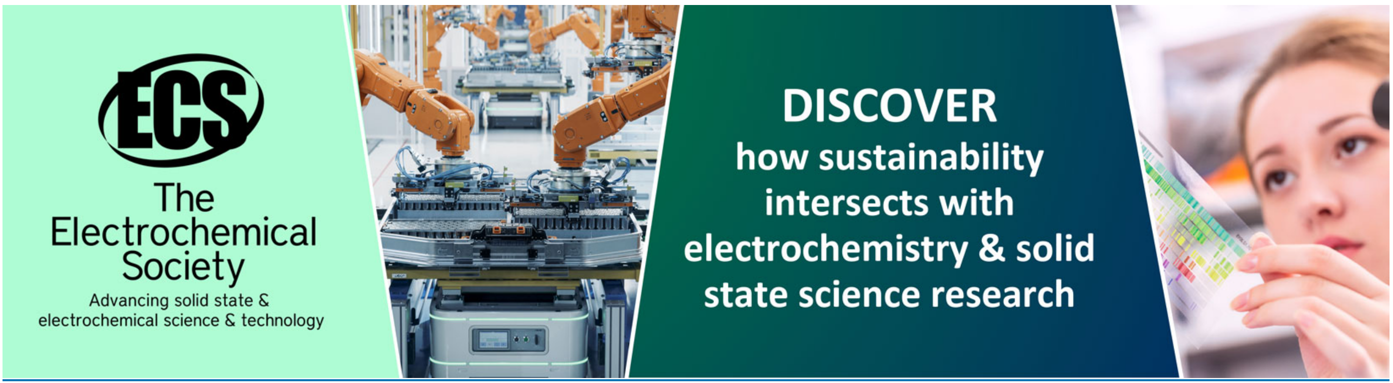
Service monitoring in the LHC experiments


ECS
The Electrochemical Society
Advancing solid state & electrochemical science & technology
DISCOVER
how sustainability intersects with electrochemistry & solid state science research

Service monitoring in the LHC experiments

Fernando Barreiro Megino[1], Vincent Bernardoff[2], Diego da Silva Gomes[3], Alessandro di Girolamo[1], José Flix[4], Peter Kreuzer[5], Stefan Roiser[1]

[1]CERN IT Experiment Support, CH-1211 Geneva 23

[2]Univ. Pierre et Marie Curie (Paris VI)

[3]Universidade do Estado do Rio de Janeiro

[4]Port d'Informació Científica & Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas (CIEMAT)

[5]Rheinisch-Westfaelische Technische Hochschule

Abstract. The LHC experiments' computing infrastructure is hosted in a distributed way across different computing centers in the Worldwide LHC Computing Grid (WLCG [1]) and needs to run with high reliability. It is therefore crucial to offer a unified view to shifters, who generally are not experts in the services, and give them the ability to follow the status of resources and the health of critical systems in order to alert the experts whenever a system becomes unavailable. Several experiments have chosen to build their service monitoring on top of the flexible Service Level Status (SLS) framework developed by CERN IT. Based on examples from ATLAS, CMS and LHCb, this contribution will describe the complete development process of a service-monitoring instance and explain the deployment models that can be adopted.

1. SLS overview

1.1. Introduction to SLS

LHC experiments and IT departments provide computing services of an increasingly heterogeneous nature. There is thus a growing need for a status display that groups these different services and reports status and availability in a uniform way, enabling shifters and service managers to follow up the health of the different services. The Service Level Status (SLS) system [2] is the framework developed within the CERN IT department that addresses these needs by providing a web-based display that dynamically shows availability. It helps shifters to monitor the basic information, follow the statistics about various services, as well as the dependencies between them, and it provides an integrated, customizable alert system to send out notifications to service managers in case a service becomes unavailable.

1.2. SLS architecture

Figure 1 shows a simplified architecture of the SLS system: it is the service managers who have to implement a *service specific agent* that collects the information by querying different *source data* such as e.g. Lemon [3] or service internal metrics and then publishes it to the *SLS data collector*. SLS collects, stores and displays information (*Logic* and *Presentation* columns) and makes it available for different *client applications* as the web browser or RSS clients.

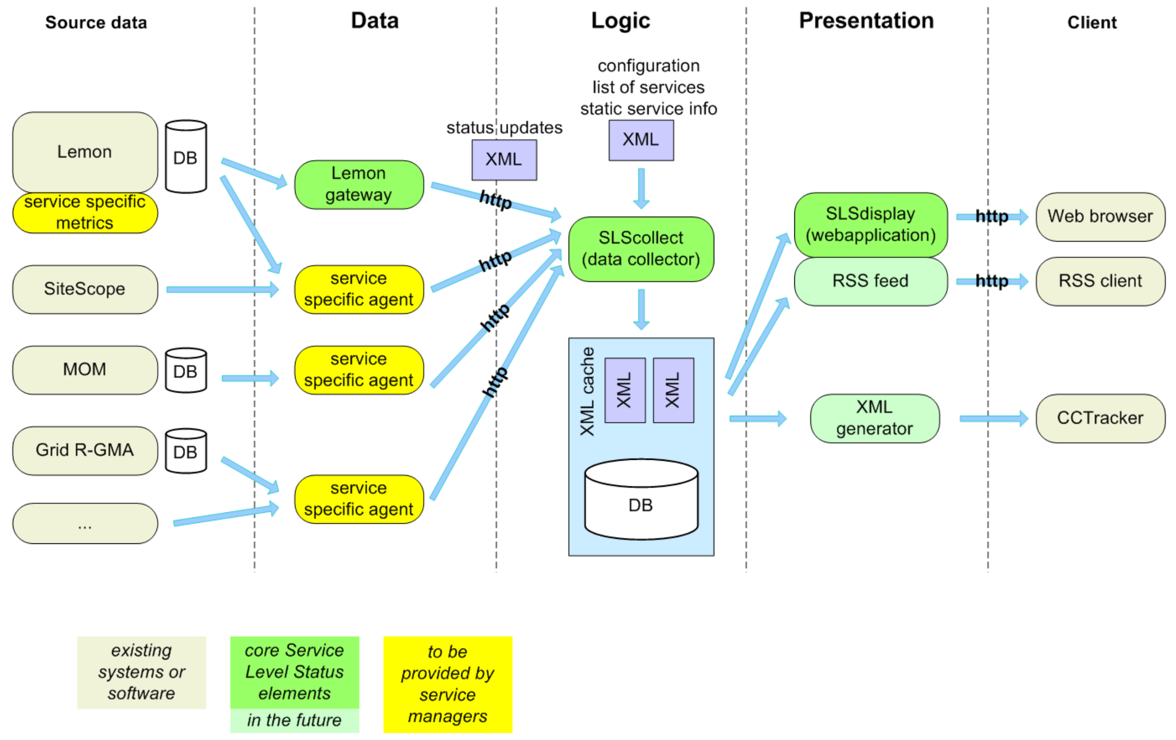


Figure 1 - Service Level Status architecture [2]

2. Implementing the service specific agents

As mentioned before, it is the service managers who need to develop the agents that publish the information to SLS in order to populate the database. As very first step, the service manager should be aware about which are the metrics that characterize the health of his service and which metrics are interesting to display additionally in SLS. From our personal experience, useful metrics to define the health of a service is the number of restarts, errors appearing in the own log file or in the log files of underlying services e.g. web server and database errors.
A very useful toolkit to obtain the metrics of a service is the Lemon toolkit [4], which provides sensors for parsing log files, performance measurements of the machine hosting the service (e.g. CPU and memory usage), etc. We use this toolkit continuously to identify and count regular expressions in the logs. Sometimes it is also useful to instruct the own services to write out periodically a report file with the latest activity.
With the acquired information we have to calculate the most important value: the availability. Combining the metrics and defining the availability is entirely the responsibility of the service specific agent and the algorithm is completely free. One could define the availability as the number of successes divided by the total number of attempts for a certain action, a weighted combination of metrics or simply assign a discontinuous function where the values depend on different parameters like in the following example from ATLAS Distributed Data Management:

$$availability = \begin{array}{l} 100\%\ if\ \#CRITICAL\ errors < 3 \\ 50\%\ if\ \#CRITICAL\ errors \geq 3 \\ 30\%\ if\ report\ older\ than\ 33\ min \\ 0\%\ if\ \#restarts\ in\ last\ 30\ min\ \geq 3 \end{array}$$

Additionally one has to define three availability thresholds (t1 < t2 < t3), which will characterize the status of the service as one of the below:

- **fully available** if *availability > t3*
- **affected** if *t2 < availability < t3*
- **degraded** if *t1 < availability < t2*
- **not available** if *availability < t1*

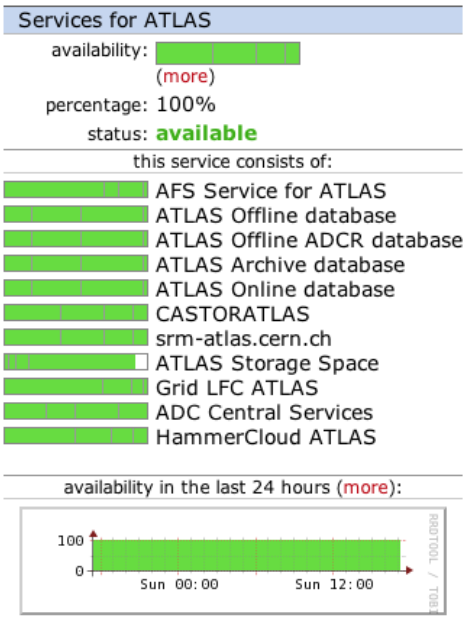
All the gathered information has to be combined in a XML availability update file, following a simple structure and this file has to be made available to SLS on a HTTP server. In the next sections we will see different examples and deployment models.

3. Experiment usage

ATLAS, CMS and LHCb have been actively using SLS for years to monitor the most critical services. The following sections will give an insight in how each experiment uses it.

3.1. ATLAS

As seen in Figure 2, the ATLAS Distributed Computing (ADC) community uses SLS [4] to monitor critical services provided by CERN IT (e.g. database clusters, central file catalog etc.), the availability of grid storage resources and its own centrally hosted and managed services – ADC Central Services. The ADC Central Services category includes the Distributed Data Management (DDM) services [5], the workload management system PanDA [6], the ATLAS Metadata Interface (AMI) [7] and the Frontier servers. When one of these services becomes unavailable the service administrators are alerted by mail directly. In addition shifters are instructed to check the status of this category periodically and get in contact with the expert on call to follow up the situation.



Services for ATLAS

availability: 🟩 (more)

percentage: 100%

status: available

this service consists of:

🟩 AFS Service for ATLAS
🟩 ATLAS Offline database
🟩 ATLAS Offline ADCR database
🟩 ATLAS Archive database
🟩 ATLAS Online database
🟩 CASTORATLAS
🟩 srm-atlas.cern.ch
🟩 ATLAS Storage Space
🟩 Grid LFC ATLAS
🟩 ADC Central Services
🟩 HammerCloud ATLAS

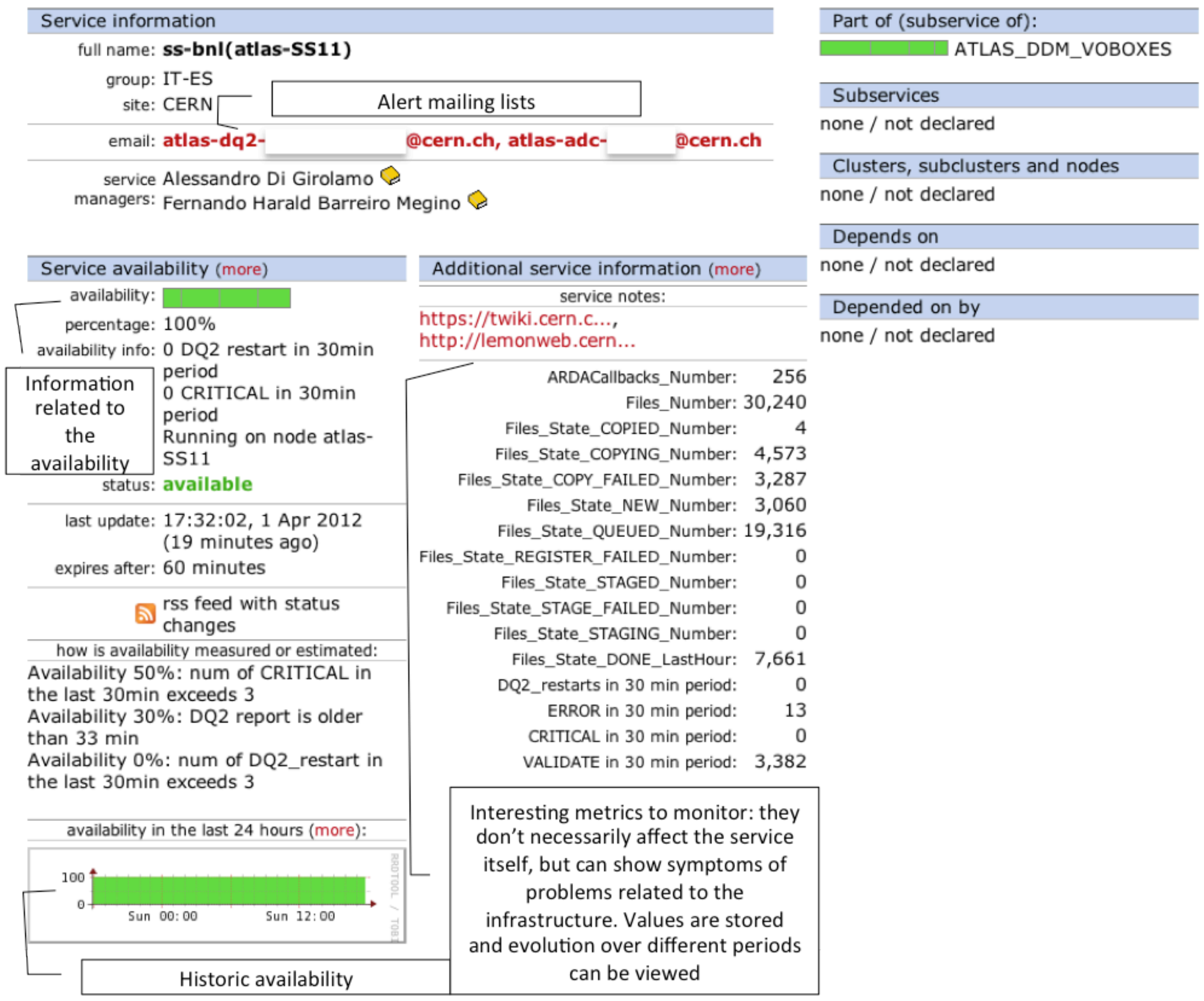
availability in the last 24 hours (more):

Figure 2 - Services monitored by ATLAS Distributed Computing

One of the most complete examples for service monitoring in the Central Services category can be seen in Figure 3 that shows a screenshot of the SLS monitoring for the DDM file transfer agents. The displayed information is gathered from two sources:

1. **Log file:** Particular events are counted in the log file with the help of Lemon sensors. This is used to retrieve the amount of CRITICAL and ERROR messages, as well as restarts of the service.
2. **Service reports**: The SLS agent also parses service reports that are written out periodically, in order to provide information about the number of files being copied, failed file transfers or queued files.

The SLS agent merges the information from the sources, calculates the service availability based on the restarts of the agents and the number of CRITICAL messages in the logs and publishes all the information to SLS.



Service information
full name: **ss-bnl(atlas-SS11)**
group: IT-ES
site: CERN
email: **atlas-dq2-[redacted]@cern.ch, atlas-adc-[redacted]@cern.ch**
service managers: Alessandro Di Girolamo, Fernando Harald Barreiro Megino

Alert mailing lists

Part of (subservice of):
🟩 ATLAS_DDM_VOBOXES

Subservices
none / not declared

Clusters, subclusters and nodes
none / not declared

Depends on
none / not declared

Depended on by
none / not declared

Service availability (more)
availability: 🟩
percentage: 100%
availability info: 0 DQ2 restart in 30min period
0 CRITICAL in 30min period
Running on node atlas-SS11
status: **available**

Information related to the availability

last update: 17:32:02, 1 Apr 2012 (19 minutes ago)
expires after: 60 minutes

rss feed with status changes

how is availability measured or estimated:
Availability 50%: num of CRITICAL in the last 30min exceeds 3
Availability 30%: DQ2 report is older than 33 min
Availability 0%: num of DQ2_restart in the last 30min exceeds 3

availability in the last 24 hours (more):

Historic availability

Additional service information (more)
service notes:
https://twiki.cern.c...,
http://lemonweb.cern...

ARDACallbacks_Number:	256
Files_Number:	30,240
Files_State_COPIED_Number:	4
Files_State_COPYING_Number:	4,573
Files_State_COPY_FAILED_Number:	3,287
Files_State_NEW_Number:	3,060
Files_State_QUEUED_Number:	19,316
Files_State_REGISTER_FAILED_Number:	0
Files_State_STAGED_Number:	0
Files_State_STAGE_FAILED_Number:	0
Files_State_STAGING_Number:	0
Files_State_DONE_LastHour:	7,661
DQ2_restarts in 30 min period:	0
ERROR in 30 min period:	13
CRITICAL in 30 min period:	0
VALIDATE in 30 min period:	3,382

Interesting metrics to monitor: they don't necessarily affect the service itself, but can show symptoms of problems related to the infrastructure. Values are stored and evolution over different periods can be viewed

Figure 3 - Example of information collected for ATLAS DDM Site Services

We need to provide a web URL to SLS, where it can retrieve the XML availability update file. However not all the machines hosting ADC Central Services need to run a web server and for security reasons ADC is not interested in installing *httpd* on all machines. Therefore different deployment models for the SLS monitoring have originated (see Figure 4). Talking plainly, DDM Central Catalog machines can be considered the web frontends to an Oracle database and therefore need to run httpd and as a consequence can publish the XML availability update file directly to the SLS data collector.

On the contrary most of the other DDM services do not run web servers, since they are agents that are running and performing certain actions. Here the XML health reports need to be sent through the MSG messaging service**Error! Reference source not found.** and a lightweight server running on a minimal Virtual Machine does the publishing to the SLS data collector.

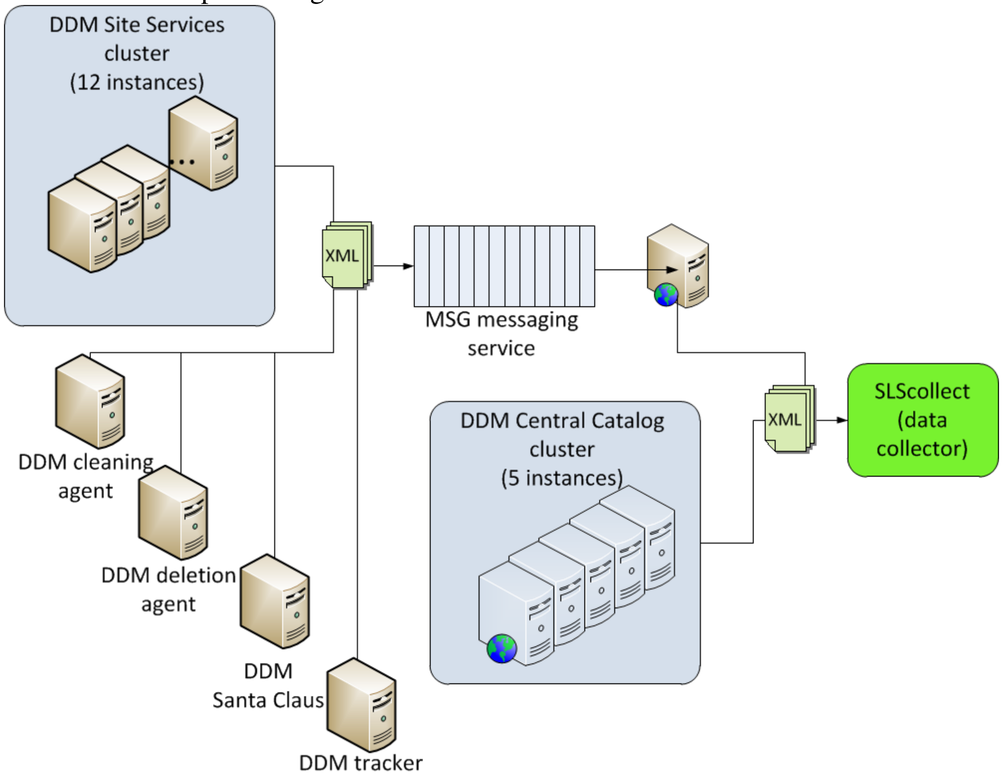

DDM Site Services cluster (12 instances)
XML
MSG messaging service
XML
SLScollect (data collector)
DDM Central Catalog cluster (5 instances)
DDM cleaning agent
DDM deletion agent
DDM Santa Claus
DDM tracker

Figure 4 – ATLAS Distributed Data Management service monitoring infrastructure

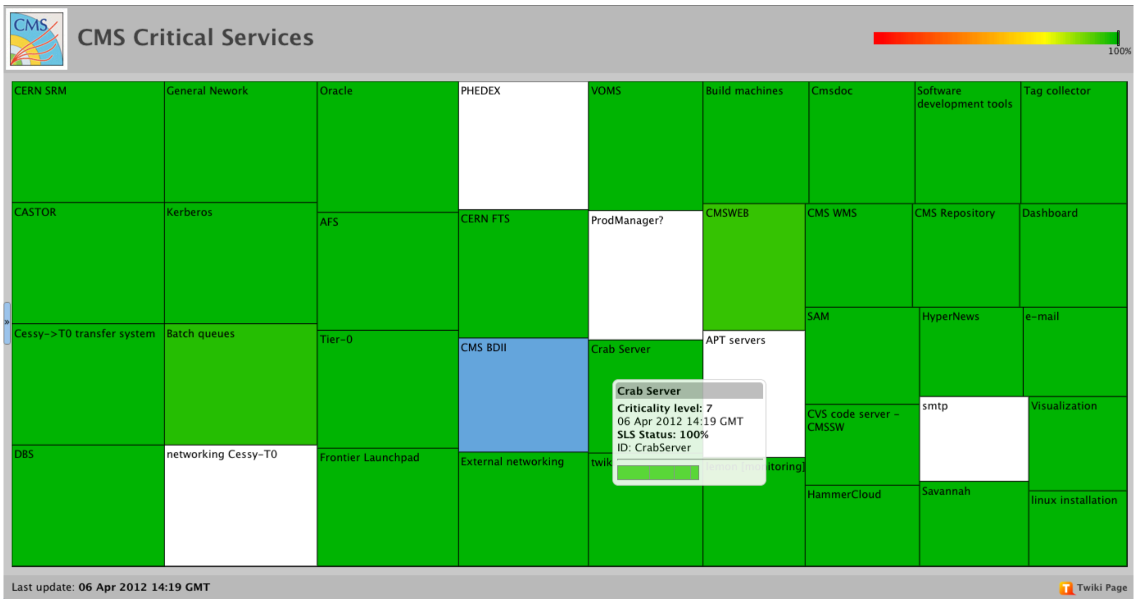
3.2. CMS

The CMS experiment does an extensive usage of SLS for a variety of general services. CMS characterizes the usage of SLS in the following ways:

i. SLS is being used to monitor the quality of a service by instructing agents that play directly the role of a user. The most common use case is to evaluate that web services are alive: not working URLs are considered unavailable. The CMS document and software development tool frontends are monitored in this way. In more complex scenarios the agents try to open URLs from several servers, evaluate the returned content (e.g. by looking for certain patterns or verifying information in the output) and combine the information. This method is applied to the CMS workload management frontend cluster CRAB **Error! Reference source not found.**[9]. Shifters should report any unavailable service and ask the experts to take action.
ii. SLS just translates what other monitoring systems are showing. This is the case for all the CMSWEB services [10]. This is achieved by an agent that queries Lemon and generates the SLS feed for each subservice. This use case is considered merely informative, but will not trigger alarms, since alarms to operators are already raised by Lemon.
In both previous cases i and ii, the agent that outputs the SLS feed is able to calculate the service availability remotely and only requires access to the AFS [11] web area where the status reports are published to the SLS collector.

iii. Finally SLS is used in a similar way to the ATLAS DDM use case: the SLS agent runs on the machine that hosts a particular service so that it has access to the same environment, logs and state. In this way the agent can check file contents, parse error messages in the log files, count number of threads and processes, and verify that there is no backlog in queues. This is applied for instance to the CMS Tier0 Production [12].

Eventually, all critical CMS SLS entries are included in the gridmap view, offering the overall picture of how the services are behaving.



Crab Server
Criticality level: 7
06 Apr 2012 14:19 GMT
SLS Status: 100%
ID: CrabServer

Last update: **06 Apr 2012 14:19 GMT**

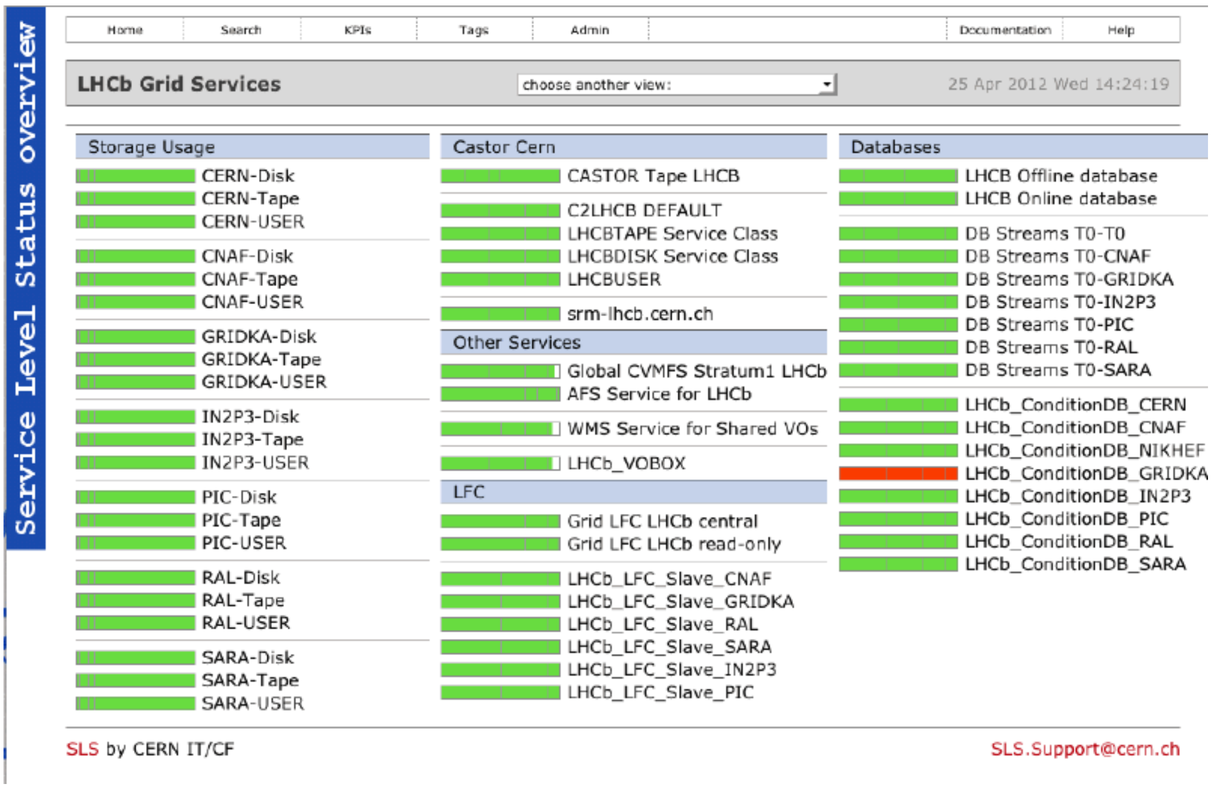
Figure 5 - CMS Critical Services monitoring system [13] using *treemap* visualization technique (author: Lukasz Kokoszkiewicz)

3.3. LHCb

LHCb uses SLS monitoring not only for CERN based, but also remote services. The sensors are either provided by the experiment itself or integrated from other groups (mostly from CERN IT). The experiment groups the sensors into two main categories:

i. The first group provides a collection of sensors related to LHCb's grid services that are hosted either at CERN or at Tier1 sites. Examples are the monitoring of the Tier1 capacities and thresholds for storage usage (e.g. disk and tape), the Local File Catalog (LFC) [14] instances and the online and offline databases and their streaming to Tier1 sites. An overview page (see Figure 6) displaying the most important sensors in a single page has been set up and is being used by the LHCb shifters on grid operations to spot problems with services. Furthermore alarms about these services are being sent to the current shifter through a dynamic mailing list (CERN e-group), where the mailing list is only populated with the email address of the currently working shifter. The sensors for the grid related SLS sensors are being executed within the LHCb/Dirac framework [15]- the experiment's tool for handling grid activities -, which is also publishing the standard XML files to SLS.

ii. The second group currently provides sensors to monitor services used by the experiment's physicists and developers. The service providers from CERN IT mostly implement the sensors and LHCb groups the relevant ones in a dedicated SLS view. Examples of services to monitor are the interactive batch system, code repositories, twiki servers, tape storage, mail servers and the agenda tool. In the future this group shall be split into two separate groups providing information needed by LHCb physicists (e.g. batch system and twiki) and another group for developers (e.g. code repositories and nightly builds).


Service Level Status overview

Home | Search | KPIs | Tags | Admin | Documentation | Help

LHCb Grid Services choose another view: 25 Apr 2012 Wed 14:24:19

Storage Usage	Castor Cern	Databases
🟩 CERN-Disk	🟩 CASTOR Tape LHCB	🟩 LHCB Offline database
🟩 CERN-Tape	🟩 C2LHCB DEFAULT	🟩 LHCB Online database
🟩 CERN-USER	🟩 LHCBTAPE Service Class	🟩 DB Streams T0-T0
🟩 CNAF-Disk	🟩 LHCBDISK Service Class	🟩 DB Streams T0-CNAF
🟩 CNAF-Tape	🟩 LHCBUSER	🟩 DB Streams T0-GRIDKA
🟩 CNAF-USER	🟩 srm-lhcb.cern.ch	🟩 DB Streams T0-IN2P3
🟩 GRIDKA-Disk	**Other Services**	🟩 DB Streams T0-PIC
🟩 GRIDKA-Tape	🟩 Global CVMFS Stratum1 LHCb	🟩 DB Streams T0-RAL
🟩 GRIDKA-USER	🟩 AFS Service for LHCb	🟩 DB Streams T0-SARA
🟩 IN2P3-Disk	🟩 WMS Service for Shared VOs	🟩 LHCb_ConditionDB_CERN
🟩 IN2P3-Tape	🟩 LHCb_VOBOX	🟩 LHCb_ConditionDB_CNAF
🟩 IN2P3-USER	**LFC**	🟩 LHCb_ConditionDB_NIKHEF
🟩 PIC-Disk	🟩 Grid LFC LHCb central	🟥 LHCb_ConditionDB_GRIDKA
🟩 PIC-Tape	🟩 Grid LFC LHCb read-only	🟩 LHCb_ConditionDB_IN2P3
🟩 PIC-USER	🟩 LHCb_LFC_Slave_CNAF	🟩 LHCb_ConditionDB_PIC
🟩 RAL-Disk	🟩 LHCb_LFC_Slave_GRIDKA	🟩 LHCb_ConditionDB_RAL
🟩 RAL-Tape	🟩 LHCb_LFC_Slave_RAL	🟩 LHCb_ConditionDB_SARA
🟩 RAL-USER	🟩 LHCb_LFC_Slave_SARA	
🟩 SARA-Disk	🟩 LHCb_LFC_Slave_IN2P3	
🟩 SARA-Tape	🟩 LHCb_LFC_Slave_PIC	
🟩 SARA-USER		

SLS by CERN IT/CF SLS.Support@cern.ch

Figure 6 - LHCb grid service monitoring overview

4. Conclusions

Service monitoring is a common need for the different LHC experiments and other groups in the CERN IT department. SLS offers an organized, extensible and customizable single point of entry for all possible services. Therefore SLS helps shifters, experts and operators the ability of having an overview of the different systems at a single glance, without the need of consulting one page per service.

Three of the LHC experiments have embraced this system to monitor an immense variety of their services, sometimes adopting common solutions (e.g. ATLAS and LHCb storage space monitoring) and sometimes building their own infrastructure on top of the SLS monitoring (e.g. CMS gridmap view). We have also identified a wide choice of deployment models for the monitoring agents depending on the service - agents running on the same machine as the service and agents running remotely and pinging the service externally. To publish the report to SLS we have seen different options as well: agents publishing the reports locally on machines running a web server, agents publishing on an AFS web area and, when needed, agents sending out the report through a message queue and depending on a separate collector that publishes the report for them.

We can conclude that SLS is a useful tool to monitor a variety of services and information. The implementation of a new entry for a service is a simple process.

References

[1] Bird I et al 2005 *LHC computing Grid. Technical Design Report* (Geneva, CERN Press)
[2] Lopienski S, Service Level Status – a new real-time display for IT services, Proc. of the 16th Int. Conf. on Computing in High Energy and Nuclear Physics (CHEP2008)

[3] http://lemonweb.cern.ch/lemon-web/
[4] http://lemon.web.cern.ch/lemon/doc/sensors.shtml
[5] Stewart G et al, Advances in Service and Operations for ATLAS Data Management, Proc. of the Int. Conf. on Advanced Computing and Analysis Techniques in Physics Research (ACAT 2011)
[6] Maeno T, Overview of ATLAS PanDA Workload Management, Proc. of the 18th Int. Conf. on Computing in High Energy and Nuclear Physics (CHEP2010)
[7] Albrand S et al, The AMI Database Project: Atlas Data Challenge Bookkeeping, and the Tag Collector, a new tool for Release Management, Proc. of the 13th Int. Conf. on Computing in High Energy and Nuclear Physics (CHEP2003)
[8] Cons L, The WLCG Messaging Service and its Future, Proc. of the 20th Int. Conf. on Computing in High Energy and Nuclear Physics (CHEP2012)
[9] Spiga D et al, CRAB (CMS Remote Analysis Builder), Proc. of the 16th Int. Conf. on Computing in High Energy and Nuclear Physics (CHEP2007)
[10] Metson S et al, CMS Offline Web Tools, Proc. of the 16th Int. Conf. on Computing in High Energy and Nuclear Physics (CHEP2007)
[11] Howard JH, An Overview of the Andrew File System, Proc. of the USENIX Winter Technical Conference, 1988
[12] http://cmsprod.web.cern.ch/cmsprod/sls/
[13] http://cms-critical-services.cern.ch/
[14] Frohner A et al, Data Management in EGEE, Proc. of the 17th Int. Conf. on Computing in High Energy and Nuclear Physics (CHEP2009)
[15] Tsaregorodtsev A et al, DIRAC - The Distributed MC Production and Analysis for LHCb, Proc. of the 14th Int. Conf. on Computing in High Energy and Nuclear Physics (CHEP2004)