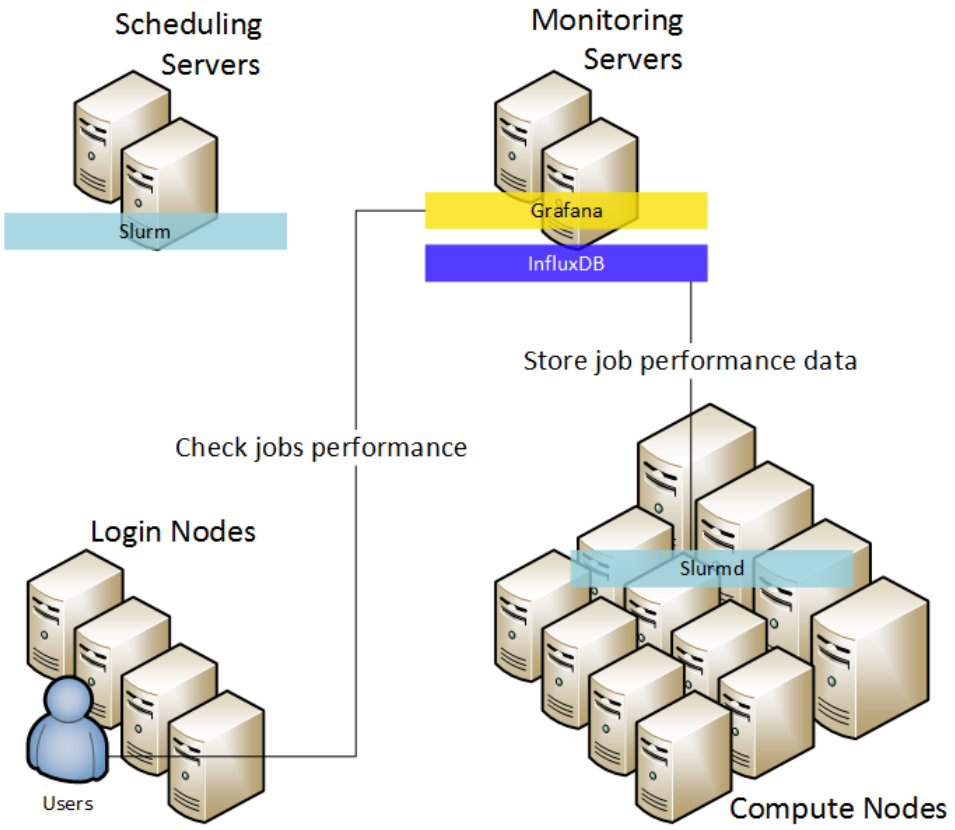
- Slurm support profiling of applications using HDF5 as storage
  - It gets resource usage every few seconds
  - Stores the information in an HDF5 file per host
  - Once the job is finished the users have to merge all the .hd5 files to create a single per job file

Current Slurm profiling (II)

- Pros
  - No need for a central monitoring storage or to send data though 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

Roche

- “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 precission

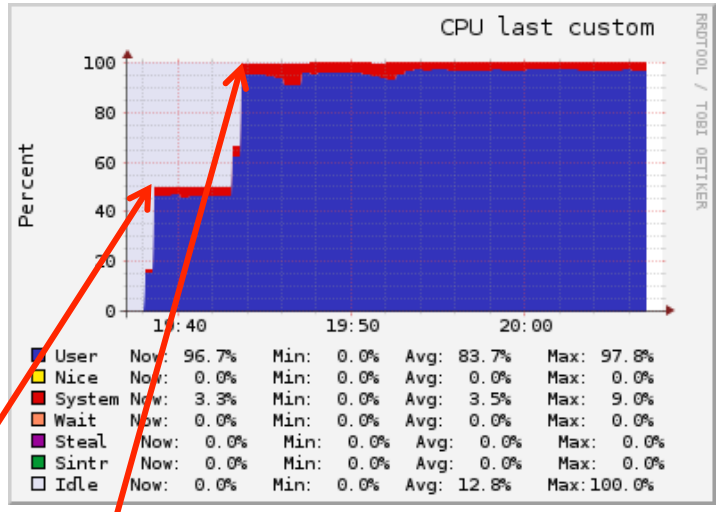
Sending data (II)

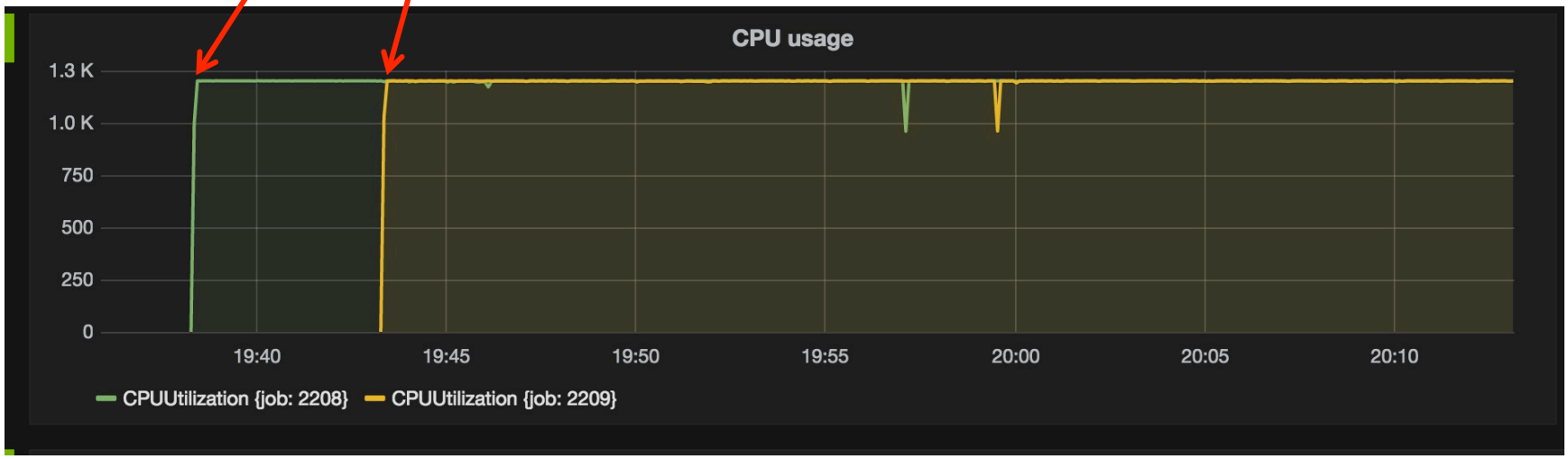
- Information is sent through curl to the database server
  - INFLUXDB_SERVER/write?db=slurm&rp=default&precision=s
  - If an error is returned by the server the data is dropped
  - Some profiling data may be lost
- You can also send the data to a Logstash server to store it in a different DB.

Our solution (II)

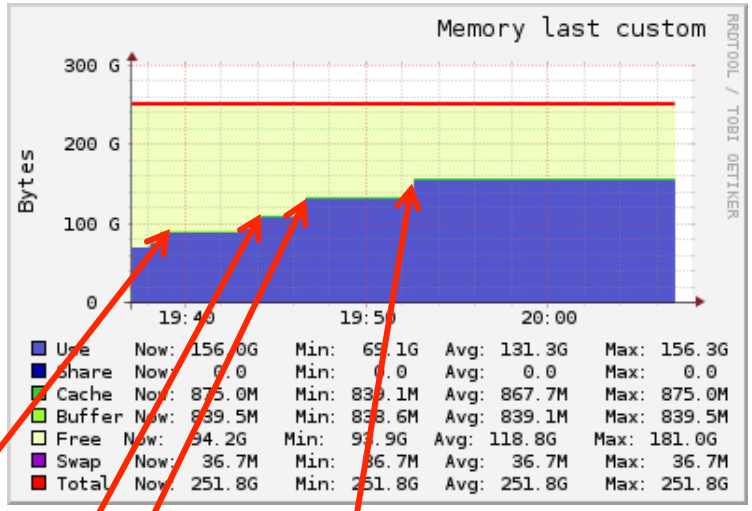
- Pros
  - Light-weight collection and storage of data
  - All the information is available almost in real-time
  - No information stored locally on the nodes, and no possibility of data corruption due to a node crash
  - Information available per job/task enhances understanding of the usage
- Cons
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

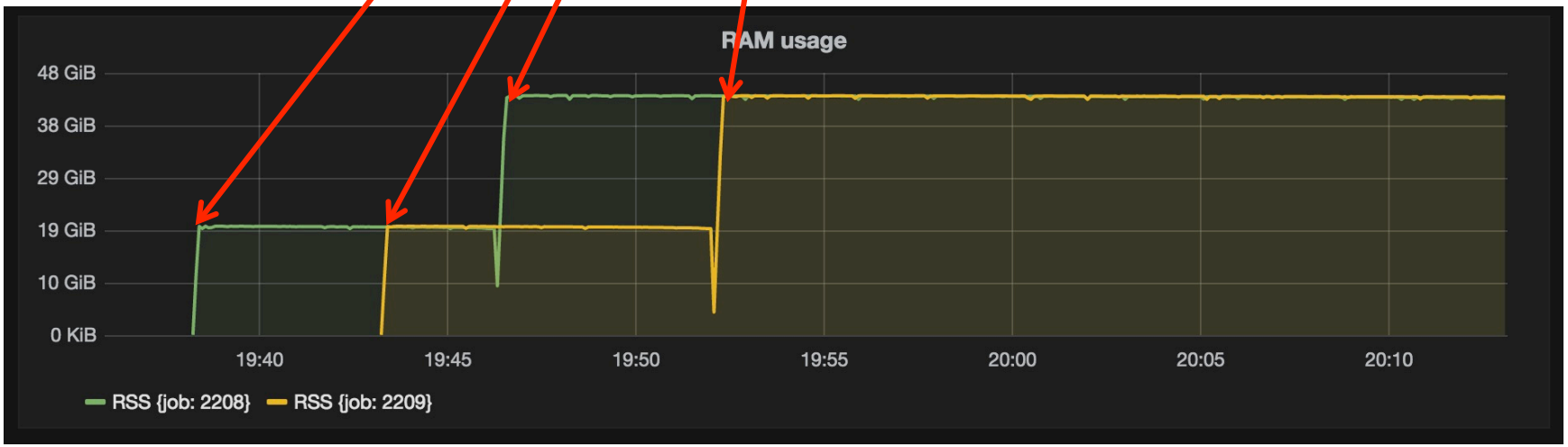
Examples


CPU last custom
RRDTOOL / TOBI OETIKER
Percent
100
80
60
40
20
0
19:40
19:50
20:00
User Now: 96.7% Min: 0.0% Avg: 83.7% Max: 97.8%
Nice Now: 0.0% Min: 0.0% Avg: 0.0% Max: 0.0%
System Now: 3.3% Min: 0.0% Avg: 3.5% Max: 9.0%
Wait Now: 0.0% Min: 0.0% Avg: 0.0% Max: 0.0%
Steal Now: 0.0% Min: 0.0% Avg: 0.0% Max: 0.0%
Sintr Now: 0.0% Min: 0.0% Avg: 0.0% Max: 0.0%
Idle Now: 0.0% Min: 0.0% Avg: 12.8% Max: 100.0%

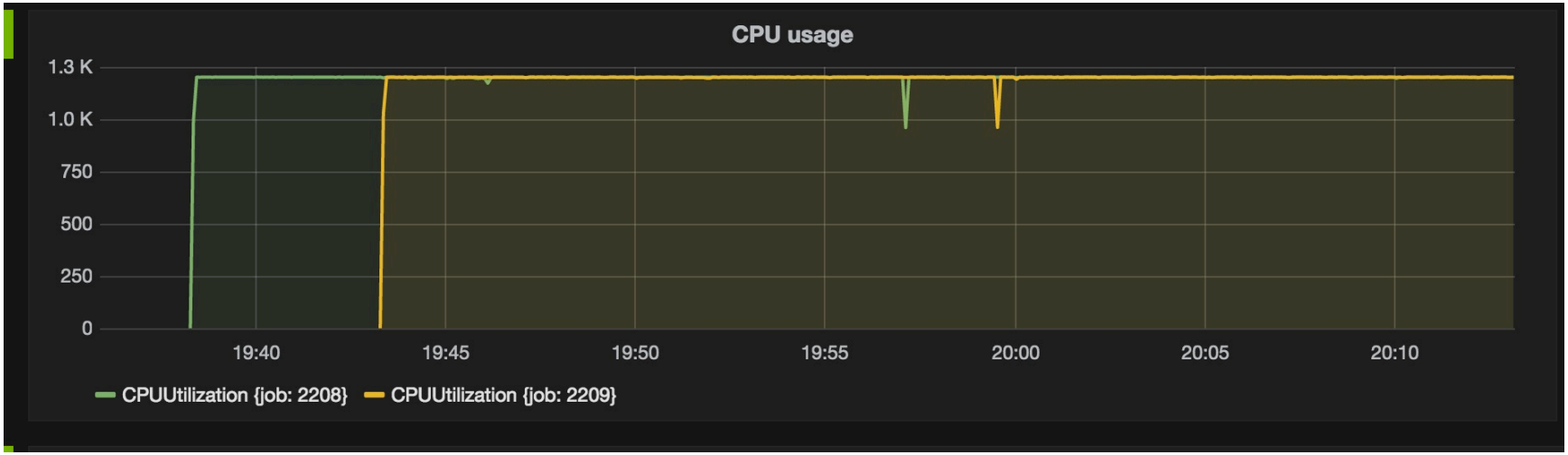

CPU usage
1.3 K
1.0 K
750
500
250
0
19:40
19:45
19:50
19:55
20:00
20:05
20:10
CPUUtilization {job: 2208}
CPUUtilization {job: 2209}

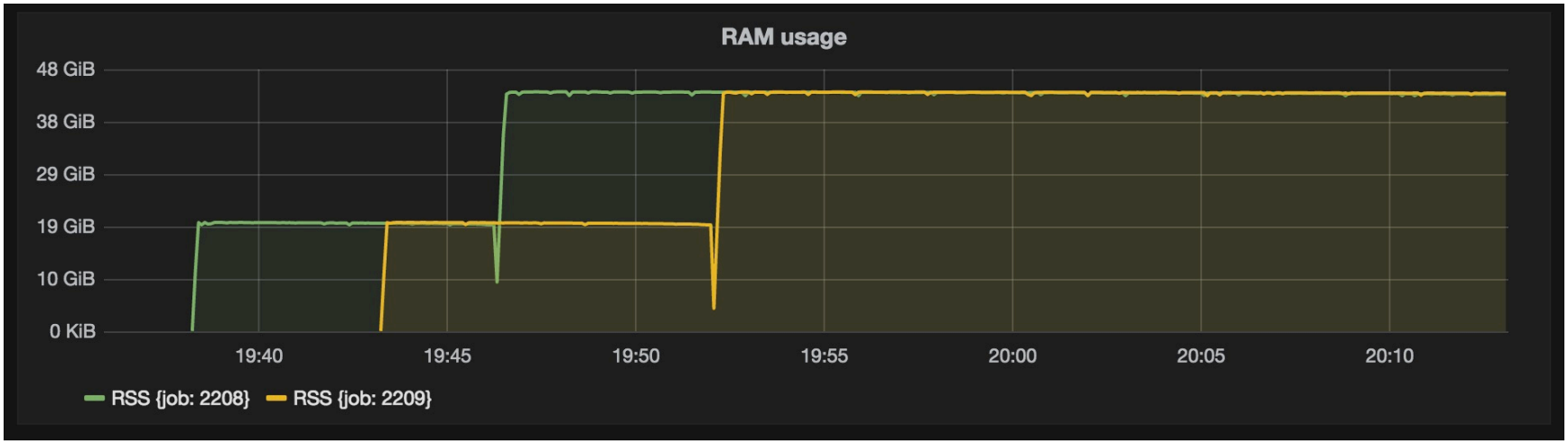
Examples


Memory last custom
RRDTOOL / TOBI OETIKER
Bytes
300 G
200 G
100 G
0
19:40
19:50
20:00
Use Now: 156.0G Min: 69.1G Avg: 131.3G Max: 156.3G
Share Now: 0.0 Min: 0.0 Avg: 0.0 Max: 0.0
Cache Now: 875.0M Min: 830.1M Avg: 867.7M Max: 875.0M
Buffer Now: 839.5M Min: 838.6M Avg: 839.1M Max: 839.5M
Free Now: 94.2G Min: 93.9G Avg: 118.8G Max: 181.0G
Swap Now: 36.7M Min: 36.7M Avg: 36.7M Max: 36.7M
Total Now: 251.8G Min: 251.8G Avg: 251.8G Max: 251.8G


RAM usage
48 GiB
38 GiB
29 GiB
19 GiB
10 GiB
0 KiB
19:40
19:45
19:50
19:55
20:00
20:05
20:10
RSS {job: 2208}
RSS {job: 2209}

Examples


CPU usage
1.3 K
1.0 K
750
500
250
0
19:40
19:45
19:50
19:55
20:00
20:05
20:10
CPUUtilization {job: 2208}
CPUUtilization {job: 2209}


RAM usage
48 GiB
38 GiB
29 GiB
19 GiB
10 GiB
0 KiB
19:40
19:45
19:50
19:55
20:00
20:05
20:10
RSS {job: 2208}
RSS {job: 2209}

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