The ATLAS data management system Rucio: *Horizontal scalability and failsafe deployment*

Mario Lassnig and Ralph Vigne, CERN PH-ADP, on behalf of the ATLAS Collaboration

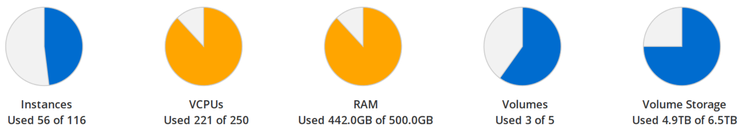
Rucio

Rucio manages all analysis, production, and user data for the ATLAS experiment. It is one of the technical underpinnings of the ATLAS Distributed Computing project, and provides a unified interface to globally distributed data. The Rucio components are in charge of indexing, querying, transferring, deleting, and monitoring. Reliability and performance are the priorities.

Scalability and deployment

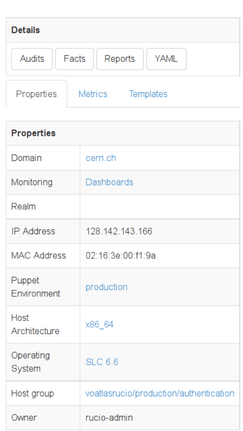
Many components of the ATLAS Distributed Computing project, e.g., the workflow system PanDA, or the HammerCloud testing framework, require a 24/7 service for data management. Even a few minutes downtime can cause jobs to fail. It is therefore imperative to have a data management deployment that is scalable and fail-safe. Different components of Rucio require different kinds of scalability and fault-tolerance though, so selective customization, while striving for uniformity, was the main challenge.

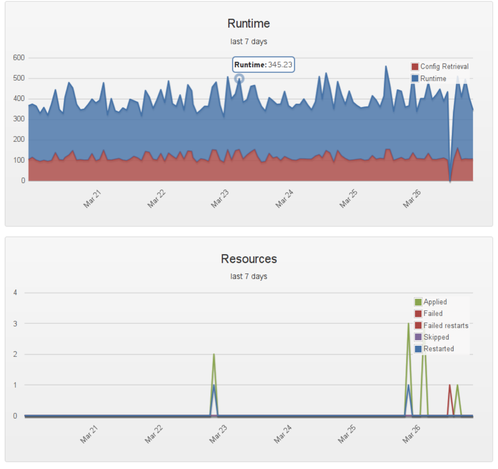
Size of the deployment



The deployment is fully virtualized in the CERN Computing Centre. All nodes are virtual machines instances from OpenStack, and configured automatically with Puppet. The instances are clustered into their actual usage patterns: load balancing, authentication, catalogues, services, and monitoring. A duplicate deployment for integration and pre-production is run alongside, however, with a reduced number of virtual machines. High volume storage is provided through attached virtual drives using a Ceph object storage backend.

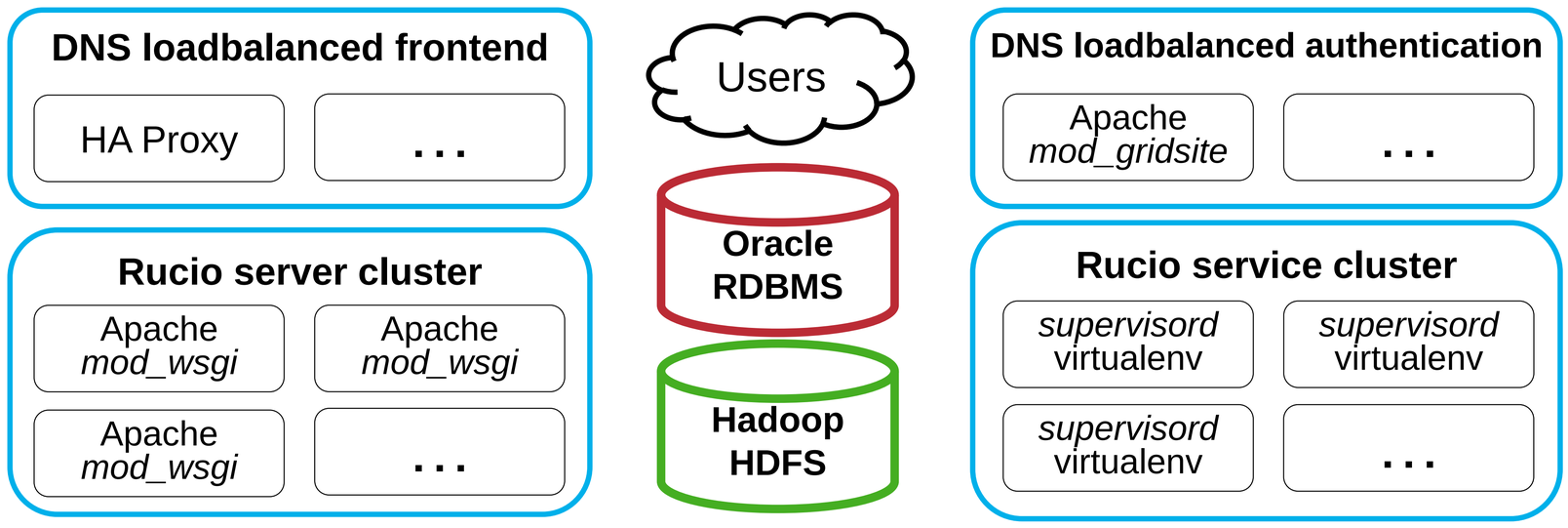
Configuration management





Configuration of the nodes is done via Puppet, a declarative client/server management system. A Ruby-like language describes which software should and should not be installed, and how it must be configured. Puppet will ensure that this configuration is enforced at all times.

Schema of the deployment

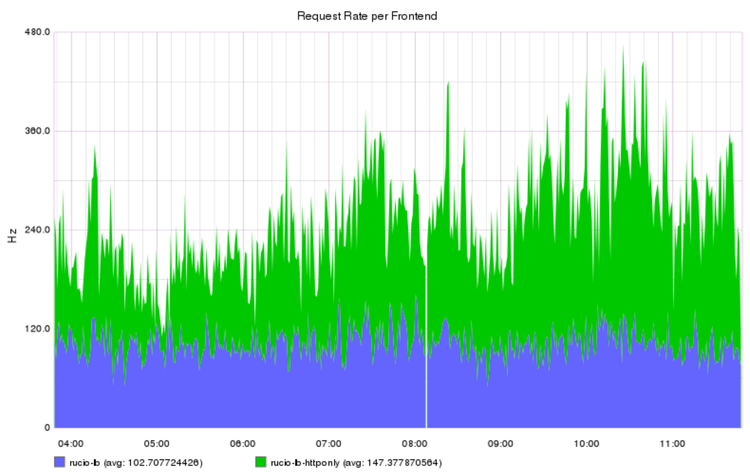


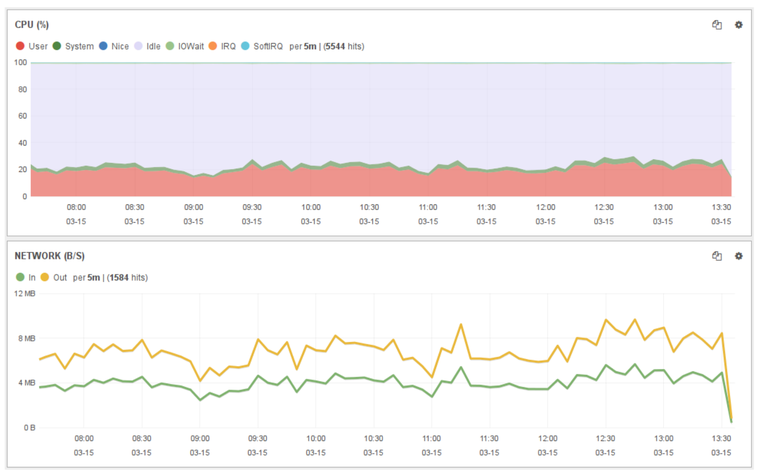
All virtual machine configurations are clustered as elastic Puppet hostgroups. All nodes in a hostgroup are stateless, and can thus be instanced depending on the intensity of the workload. The client-facing interface for the Rucio servers is not managed by DNS, but by a dedicated cluster of HAProxy. This allows sophisticated load-balancing rules. The Rucio authentication cannot be proxied. The service cluster runs the Rucio daemons, which work asynchronously. Each of these services can be configured to run on independent sets of input data, to ensure horizontal scalability on the application level. All state is preserved on the database, and simply requires a restart of the service without any loss of data. The load on the database is throttled even when the intensity and rate of requests is higher than usual. Hadoop is used as the backend for storing monitoring data and content snapshots.

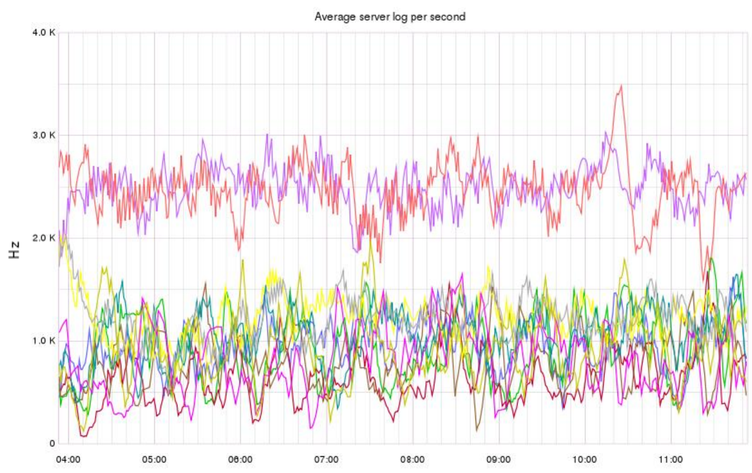
Powered by

openstack CLOUD SOFTWARE · puppet labs · HAPROXY

System load characteristics







System load is monitored in two ways: application level monitoring is done in Graphite (left) by Rucio, and node level monitoring in ElasticSearch/Kibana (right) by CERN IT. The Graphite data is populated by the application code directly. Servers and services send UDP packets to a central Graphite instance, which collects and prepares them. The node-level information is collected via Flume onto a separate CERN IT provided Hadoop cluster. All the nodes in the computing centre report there, and with Kibana customized views on different hostgroups are possible. Alarms are available for both systems.

There are two types of application load: user-generated, and service internal. Especially the user workload needs to be distributed as evenly as possible, as it shows very diverse characteristics (top left). The user load is evenly distributed by HAProxy onto the backend servers, regardless of their type (bottom left). It can be seen that there are actually two different types of workloads that are spread between two separate clusters of backend clusters. This is due to our main client PanDA, which brokers jobs based on available data. This requires a high interaction rate on small sets of data. In contrary, users generally list the contents of large datasets to make decisions on their future work. Despite this imbalance, the average CPU and network load is kept constant by HAProxy (top right), with enough capacity provisioned in OpenStack to support even much higher workloads. HAProxy can additionally shift nodes between groups, e.g., making an idle node that serves user request into a PanDA node on-the-fly, if HAProxy notices that PanDA workload is increasing.

References

- Garonne et al., Rucio – The next generation of large scale distributed system for ATLAS Data Management, 2014 J. Phys.: Conf. Ser. 513 042021
- Vigne et al., DDM Workload Emulation, 2014 J. Phys.: Conf. Ser. 513 042048