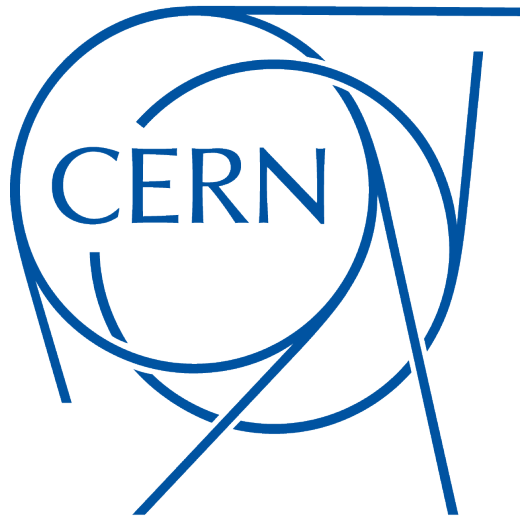


Migration of EOS SLS & availability probing scripts

Amine Riad REMACHE
(ea_remache@esi.dz)
Supervisor : Cristian Contescu



A work project report for the
Summer Student program, 2019

CERN, IT-ST-FDO
Geneva, Switzerland
July-August, 2019

Acknowledgements

I am so grateful to CERN and the donors, since I'm coming from a Non-member State, and all the Summer Student Team for the chance they gave me to live this life-changing experience. Then I would like to give a special thanks to my supervisor, Cristian Contescu and the amazing guy that replaced him during his holidays, Jan Iven for their patience and guidance through this project. I would like to thank also all the members of the section and different groups in the IT department that were here in the early days to make the integration go smoothly (Paul, Theo, Julien, Remy, Mihai and the others, this list could be so long). In the end, I'm sure I made some friendships that would last for a long time and I wish them a career full of success.

Abstract

In this report, we first introduce the Service Level Status (SLS) service, and the main concepts behind it. Then the architecture of the EOS storage service and the different protocols to probe.

Keywords : Service Level Status, EOS, Probing

1 Introduction

The need for a service status display is not something new to an IT department, on the contrary, it is mandatory for a large scale organization like CERN to have it. Service Level Status or SLS, is a web-based display that provides real-time information, statistics and availability of the different IT services. (Lopienski, 2008)

The EOS project was started in April 2010 in the CERN IT data storage group. with The main goal being to provide fast and reliable disk only storage technology for CERN LHC use cases. EOS manages over 250 PB of raw disk space and represents now the backbone of many CERN Services such as CERNBox. The native protocol is the XrootD protocol. (EOS, 2019)

During my time as a Summer Student here at CERN, my main role in this project was to migrate the current scripts for EOS SLS probing to Python and try to optimize it and make it scalable. My tasks went from writing the code to preparing the needed execution environment and testing the new scripts.

2 Definitions

2.1 Service Level Status (SLS)

"Given the diversity of computing services, SLS would not be able to measure availability of each service. Therefore, it is the manager of a given service who defines a way to calculate or estimate its availability, depending on its configuration, functionality, specific features etc. It means that SLS is not a monitoring system itself – it collects status and availability information for registered services, and displays it on the Web." - (Lopienski, 2008)

So as we can see in the given definition, SLS is not a monitoring system, but a kind of dashboard that needs input from the different services, such as EOS. SLS is based on concepts like :

- a. **Availability and status :** which is the measure to say if you can access a service and that it is working as expected, its state is in one of the three status levels : *available*, *degraded*, *unavailable* based on the percentage of the availability.

- b. **Key Performance Indicators** : KPIs are metrics that indicate whether a service meets its requirements (performance or utilization level). It is an extension of the concept of availability, as KPIs describe various aspects of a service, as compared to its expected performance. KPIs in SLS are pairs of two values: measured and target. If the measured value is worse than target value, the KPI shows that the service doesn't meet its requirements.

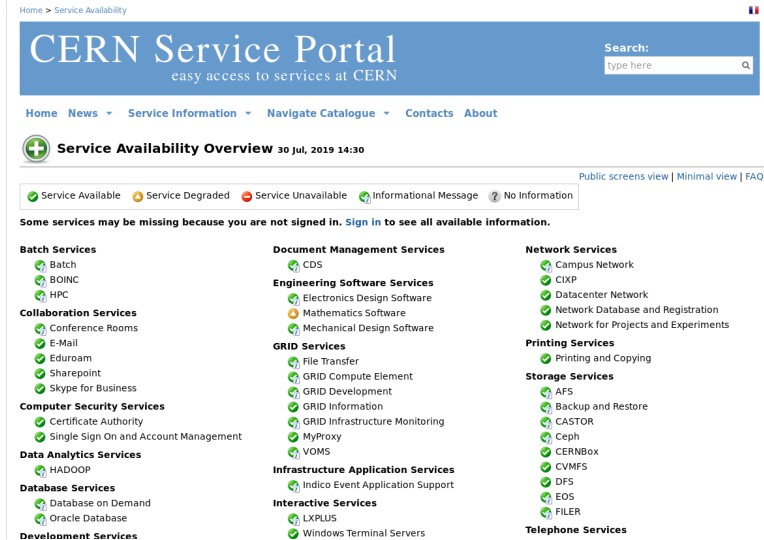


Figure 1: Service availability overview

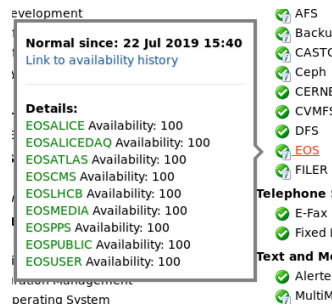


Figure 2: EOS sub-services availability

- c. **Sub-services and meta-services** : If a given service is composed of different parts, and users might be interested in seeing details and availabilities of these parts, its service manager can declare these parts as sub-services. Sub-services are displayed in SLS in the same way as regular services.
- d. **Dependencies** : Each service can declare which other services it uses and depends on. For example, the CVS service at CERN uses e-mail service for notifying users about commits, but if these e-mail messages are not delivered, core functionality of the CVS service is not affected. A CASTOR (CERN hierarchical mass storage system) instance, on the other hand, depends on underlying Oracle databases, and will not work if these databases are not available.
- e. **Views** : A view is a set of top-level services and meta-services that is displayed on the SLS entry page. Views are freely configurable, and an SLS instance may have multiple independent views set up. The default view in the CERN instance of SLS groups services by functionality.

2.2 EOS

EOS is a multi-protocol disk-only storage system developed at CERN since 2010 to store physics analysis data for LHC and non-LHC experiments. The system became operational in 2011 for the ATLAS experiment with 2 PB in size. The total storage space today is segmented into six independent failure domains (instances) for the four LHC experiments ALICE, ATLAS, CMS and LHCb, a shared experiment instance for non-LHC experiments (PUBLIC) and a generic user instance USER for all CERN users. (Peters, Sindrilaru, & Adde, 2015)

2.3 Probing

Probing is the process of acquiring control or data from a telecommunication or data network without disturbing the network being monitored. This ensures that applications using the probed data can do so without any threat to the operation of the observed network. This is particularly important for network management/OSS applications where disturbing the observed network with active devices can itself affect network operation and destroy the value of the probed data. (Wikipedia, 2019a) The most common word used to talk about this concept is "pinging", referring to "ping" command usually used as a simple way to verify that a computer can communicate over the network with another computer or network device.

3 Architecture & design

3.1 Service Level Status (SLS)

As already mentioned, SLS does not monitor services – it expects service managers to provide availability information for their services. Service managers often use existing monitoring systems for calculating or estimating availability and status of their services.

In my case, I was working on the data source part for EOS service.

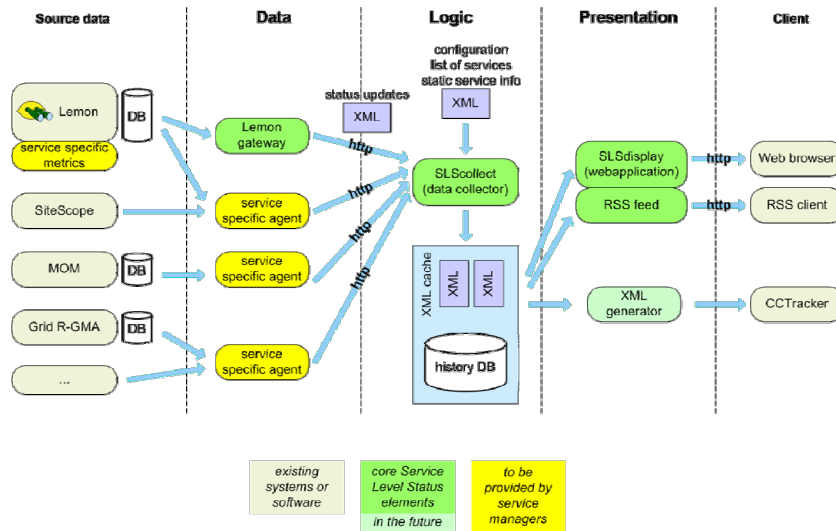


Figure 3: SLS architecture

3.2 EOS

EOS separates the IO path into meta data access via a meta data service MGM and data access via file IO services FST. To guarantee minimal file access latencies all meta data is kept in-memory on meta data server nodes and persisted using WAL technology, but also,

more recently, on a highly available datastore : QuarkDB. Files are in most cases replicated on two JBOD disks. EOS supports additionally erasure-encoding of files with two or three redundancy stripes. The current cluster state and configuration is represented by a so-called shared hash. The state and configuration modifications are exchanged between storage and meta data servers using a message queue service MQ. All three services MGM, FST and MQ have been implemented using the XRootD client-server framework.

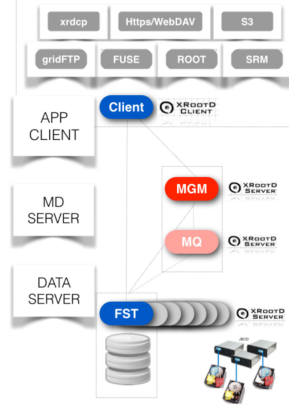


Figure 4: EOS architecture

3.3 EOS protocols

1. **XRootD** : The XROOTD project aims at giving high performance, scalable fault tolerant access to data repositories of many kinds. The typical usage is to give access to file-based ones. It is based on a scalable architecture, a communication protocol, and a set of plugins and tools based on those. The freedom to configure it and to make it scale (for size and performance) allows the deployment of data access clusters of virtually any size, which can include sophisticated features, like authentication/authorization, integrations with other systems, WAN data distribution, etc.

XRootD software framework is a fully generic suite for fast, low latency and scalable data access, which can serve natively any kind of data, organized as a hierarchical filesystem-like namespace, based on the concept of directory. As a general rule, particular emphasis has been put in the quality of the core software parts. (*XrootD.org*, 2019)

2. **GridFTP** : GridFTP is an extension of the File Transfer Protocol (FTP) for grid computing. The protocol was defined within the GridFTP working group of the Open Grid Forum. There are multiple implementations of the protocol; the most widely used is that provided by the Globus Toolkit. The aim of GridFTP is to provide a more reliable and high performance file transfer, for example to enable the transmission of very large files. GridFTP is used extensively within large science projects such as the Large Hadron Collider and by many supercomputer centers and other scientific facilities.

GridFTP provides a uniform way of accessing the data, encompassing functions from all the different modes of access, building on and extending the universally accepted FTP standard. FTP was chosen as a basis for it because of its widespread use, and because it has a well defined architecture for extensions to the protocol (which may be dynamically discovered). (*Wikipedia*, 2019b)

3. **SRM** : The SRM ("Storage Resource Manager") is a protocol for Storage Resource Managment. The SRM protocol does not do any data transfer. The protocol is used to ask a Mass Storage System (MSS) to make a file ready for transfer, or to create space in

a disk cache to which a file can be uploaded. The file is then transferred to or from a Transfer URL or TURL. (*GridPP*, 2019)

4. **HTTP** : HTTP is a protocol used for any data exchange on the Web and it is a client-server protocol, which means requests are initiated by the recipient. Clients and servers communicate by exchanging individual messages (as opposed to a stream of data). The messages sent by the client, are called requests and the messages sent by the server as an answer are called responses. (*MDN web docs*, 2019)

4 Contribution

To be short, the main goal of the project is to try to improve the current system of probing the different EOS services to display their availability in SLS. The current scripts are in Bash, and the execution of the tests is sequential and takes a considerable time, and therefore limits the number of services to manage (because there's a refreshing timeout to respect). In other words, the project consists of rewriting the scripts in Python and optimize it : parallelize the execution and make it as extensible as possible.

The different tasks that I had to do and the different things I learned so far were :

- Understanding the current architecture and drawing my own scheme based on my interpretation.

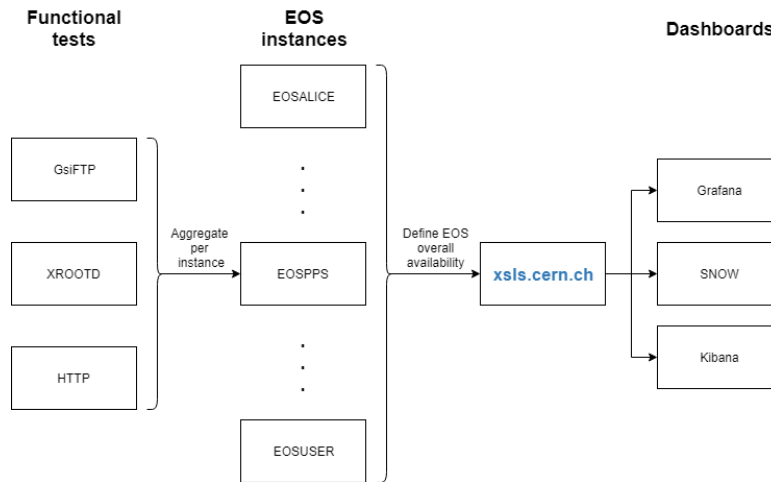


Figure 5: EOS architecture based on my interpretations

- First time dealing with CentOS and RedHat based distros in general.
- Getting a deeper view of Bash, and not just the old *echo* “Hello world” commands.
- Setting up a **puppet** managed VM and discovering the YAML and puppet syntax.
- Dealing with different authentication protocols (X509, krb5)
- Migrating the current scripts in BASH to a Python2 version.
- Writing functional tests for GridFTP, HTTP and slightly adapting the XROOTD with python syntax.
- Building RPM package for the project.
- Parametrizing the script in a configuration file.

5 Results

Due to short time of the staying period, we didn't have the time to test the new scripts on a large scale, and we only tested it on one EOS instance : eospps, which is an instance dedicated to tests. The output of a command is something like :

```
root@eosslsprobe003 ~/git/eos-sls-probe-py (master) $ python eos-sls-probe.py --instance=eospps --protocol=xrd --debug
got X509 credentials
DEBUG: will test XROOTD under 'root://eospps//eos/ppsscratch/test/slstest-eospps/xrd-eospps-f7c34b76-c423-11e9-b7f0-fa163e9d8a9e'
DEBUG: (out:, err:, elapsed:-1)
DEBUG: will run 'xrdcp -DlConnectTimeout 30 -DlTransactionTimeout 60 -DlRequestTimeout 60 -DlRedirCntTimeout 60 -d l -v -f -DlReadCacheSize
0 /tmp/xrd-eospps-f7c34b76-c423-11e9-b7f0-fa163e9d8a9e.in root://eospps//eos/ppsscratch/test/slstest-eospps/xrd-eospps-f7c34b76-c423-11e9-b7
f0-fa163e9d8a9e'
DEBUG: will run 'xrdfs root://eospps stat /eos/ppsscratch/test/slstest-eospps/xrd-eospps-f7c34b76-c423-11e9-b7f0-fa163e9d8a9e'
DEBUG: will run 'xrdcp -DlConnectTimeout 30 -DlTransactionTimeout 60 -DlRequestTimeout 60 -DlRedirCntTimeout 60 -d l -v -f root://eospps//eo
s/ppsscratch/test/slstest-eospps/xrd-eospps-f7c34b76-c423-11e9-b7f0-fa163e9d8a9e /tmp/xrd-eospps-f7c34b76-c423-11e9-b7f0-fa163e9d8a9e.out'
DEBUG: will run 'xrdfs root://eospps rm /eos/ppsscratch/test/slstest-eospps/xrd-eospps-f7c34b76-c423-11e9-b7f0-fa163e9d8a9e'
DEBUG: getting node addr : (family:2, socktype:1, proto:6, canonname:, sockaddr:('137.138.62.124', 0))
DEBUG: getting host by addr : (hostname:eospps-fe1.cern.ch, aliaslist:[], ipaddrlist:['137.138.62.124'])
DEBUG: will check instance 'eospps [eospps-fe1.cern.ch]'
DEBUG: successfully pushed data for id eospps to XSLS
Overall result available : Functional test of eospps via xrootd was OK, testfile=/eos/ppsscratch/test/slstest-eospps/xrd-eospps-f7c34b76-c42
3-11e9-b7f0-fa163e9d8a9e
Tests used headnode eospps-fe1.cern.ch
```

Figure 6: Example of the output of testing the XRootD protocol

6 Conclusion

The primary goal of the project was to migrate the current system to python, which has been done. And then, add the default config files to automate the process without specifying the protocols to probe. And to conclude, this project will soon be deployed in production and extended to support the rest of instances, and even new ones.

References

- Eos.* (2019). Retrieved 2019-08-21, from <https://eos.web.cern.ch/>
- Gridpp.* (2019). Retrieved 2019-08-21, from <https://www.gridpp.ac.uk/wiki/SRM>
- Lopienski, S. (2008). Service level status - a new real-time status display for it services. *Journal of Physics: Conference Series 119 (2008) 052025*.
- Mdn web docs.* (2019). Retrieved 2019-08-21, from <https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview>
- Peters, A. J., Sindrilaru, E., & Adde, G. (2015). Eos as the present and future solution for data storage at cern.
- Wikipedia.* (2019a). Retrieved 2019-08-21, from https://en.wikipedia.org/wiki/Passive_probing
- Wikipedia.* (2019b). Retrieved 2019-08-21, from <https://en.wikipedia.org/wiki/GridFTP>
- Xrootd.org.* (2019). Retrieved 2019-08-21, from <http://xrootd.org/>