WLCG Monitoring Consolidation Project Project Plan (Second Phase)

Status: Draft

Version: 1.0

Date: 4/12/2013

Abstract

This document describes the activities and plans of the WLCG Monitoring Consolidation Project. The project has to perform a critical analysis of what is monitored, the technology used and the deployment and support model, and to propose and implement a technical solution that would offer similar quality of service with a reduced effort.

Working Group Members:

Pablo Saiz [a)], Pedro Andrade [a)], Julia Andreeva [a)], Marian Babik [a)], Alexandre Beche [a)], Latchezar Betev [a)1)], Lionel Cons [a)], David Crooks, Ivan Dzhunov [a)], Josep Flix [b)], Alessandra Forti [c)2)], Paloma Fuente [a)], Alessandro di Girolamo [a)2)], Maria Girone [a)], Costin Grigoras [a)1)], Edward Karavakis [a)], Michael Kenyon [a)], Maarten Litmaath [a)1)], Nicolo Magini [a)3)], Luca Magnoni [a)], Stefan Roiser [a)4)], Andrea Sciabà [a)3)], Jacobo Tarragon [a)], David Tuckett [a)], Robert Veznaver [a)]

a) CERN, Geneva, Switzerland
b) CIEMAT, Spain
c) University of Manchester, UK

1) ALICE
2) ATLAS
3) CMS
4) LHCb

Table of Contents

Executive Summary

This document describes the work of phase 1 (requirements gathering, evaluation and prototyping) and defines the project plan of the phase 2 (implementation and initial deployment) of the new WLCG Monitoring systems. The project phase 1 started in July 2013, and phase 2 will conclude in June 2014.

The goal of the project is to decrease the amount of resources needed to support all the WLCG Monitoring applications maintained by IT-SDC. There are multiple approaches to get that result. The main ones that have been used are:

- reduce the applications and scope while still providing the functionality needed
- adopt a modular design that will allow the easier combination of existing solutions, shared with other groups or projects
- reduce the instances where the same information is collected and stored.

The phase one consisted of:

- An assessment of the functionality and applications currently offered by IT-SDC-MI
- An extensive testing and evaluation of possible tools and solutions of what is used within the WLCG, in IT Department and the Experiments.

The results of these tests are available in Section 5.

This document highlights for each layer of the architecture the work done and the plan for the next six months. Without going into details, available in Section 9, the work in phase 2 will mostly consist in:

1. simplify the probes and reducing the need of the Nagios framework
2. combine all the data into a single Metrics Store
3. evaluate alternative DB technologies to Oracle,
4. enforce modular visualization solutions
5. move the applications and the reporting needed to use the new monitoring system

The last section of the document shows the detailed master plan for phase 2 and all the actions mentioned in the final table are going to be followed up weekly using a project tracking system.

1 Introduction

This document describes the activities and plans of the WLCG Monitoring Consolidation Project. The project has to perform a critical analysis of what is monitored, the technology used and the deployment and support model, and to propose and implement a technical solution that would offer similar quality of service with a reduced effort. More details can be found at the project twiki page [1].

1.1 Project Goals

The goals of the project are to:

1. Reduce the complexity of the system
2. Ensure simplified and more effective operations, support and service management
3. Encourage an efficient deployment strategy, with a common development process
4. Unify, where possible, the implementation of the monitoring components
5. Use, wherever possible, the same solutions as in the Agile Infrastructure Monitoring

The main objective of the project is to get to the point where the effort required for WLCG monitoring can be reduced to half of its initial level.

Wherever reasonable, the effort should be aligned with the activities of the Agile Infrastructure Monitoring team at CERN.

The WLCG Monitoring Consolidation group is composed of experiments' representatives, operations representatives, an Agile Infrastructure Monitoring representative, and the members of the IT-SDC-MI section, who are responsible for the development, maintenance and support of the WLCG Monitoring tools.

The project should take into consideration the needs of all the stakeholders, in particular by:

- Reviewing all the existing monitoring applications supported by IT for the WLCG.
- Gather requirements from experiments, regarding the remote testing of the distributed sites and services.
- Understand the needs of the site administrators, answering the famous question 'I like to see at a single display how my site is working for all LHC VOs' .
- Reviewing the test submission part, both for stress testing and for functional testing.

Once that is done, the project should:

1. Define an overall architecture of the system.
2. Define priorities for any required further development
3. Agree on a deployment and support model.
4. Use, where possible, a common framework and common implementations for all the components
5. Provide a common web UI to WLCG monitoring applications

1.2 Project Phases

The project has been divided in two phases:

1. July-October 2013. The group performed the analysis of the current status and agreed on a plan to follow during the second phase.
2. October 2013-July 2014. The focus will be on implementing and deploying the plan (this document) defined during the first stage.

2 Initial Situation

The working group started by taking an in depth look at the current monitoring applications supported by IT-SDC and their usage. A summary of this investigation can be found in 'Appendix A: Usage of monitoring applications'. The review identified a couple of applications that are not necessary to maintain anymore.

The next step, once the functionality needed was well defined and understood, was to identify the areas where applications could be combined, offering the same, or even better functionality, while reducing the support effort. In particular, the area of Site and Service Monitoring was identified as the one with the most potential for optimization. There were several applications (SAM, SSB and SUM), which offered overlapping functionality, and it was agreed that the monitoring infrastructure would benefit if they could be integrated.

The current schema of the monitoring applications on the Site and Services infrastructure can be seen in Figure 1.

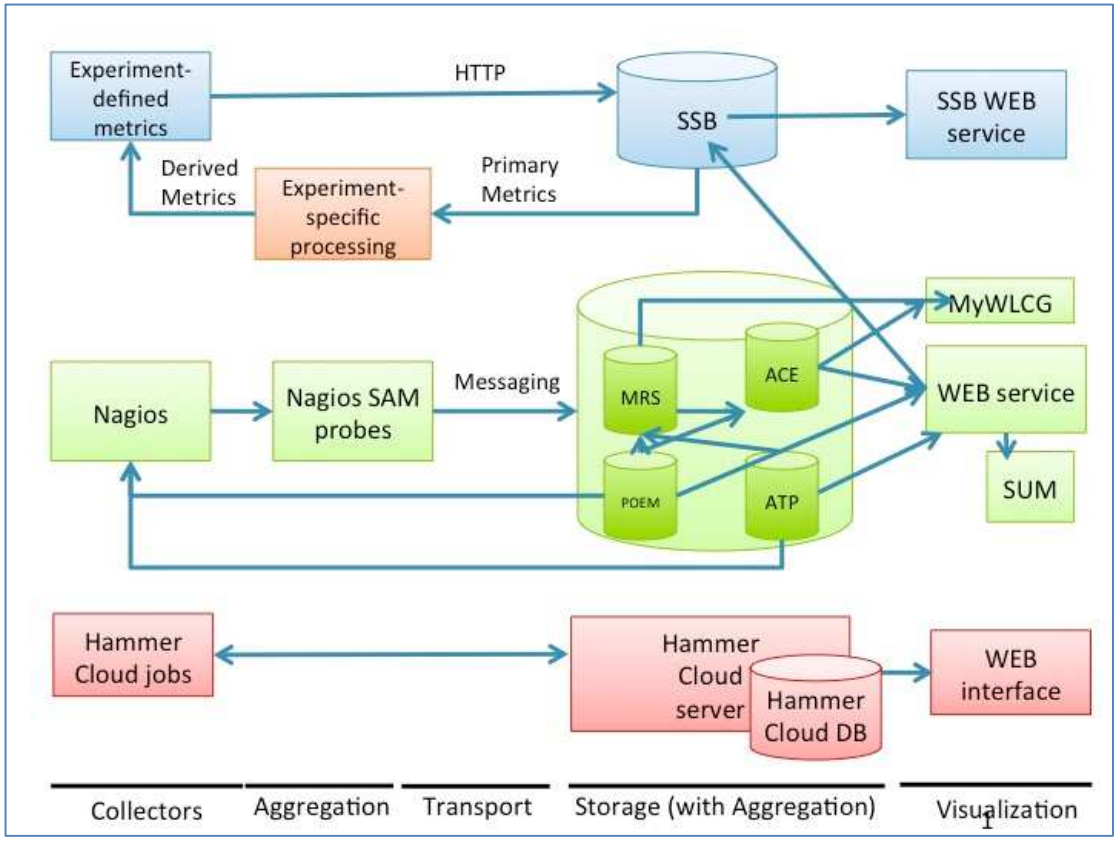


Figure 1. Architecture and design of the current monitoring systems. SBB,SAM/Nagios and HammerCloud. In addition several experiment specific systems are in use.

There are three main applications used for the Site and Services monitoring:

- SAM (Service Availability Monitoring). This application was originally conceived as a distributed testing and reporting infrastructure. It sends periodically tests to services to evaluate their behavior. Based on the results of those test, SAM creates monthly site availability and reliability reports. These reports are propagated to the WLCG and EGI communities.
- SSB (Site Status Board). The SSB provides a framework where users can publish their data, and the SSB stores its evolution over time. It was designed as a very flexible metric

store, allowing the definition of new metrics or instances to be monitored in a very dynamic way.

- HammerCloud (HC). HC is a Distributed Analysis testing system. It provides the submission framework for the remote tests, repository for the test results and user interface. The system can be used for functional testing and for stress testing.

The figure shows several limitations of the current system:

1. There are three independent applications, which offer overlapping functionality. They also have to handle similar issues, like evaluating the topology of the infrastructure.
2. Some applications offer different API for exporting data and for its native web interfaces.
3. The information from one of the application might be relevant to the others; currently, the only possible solution is to duplicate the data.
4. Several components talk directly to the databases of different components.
5. The deployment and support of each of the applications is done in a different way.

3 Proposed Architecture

The architecture proposed for the applications is depicted in Figure 2. It is worth pointing out that most of the current monitoring applications provided by the group already follow this approach, and the effort should be put into ensuring that all of them follow the same structure.

This layered design offers a great level of flexibility:

1. No dependencies between layers in terms of data formats and storage.
2. Components communicate only via APIs.
3. Different alternatives can be experimented for each layer without impact on the other layers.

For instance, it allows evaluating different visualization options, based on an existing storage layer. To allow this flexibility, it is very important that the layers communicate through well-defined APIs and data formats.

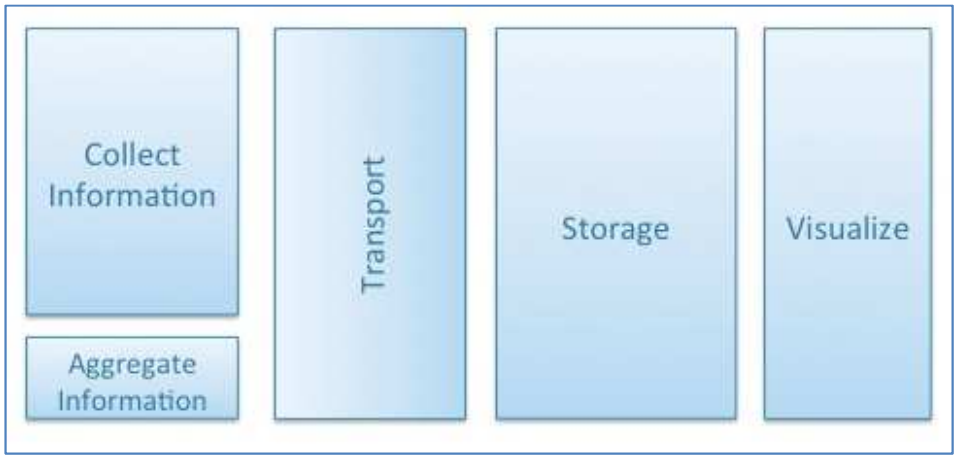


Figure 2. Components of the layered architecture

The next section describes some of the decisions taken. After that, the document will go through each of the layers depicted in Figure 2, identifying the areas that need more work. The structure of each chapter will be the same, first presenting the initial status and the changes done during the first phase of the project, and then introducing the solution agreed by the group, and finally the actions that have to be taken.

4 Layered Monitoring Architecture

With this layered design, the structure of the Service and Infrastructure monitoring application can look like the Figure 3.

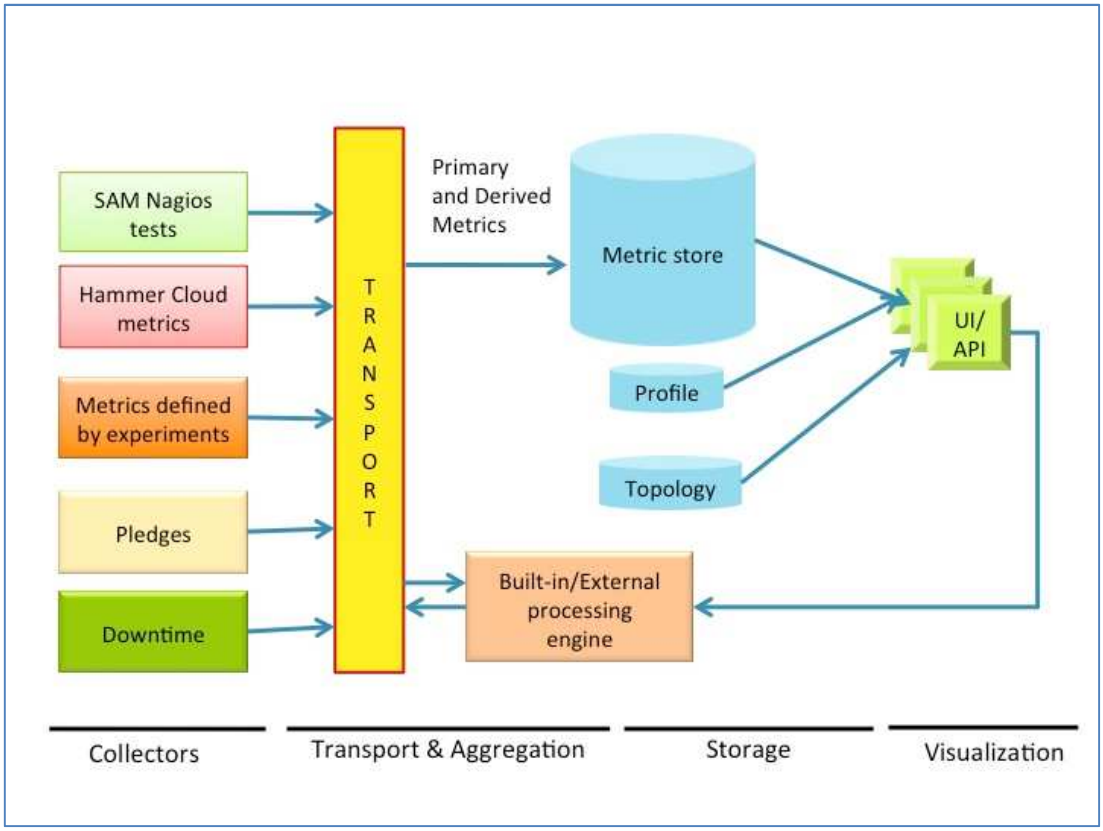


Figure 3. The layered design applied to the Monitoring Infrastructure

There are multiple producers of information on the left hand side. All of them publish the information to the transport in a predefined format. The information is then stored in a common metric store.

There are other databases containing the topology of the experiments and the profiles. The topology will have as input only the VOfeeds provided by the experiments.

The aggregation is taken out of the storage and the transport. IT-SDC will provide several algorithms to combine metrics and to register them again in the metric store. This algorithm will be used for the availability and reliability computation. On top of that, the experiments have the possibility of getting data out of the metric store (using the predefined API), apply their own algorithm, and feed the results back into the system.

5 General Considerations and Solutions Adopted

This section explains the main strategic decisions and their motivations.

5.1 Relationship with the Agile Infrastructure Monitoring

The AI monitoring goal [3] is to deliver new solutions for the operation of the computing centre (notifications, alarms, tickets, etc.) and for the analysis and visualization of monitoring data (dashboards, correlation engine, etc.). Whereas the actual information monitored by the applications considered by the WLCG Monitoring Consolidation and the AI monitoring is different, the approach should be, if possible, the same. The schema of an application that is described in the next chapter follows a very similar architecture than the one used by the AI monitoring team. The WLCG Consolidation Monitoring team evaluated the technologies considered by the AI Monitoring team and, where appropriate, it would adapt the same solutions.

Considerations on Message Format

The service and instance monitoring and the fabric monitoring have several things in common. First of all, they are monitoring the values of metrics over a set of instances (where the instances could be machines, services or even sites). Moreover, both use a transport layer to send the results. The format of the information that has to be sent can be described in a common way [4]. This would allow a closer interaction between the two groups, and an easier adoption of the technologies and ideas used by the two groups. This will also enable straightforward exchange of information between two monitoring domains.

Considerations on LogStash

The AI monitoring team evaluated LogStash as a tool to collect information from logs. After some tests, it decided not to integrate it for the time being. LogStash fits very nicely in the case where the information is already available in log files. Since the parsing of the text-based logs is not an important use case for the WLCG monitoring applications, LogStash technology has a limited value for the WLCG Monitoring.

Considerations on FLUME

Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data. It has a simple and flexible architecture based on streaming data flows. Flume is well suited for static configuration of information producers and consumers. In the case of WLCG Monitoring, the central monitoring tools provided by CERN IT are not the only information consumers. The LHC experiments might also consume information for various purposes, and the consumers can come and go. The information sources are even more dynamic. Therefore for some of the WLCG Monitoring applications, other transport layers seem to be a better choice. At the same time, and since it is not necessary to have a single transport method for all the use cases, there might be other use cases where FLUME can be adopted.

Considerations on ElasticSearch

The Agile Infrastructure Monitoring uses ElasticSearch as the main storage. The application seems to fit also some of the use cases of the WLCG Monitoring Consolidation, and therefore it is being evaluated as a possible alternative to ORACLE. This will be described in more detail in the following sections.

Considerations on HDFS

The AI Monitoring uses HDFS as the archival storage. This approach seems also possible for this working group. It will be evaluated during the second phase of the project.

Considerations on Kibana

Kibana is a web frontend on top of ElasticSearch. It is flexible, and it allows the creation of multiple graphs and tables. The initial setup of the tool is also very easy. The two major concerns that the group has about this particular tool are that it ties the system to ElasticSearch, and that the flexibility of the UI might not be enough for all the use cases.

5.2 Brief Assessment of Existing Solutions

Experiment Dashboard

The Experiment Dashboard [10] is a modular framework that allows the creation of monitoring applications. It provides python base classes for collectors of information, database objects and web UI. Each application can pick and choose which classes are needed, and then extend them according to its needs.

There are multiple applications built on top of the Experiment Dashboard Framework. The applications are independent, and they focus on four categories: job monitoring, data monitoring, site and services and finally outreach. The full list of applications can be found in 'Appendix A: Usage of monitoring applications'.

The experiment Dashboard project has been evolving over the last years. In particular, one of the latest changes has been the adoption of an MVC model built on top of external JavaScript libraries. It has to be noted that not all of the applications using the Experiment Dashboard are currently using this approach.

SAM

SAM [11] is a system that enables remote testing of distributed services and calculates the availability and reliability of services and sites based on the results of the remote tests. It gathers information from different systems, including information from GOCDB, BDII, OSG and VOfeeds. It was originally conceived as a distributed tool for testing and reporting. SAM has multiple components, described in the following list:

- Nagios framework for the execution of probes
- Metric Results Store (MRS)
- Aggregated Topology Provide (ATP)
- Profile Management (POEM)
- Availability Calculation Engine (ACE)
- MyWLCG (and MyEGI) web portal built on top of Django

The current version of SAM was developed for EGEE and EGI. The EGI's concept of NGIs and local control led to a high complexity, and more functionality than what is strictly necessary for the WLCG. In particular, most of the components had been prepared for two deployment scenarios: a distributed one and a centralized. Limiting to a centralized deployment simplifies the system.

Considerations on HammerCloud

HammerCloud is another system that provides remote testing of the distributed infrastructure. It can be used in two modes:

- *Functional testing*: test configured by experts. These tests frequently submit a few "ping" jobs. They are usually configured once, and then run from that moment onwards, giving a basic site validation.
- *Stress testing*: test configured by any user using a template system, which submits a large number of jobs concurrently during a slot of time. They run on-demand, and they are used to help commissioning new sites, evaluating changes, comparing site performances and checking site evolution.

HammerCloud (HC) should be aligned with the development, deployment and support model used for the other monitoring tools. The development should be aligned, in terms of version control tools, build systems and version numbering. The deployment and support will be described later on.

It should also be evaluated whether the HC schema could be merged with the Dashboard Job Monitoring. That would reduce the number of different schemas that have to be supported and it would facilitate data mining of the job monitoring data including results HC tests

Considerations On REBUS

REBUS [12] is the WLCG Resource, Balance & Usage data repository. It provides the official WLCG topology of federations and sites, which is a result of the signed MoUs. It provides as well the pledges and the monthly cpu and storage resources provided by the different sites to the experiments. REBUS is used to crosscheck that the usage of the site matches the pledges. The information that REBUS provides is very useful for other monitoring applications.

5.3 Selection of Storage Technology

ORACLE and Tests of PostgreSQL as Storage Solution

At the moment, all the monitoring applications based at CERN use ORACLE as a storage backend. This project should also evaluate the existing open source technologies for the WLCG monitoring use cases.

Evaluation of PostgreSQL

The Database on demand group in IT provides now other SQL back ends, including MySQL and PostgreSQL. The feasibility and the amount of work that it would take to move to one of them should be investigated. In particular, PostgreSQL is more promising since it is an open source project, used by a large community. The transition from ORACLE to PostgreSQL should be easier than to a NoSQL solution, since the schema of the database, queries, procedures, analytical functions and the job execution provided by PostgreSQL is similar to the ORACLE. The performance of PostgreSQL and ORACLE should be compared with the same amount of data.

Evaluation and Use of ElasticSearch

As stated in the previous section, the Agile Infrastructure Monitoring team uses ElasticSearch as the storage backend. ElasticSearch was also evaluated for the WLCG monitoring applications. For evaluation and testing purposes, the ElasticSearch cluster was setup and supported by IT-SDC group. In case the technology is considered for the production use, we assume that the cluster would be provided centrally for all potential users in the IT department

Evaluation and Use of Hadoop/Hbase

Another alternative to ORACLE being evaluated is the Hadoop ecosystem. There is already the support for a Hadoop cluster within IT. The group also created its own Hadoop/HBase cluster

and performed an evaluation of the performance. The early results shown that this approach is worse than ORACLE for time-series data.

HDFS should be considered as an archival and offline analysis solution.

6 Status and Plans for Each Component

6.1 Collectors and Probes

Initial Situation

Before the creation of this group, the standard way of running probes was to configure them in Nagios. The four experiments have already defined a set of probes that test service functionality. The submission of all the worker nodes probes was done using WMS.

Experiments could inject their own metrics in the experiment "nagioses", as long as the metrics were properly defined in POEM, and the services and sites were defined in ATP.

Besides all the experiment tests, the services were tested with the credentials of the OPS virtual organization. Some sites have configured high priority queues to handle these tests. Currently, official monthly reports with site availabilities are based on the results of OPS tests.

Actions Performed During Phase 1

- In spring this year the monthly availability reports based on the experiment test results were enabled and validated. From October to January 2014 two sets of reports will be generated. Sites are encouraged to check new reports and to notify the monitoring team in case they see any issues with them. In January 2014 the reports based on the experiment tests will become official ones.

Consolidated Solutions

The consensus was that Nagios should stay as an optional way of submitting remote tests. There is also a need for the modification of the Nagios probes, in order to submit through Condor-G and to complement the direct CREAMCE submission probes with the worker node test payloads.

At the same time, there should be a way of incorporating metrics produced by other sources, like for example HammerCloud, DIRAC or MonALISA. The injection should be done at the transport layer, without having to pass by the Nagios box. This should be straightforward, assuming that the monitoring architecture follows the layered design introduced at the beginning of this section. If the format in which the information has to be passed to the transport layer is well defined, any authorized application that follows that format would be able to publish metrics into the system. This would allow also the submission of metrics from the standard workflows of the experiment, instead of depending on tests submitted through different channels. There should be an easy way of adding/removing metrics, sites and services.

The group is also investigating the possibility of the simplification of the current Nagios probe framework.

OPS tests are not needed anymore and can be stopped after the end of EGI.

Open Issues

No Open issues identified.

Actions for Phase 2

Task ID	Main Task	Who	Priority
COLL-1	Implement Condor-G probes	Luca	High

COLL-2	Ensure that experiments can inject their own values in the transport layer	Marian, Costin, Stefan	High
COLL-3	Complement the CREAMCE submission probes with the worker node test payloads	Luca	High
COLL-4	Simplify the Nagios probe framework	Luca	Medium
COLL-5	VOfeed becomes the only source of description of the experiment topology	Ivan	High
COLL-6	Include filtering and more flexibility on FQANs on the profile description	Jacobo	High
COLL-7	Stop OPS tests after the end of EGI	Marian	Medium
COLL-8	Support Nagios as an optional component	Marian	High

6.2 Transport

Initial Situation

This layer decouples the information providers and the storage, and enables asynchronous communication between producers and consumers of the monitoring data. This approach facilitates data access from multiple consumers, and it endures better fault-tolerance and scalability. At the moment, there are three different transport modes used:

- Messaging
- HTTP put/get
- MonALISA UDP

Actions Performed During Phase 1

No action needed, the existing transport mechanisms were well known.

Consolidated Solutions

Each mode is best suited for a different use case, and the current approach seems to be adequate for the foreseeable future. For usage of the Messaging system there is a set of libraries developed by the CERN Messaging team, which serves as an abstraction layer between the particular messaging implementation and applications producing or consuming monitoring data. It is important to keep in contact with the CERN Messaging team and follow their recommendations.

The DDM Monitoring currently uses HTTP as a transport mechanism. This should be migrated to messaging.

Open Issues

- The AI Monitoring team uses a different technology, FLUME. As explained in a previous chapter, this does not seem to fit the use cases covered in this document due to the dynamic number of clients.

Actions for Phase 2

Task ID	**Main Task**	**Who**	**Priority**

TRSP-1	Coordinate the integration of the latest libraries provided by the Messaging team into the Monitoring applications	Luca, Lionel	Medium
TRSP-2	Migrating DDM Monitoring to messaging	David	Medium

6.3 Storage - DB Schema

Initial Situation

For historical reasons, there are too many different schemas, even if the data structure is very similar. Moreover, some of the applications interact with each other through shared database tables, instead of using well-defined APIs. This creates too many dependencies between the applications, and makes it impossible to introduce changes in any component without touching the whole chain.

Actions Performed During Phase 1

- Identified the schemas of SSB and SAM as suitable for merging.

Consolidated Solutions

A more modular approach is recommended, where each component presents an API that will be used by other components. This way, it will be easier to replace/modify components and evaluate alternative implementations.

Considering functionality and database schema, SSB and SAM are very similar: given a set of instances and metrics, record their status evolution over time. The approach used in SSB is more generic, allowing the instances to be services, sites or even channels. It also simplifies the definition of new metrics, and gives the possibility of combining metrics into views. Moreover, since the SSB stores by default only the status changes of the metrics (instead of every single value), the size of the database stays under control. For all these reasons, it was decided to use the SSB schema to store also the SAM data. This way, the number of schemas maintained by the group decreases, and it even offers more functionality, since it allows the combination of SAM metrics with any other experiment-defined metrics.

A similar approach can be taken for merging the HammerCloud and Job Monitoring schemas.

Open Issues

No open issues.

Actions for Phase 2

Task ID	Main Task	Who	Priority
SCHM-1	Combine SSB and SAM into a common Metric Store schema	Ivan/Jacobo/Pedro	High
SCHM-2	Investigate HammerCloud schema, and check the differences with JobMonitoring	Valentina/Edward	Medium

6.4 Storage - DB Back-end Technologies

Initial Situation

All the applications based at CERN are currently running on top of ORACLE. For SAM, MySQL was also supported.

Actions Performed During Phase 1

There was an investigation for alternatives to ORACLE, in particular NoSQL solutions.

The group evaluated Hadoop/HBase and ElasticSearch (the later following the advice from the Agile Monitoring representative, since they also use ElasticSearch in their schema). Two testbeds, one with Hadoop/HBase and the other with ElasticSearch, were instantiated and evaluated for different monitoring components. The main limitation of the current version of ElasticSearch is the grouping by multiple fields. For some applications, like the Data Management or Job Accounting, this is a very serious drawback. These applications depend on being able to aggregate the data by multiple fields at the same time (site, user, service, date, job category, etc.) For other applications, like SSB or SAM, grouping by a single column would be enough.

Consolidated Solutions

ElasticSearch might be an option for the Common Metric Store use case.

For other use cases, like DDM and Job Accounting, ElasticSearch should be evaluated again once it offers support for multiple grouping. More details about this evaluation can be found in the Appendix B.

Open Issues

Actions for Phase 2

Task ID	Main Task	Who	Priority
DBBE-1	Evaluate the Common Metric Store on top of ElasticSearch	Ivan/Pedro	Medium
DBBE-2	Evaluate ElasticSearch 1.0. To be done once it becomes available	David/Pedro/Edward	Medium
DBBE-3	Evaluate the Common Metric Store on top of PostgreSQL	Robert	Medium
DBBE-4	Evaluate HDFS as an archival option	Robert/Pedro	Medium

6.5 Storage - Data Retention Policy

Initial Situation

Each application had a different policy concerning data retention. In particular, the non-enforced policy for keeping the tests results was:

- 3 months for tests results
- 12 months for availability results

Actions Performed During Phase 1

- A cleanup of the old tests results was done. The cleaned reduced the size of the database to one quarter of its initial size

Consolidated Solutions

There is an agreement that the lifetime of the test results are adequate. At the same time, the availability results should be kept indefinitely.

Open Issues

No open issues were identified.

Actions for Phase 2

Task ID	Main Task	Who	Priority
RETE-1	Implement automatic data cleanup	Robert	Medium

6.6 Data Processing (Aggregation)

Initial Situation

There are two main common use cases for data processing:

- The first one is processing the current value of a set of metrics, and as a result, produces a new metric. For example, the service status is calculated combining status of multiple service instances.
- The second case is the aggregation of metric results over a given time interval. For example, calculating the average transfer throughput based on set of distinct measurements.

In both cases a new metric is produced either as a result of processing of set of other metrics or aggregating a single metric or several metrics over time.

Actions Performed During Phase 1

No actions needed.

Consolidated Solutions

The output of the aggregation layer should be fed back into the system as a new metric, which could even be further included in the next processing iterations. This way, the same visualization layer as for the basic metrics can be reused, and, on top of that, the aggregated metrics can be as well aggregated even further. This is a concept that both CMS and ATLAS were already using in the SSB with their 'Site readiness' exercises. The goal now is that the WLCG monitoring group will offer some aggregations that will calculate the site availabilities and reliabilities.

Open Issues

No issues identified.

Actions for Phase 2

Task ID	Main Task	Who	Priority
AGGR-1	Implement Status aggregation on top of SSB metrics	Jacobo	High
AGGR-2	Include the downtime information in the status report	Ivan	High
AGGR-3	Create monthly availability and reliability reports based on SSB data and aggregation procedures	Ivan	High

6.7 Visualization and API

Initial Situation

For historical reasons, there are plenty of differences across the applications maintained by IT-SDC:

- Different backend frameworks: Django, Dashboard
- Some applications request data through AJAX, some others request rendered html fragments
- Data tables is not used consistently across all tabular views
- MVC framework is not used consistently, or not used at all sometimes
- Plots come from different sources: Google Image Charts API, Google Charts API, Highcharts, Graphtool, Raphael
- Content is rendered using different technologies: XSLT, HTML skeleton, Django templates

Actions Performed During Phase 1

No actions needed.

Consolidated Solutions

As the main design principle for the visualization of the WLCG monitoring applications, it was recommended to have a separation between the server side and the client side. The server side should produce basic html skeleton pages and a REST API with json for AJAX requests. There are frameworks that should be considered here, like Django, and server caching tools like Memcache and Varnish.

The client part would be composed of JavaScript libraries, following a Model-View-Controller paradigm. Some monitoring applications already use xbrowse for this (an MVC framework built on top of jQuery). Open source client-side MVC frameworks should be considered for existing and future applications. For plots, Highcharts is the obvious choice.

A common look and feel should be enforced where relevant. This is already the case for all the transfer monitoring applications (WLCG transfer Dashboard, ATLAS DDM Dashboard, xrootd dashboards), accounting applications (ATLAS and CMS historical views, ATLAS DDM accounting portal).

All the information available for the visualization layer should also be available in machine-readable format.

Another point that was discussed was how the information can be propagated to the site administrators. At the moment, several sites have incorporated information from the current SAM monitoring into their local Nagios. This was identified as a very useful way of propagating the data.

Open Issues

None.

Actions for Phase 2

Task ID	Main Task	Who	Priority
VIS-1	Ensure separation between server side and client side visualization for all the applications	David	High
VIS-2	Investigate open source client-side MVC frameworks	David	High
VIS-3	Ensure that the applications follow the MVC paradigm	David	Medium
VIS-4	Introduce caching tools like Memcache or varnish on the server side	Alex	Medium
VIS-5	Create a Nagios plugin to incorporate the new WLCG monitoring application should provide a way for the sites to inject some of the metrics into their local nagioses	PIC	High
VIS-6	Provide SUM-like functionality on top of SSB	Ivan	High

7 Deployment and Support

The area of deployment and support can also be consolidated,. It would simplify the operations and maintenance of the overall system.

7.1 Deployment

Initial Situation

Currently, the deployment and management of services was done in different ways, on a case-by-case basis. Most of the machines were based on Quattor, with manual intervention in multiple machines, and different ways of managing the configuration: Quattor templates, sindes, yaim, etc.

Actions Performed During Phase 1

- Creation of an OpenStack project to host all the machines.
- Migration of some of the Dashboard machines to the AI

Consolidated Solutions

The deployment should take advantage of the new developments done in the Agile Infrastructure project.

It is beneficial to use a standard way for the management of the entire cluster, following the lead of the IT department, which means using OpenStack, puppet, foreman and hiera. The process has already started, and, at the time of writing this report, more than 50 machines are already deployed in OpenStack, relying on the Agile Infrastructure.

All the machines needed for the monitoring infrastructure are under IT-SDC control. This means that the support of the platforms can be reduced also to the CERN provided platforms. In practical terms, this means that all the machines used by the group for this purpose should be on SLC6.

Another topic related to the deployment is the way applications are built and packaged. Before the working group was created, some of the applications used the Dashboard Build system (an

in-house developed framework built on top of the python distutils and its own yum repository), and other applications used the EGI Koji. It is recommended that all the applications supported by the team follow the same procedure. The procedure should be aligned with the strategy of the rest of IT. The CERN Koji is the clear candidate, and Bamboo can also be considered as a continuous integration tool.

The last topic on this layer is the source control mechanism. At the moment, there are several SVN repositories that contain different applications. It would be better to concentrate them into a single common repository, and use this opportunity also to move to newer version control tools like Git.

Open Issues

No issues identified.

Actions for Phase 2

Task ID	Main Task	Who	Priority
DEPL-1	Migration of the machines to AI. A lot of the work was already done by Mike	Pablo	High
DEPL-2	Ensure that all the machines are on SLC6	Pablo	Medium
DEPL-3	Use the CERN Koji instead of the Dashboard Build system and the EGI Koji	Pablo Marian	Medium
DEPL-4	Integrate Bamboo as a Continuous Integration System	Pablo	Low
DEP-5	Move from SVN to Git	Pablo	Medium

7.2 Documentation and User Support

Initial Situation

At the moment, the group uses four different tools: Confluence and Twiki for end user documentation, and sphinx and distutils for the source code documentation.

The number of ticketing tools should also be revisited. When the group started, there were three Savannah projects, five Jira projects, two GGUS support units and two ServiceNow support units. It has to be taken into account that savannah support is supposed to stop by the end of the year.

Actions Performed During Phase 1

The GGUS support units were merged, and so were the snow support units.

Consolidated Solutions

Migrating the current documentation to a common place has not been identified as a priority. All the new documentation should be done in a common way, and it was recommended to stay out of confluence. It has to be evaluated if Twiki or Drupal will be enough for these cases.

The savannah items that need to be kept have to be moved to JIRA, and there should be a common category for the monitoring tools maintained by the group.

Open Issues

No issues identified.

Actions for Phase 2

Task ID	Main Task	Who	Priority
USER-1	Migrate from Savannah to Jira	Pablo	Medium
USER-2	Evaluate Drupal and Twiki as alternatives for documentation	Julia	Medium
USER-3	Ensure that all the new documentation follows the previous recommendation	Julia	Medium

7.3 Recurrent Tasks

Initial Situation

There are tasks that have to be periodically executed. There are several options of how to deal with them, ranging from cronjobs, dedicated UNIX daemons or integrating the tasks into the scheduler of the RDBMS. At the moment, the three approaches are used on different monitoring applications.

Actions Performed During Phase 1

No actions needed

Consolidated Solutions

The three implementations have their advantages and limitations, and it should be decided, on a case-by-case basis, which one is the best approach.. There are tools that help implemented each of the solutions. For instance, for the daemons, the simplevisor tool provided by IT eases the management of the different services. The group also identified a couple of small cases where the selected approach could be improved.

Open Issues

No issues identified.

Actions for Phase 2

Task ID	Main Task	Who	Priority
TASK-1	Incorporate simplevisor	Alex	Medium

8 Transition Period

The work on the simplified WLCG infrastructure has already started. The new infrastructure, which will include the simplified components, will be operated in parallel with the current one. Supporting two parallel infrastructures is required due to commitments to EGI to support SAM in its actual implementation until April 2014. Furthermore, having in place the two systems would allow crosschecking results, and therefore would create ideal conditions for validation of the new one. The prototype of the new WLCG Monitoring will go through several fast-cycle iterations with the end users. It should lead to a production quality system before the end of the summer of

2014. At that point, and once all the interested parties are satisfied with the new implementation, the switch can be done and the old infrastructure can be decommissioned.

According to agreement with EGI, the WLCG Monitoring team should help in the migration of the EGI central services to Greece and Lyon prior to April 2014.

8.1 SAM Transition

Initial Situation

SAM is used both for WLCG and for EGI.

Actions Performed During Phase 1

No issue was identified.

Consolidated Solutions

The support of SAM for EGI will be passed on to the consortium of CNRS, GRNET and SRCE before the end of April 2014.

The support of SAM for WLCG will be needed until the new WLCG Monitoring Consolidation can take over. The current SAM infrastructure will be used to validate the results of the new system.

Open Issues

No issues identified.

Actions for Phase 2

Task ID	Main Task	Who	Priority
SAM-1	Transition of the current SAM infrastructure to CNRS/GRNET/SRCE	Marian	High
SAM-2	Support the current SAM for WLCG until the new system is ready	Marian	High

8.2 Prototype Transition

Initial Situation

A prototype of the new WLCG Monitoring was deployed at the beginning of October. For the time being, it has a very limited functionality.

Actions Performed During Phase 1

No action.

Consolidated Solutions

The prototype allows a fast feedback cycle between the developers and the users. The prototype should evolve into a production quality system by the end of the project.

Open Issues

No issues identified.

Actions for Phase 2

Task ID	Main Task	Who	Priority
PTTY-1	Create a prototype for fast feedback cycle	Everyone	High
PTTY-2	Evolve the prototype to production quality service	Everyone	High

8.3 HammerCloud Transition

Initial Situation

For historical reasons, HammerCloud does not share implementation and deployment, development and support model with the rest of the monitoring tools.

Actions Performed During Phase 1

Nothing done yet

Consolidated Solutions

Align the deployment, development and support of HammerCloud with other monitoring applications.

Open Issues

No issues identified.

Actions for Phase 2

Task ID	Main Task	Who	Priority
HCTR-1	Deploy HC on AI	Valentina/Pablo	High

8.4 REBUS Transition

Initial Situation

REBUS is used as the entry point of information about the WLCG resources. It has its own database where it keeps the historical values.

Using the information of REBUS as an input, an excel report is manually generated every month, and distributed by the WLCG office. It is also heavily used from other applications like the Job Accounting and DDM Accounting.

Actions Performed During Phase 1

Nothing done yet.

Consolidated Solutions

Given that the information currently stored in REBUS is very important to the other components of the WLCG Monitoring consolidation, its integration within the system should be evaluated. This would mean that REBUS would follow the same deployment and support model as all the other monitoring applications supported by IT-SDC. On top of that, the data currently stored in REBUS could be migrated to the Common Metric Store. This would eliminate another different database schema supported by the group, and it would allow that the visualization layer could also be shared.

Open Issues

No issues identified.

Actions for Phase 2

Task ID	Main Task	Who	Priority
REBU-1	Evaluate the common metric store for REBUS. This task will also decide if the following tasks are needed	Laurence/Pablo	High
REBU-2	Deploy REBUS on the Agile Infrastructure	Laurence/Pablo	High
REBU-3	Insert the Rebus metrics in the common store	Pablo	High
REBU-4	Create plots of CPU/Wall time delivered, disk and tape used	Ivan	High
REBU-5	Automatically create monthly accounting reports	Ivan	Medium

8.5 Other applications Transition

Initial Situation

There are other monitoring applications supported by the group. The full list of applications can be found in 'Appendix A: Usage of monitoring applications', including their current usage by the experiments.

Actions Performed During Phase 1

Identified applications that are no longer needed.

Consolidated Solutions

The applications that are not needed anymore should be stopped. There are other applications that are used by the WLCG community, and where no special actions were needed. These applications still need to be aligned with the deployment, development and support model described in this document.

Open Issues

No issues identified.

Actions for Phase 2

Task ID	Main Task	Who	Priority
APPS-1	Stop MyWLCG job trends, FTS Transfers, T0/T1 Siteview and SAM Tree Map. This application should be stopped at least on the MyWLCG. They might be needed in MyEGI until the end of April	Marian	Medium
APPS-2	Stop ALICE FTD Efficiency and CMS Datasets	Pablo	Medium
APPS-3	Ensure that all the Job Monitoring applications are moved to the common framework and deployment	Edward	High

APPS-4	Ensure that all the Data Monitoring applications are moved to the common framework and deployment	David	High
APPS-5	Ensure that GoogleEarth and Siteview are moved to the common framework and deployment	Edward	Medium
APP-6	Ensure that the WLCG Transfers applications are move to the common framework and deployment	Alex	High

9 Time Table for Phase 2

These are all the tasks that have been identified in the previous chapters. They will be inserted in JIRA, and their progress will be followed up there.

Task ID	Main Task	Who	Priority
COLL-1	Implement Condor-G probes	Luca	High
COLL-2	Ensure that experiments can inject their own values in the transport layer	Marian, Costin, Stefan	High
COLL-3	Complement the CREAMCE submission probes with the worker node test payloads	Luca	High
COLL-4	Simplify the Nagios probe framework	Luca	Medium
COLL-5	VOfeed becomes the only source of description of the experiment topology	Ivan	High
COLL-6	Include filtering and more flexibility on FQANs on the profile description	Jacobo	High
COLL-7	Stop OPS tests after the end of EGI	Marian	Medium
COLL-8	Support Nagios as an optional component	Marian	High

Task ID	Main Task	Who	Priority
TRSP-1	Coordinate the integration of the latest libraries provided by the Messaging team into the Monitoring applications	Luca, Lionel	Medium
TRSP-2	Migrating DDM Monitoring to messaging	David	Medium

Task ID	Main Task	Who	Priority
SCHM-1	Combine SSB and SAM into a common Metric Store schema	Ivan/Jacobo/Pedro	High
SCHM-2	Investigate HammerCloud schema, and check the differences with JobMonitoring	Valentina/Edward	Medium

Task ID	Main Task	Who	Priority
DBBE-1	Evaluate the Common Metric Store on top of ElasticSearch	Ivan/Pedro	Medium

DBBE-2	Evaluate ElasticSearch 1.0. To be done once it becomes available	David/Pedro/Edward	Medium
DBBE-3	Evaluate the Common Metric Store on top of PostgreSQL	Robert	Medium
DBBE-4	Evaluate HDFS as an archival option	Robert/Pedro	Medium

Task ID	**Main Task**	**Who**	**Priority**
RETE-1	Implement automatic data cleanup	Robert	Medium

Task ID	**Main Task**	**Who**	**Priority**
AGGR-1	Implement Status aggregation on top of SSB metrics	Jacobo	High
AGGR-2	Include the downtime information in the status report	Ivan	High
AGGR-3	Create monthly availability and reliability reports based on SSB data and aggregation procedures	Ivan	High

Task ID	**Main Task**	**Who**	**Priority**
VIS-1	Ensure separation between server side and client side visualization for all the applications	David	High
VIS-2	Investigate open source client-side MVC frameworks	David	High
VIS-3	Ensure that the applications follow the MVC paradigm	David	Medium
VIS-4	Introduce caching tools like Memcache or varnish on the server side	Alex	Medium
VIS-5	Create a Nagios plugin to incorporate the new WLCG monitoring application should provide a way for the sites to inject some of the metrics into their local nagioses	PIC	High
VIS-6	Provide SUM-like functionality on top of SSB	Ivan	High

Task ID	**Main Task**	**Who**	**Priority**

DEPL-1	Migration of the machines to AI. A lot of the work was already done by Mike	Pablo	High
DEPL-2	Ensure that all the machines are on SLC6	Pablo	Medium
DEPL-3	Use the CERN Koji instead of the Dashboard Build system and the EGI Koji	Pablo Marian	Medium
DEPL-4	Integrate Bamboo as a Continuous Integration System	Pablo	Low
DEP-5	Move from SVN to Git	Pablo	Medium

Task ID	**Main Task**	**Who**	**Priority**
USER-1	Migrate from Savannah to Jira	Pablo	Medium
USER-2	Evaluate Drupal and Twiki as alternatives for documentation	Julia	Medium
USER-3	Ensure that all the new documentation follows the previous recommendation	Julia	Medium

Task ID	**Main Task**	**Who**	**Priority**
TASK-1	Incorporate simplevisor	Alex	Medium

Task ID	**Main Task**	**Who**	**Priority**
SAM-1	Transition of the current SAM infrastructure to CNRS/GRNET/SRCE	Marian	High
SAM-2	Support the current SAM for WLCG until the new system is ready	Marian	High

Task ID	**Main Task**	**Who**	**Priority**
PTTY-1	Create a prototype for fast feedback cycle	Everyone	High
PTTY-2	Evolve the prototype to production quality service	Everyone	High

Task ID	**Main Task**	**Who**	**Priority**

HCTR-1	Deploy HC on AI	Valentina/Pablo	High

Task ID	**Main Task**	**Who**	**Priority**
REBU-1	Evaluate the common metric store for REBUS. This task will also decide if the following tasks are needed	Laurence/Pablo	High
REBU-2	Deploy REBUS on the Agile Infrastructure	Laurence/Pablo	High
REBU-3	Insert the Rebus metrics in the common store	Pablo	High
REBU-4	Create plots of CPU/Wall time delivered, disk and tape used	Ivan	High
REBU-5	Automatically create monthly accounting reports	Ivan	Medium

Task ID	**Main Task**	**Who**	**Priority**
APPS-1	Stop MyWLCG job trends, FTS Transfers, T0/T1 Siteview and SAM Tree Map. This application should be stopped at least on the MyWLCG. They might be needed in MyEGI until the end of April	Marian	Medium
APPS-2	Stop ALICE FTD Efficiency and CMS Datasets	Pablo	Medium
APPS-3	Ensure that all the Job Monitoring applications are moved to the common framework and deployment	Edward	High
APPS-4	Ensure that all the Data Monitoring applications are moved to the common framework and deployment	David	High
APPS-5	Ensure that GoogleEarth and Siteview are moved to the common framework and deployment	Edward	Medium
APP-6	Ensure that the WLCG Transfers applications are move to the common framework and deployment	Alex	High

10 References

1. https://twiki.cern.ch/twiki/bin/view/LCG/WLCGMonitoringConsolidation
2. https://indico.cern.ch/categoryDisplay.py?categId=4981
3. https://twiki.cern.ch/twiki/bin/view/AgileInfrastructure/AreaMonitoring
4. https://twiki.cern.ch/twiki/bin/view/AgileInfrastructure/AreaMonitoringMessageSpecification
5. http://logstash.net/
6. http://flume.apache.org/
7. http://elasticsearch.org/
8. http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html
9. http://www.kibana.org/
10. http://dashboard.cern.ch/
11. https://tomtools.cern.ch/confluence/display/SAMDOC/SAM+Overview
12. http://rebus.cern.ch/
13. http://wlcg-mon.cern.ch/
14. https://twiki.cern.ch/twiki/bin/view/LCG/WLCGMonitoringConsolidation#Architecture_reviews
15. http://

11 Appendix A: Usage of monitoring applications

- Used
- nice to have
- not used

Category	ALICE	ATLAS	CMS	LHCb	Comments
Job Monitoring					
Current view		Interactive View	Interactive view		
		Task Monitoring	Task Monitoring Task Monitoring on Android		
Accounting		Historical View	Historical view		
WLCG job trends	MyWLCG job trends				
Data Management					
Current view		DDM2	CMS Datasets		
Accounting		DDM Accounting			
Transfers	WLCG Transfer Dashboard				
	FTD efficiency	FAX	AAA		
		MyWLCG FTS Transfers			
Site/Service Monitoring					
VO Feed	ALICE vo feed		CMS vo feed		
SSB	ALICE SSB	ATLAS SSB	CMS SSB	LHCb SSB	
SUM	ALICE SUM	ATLAS SUM	CMS SUM	LHCb SUM	
Regional/Experiments SAM/Nagios	ALICE nagios	ATLAS nagios	CMS nagios	LHCb nagios	Other 35 regional instances (NGIs, ROCs, CERN)
VO SAM/Nagios	MIDMON				Middleware Nagios, Security Nagios, former

					gLExec Nagios, SuperB Nagios, etc.
SAM Operational Monitoring (ops-monitor)		OPS-MONITOR			Monitors sam-atlas/alice/lhcb/cms prod and preprod SAM, SAM central, etc.
SAM central service		CMS SAM GridMon			Web API, myWLCG A/R,
SAM WLCG Reports		ATLAS CMS Monthly reports			
SAM A/R Trends		A/R Trends			
SAM T0/1 SiteView		T0/1 SiteView			
SAM Tree Map		GridMap			
SAM Site Nagios		SAM Nagios installation			
Critical Services		ATLAS might be interested (the application has not been setup)	CMS Critical Services		
SAM Probe Development Framework and SAM probes		Probe development documentation			
Personalized Dashboard		Personalized dashboard			
Dissemination					
Google Earth		Web interface to Google Earth			Also installed at Globe, CC, CMS Centre Meyrin
Siteview		CMSATLAS from site admin POV (tbd)? SiteView			
Development Management					
SAM Nightly Validation Framework		SAM validation			Used to run sam-*-dev boxes and CI

12 Appendix B. ElasticSearch Evaluation

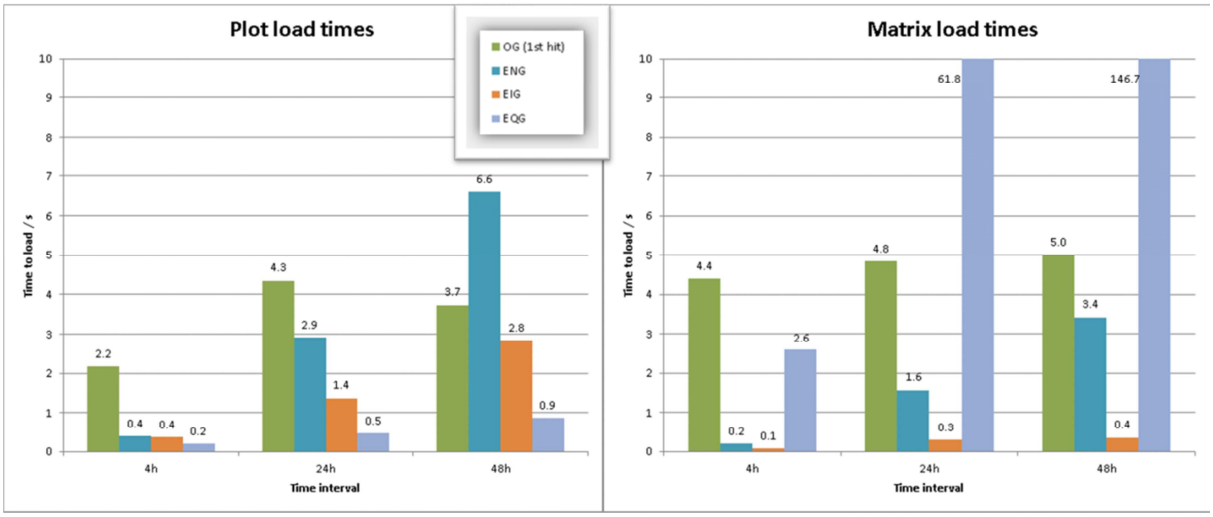
[Taken from E.Karavakis et al., Processing of the WLCG monitoring data using NoSQL]

These plots present the load time for two different use cases of the Dashboard DDM:

- matrix statistics that allow filtering and grouping by multiple fields
- plot statistics that are time series data and allow filtering and grouping by multiple fields

The current version of ElasticSearch (0.90.5) does not support grouping by multiple fields for statistical aggregations, but this will be supported in version 1.0. Since the WLCG Transfers application offers grouping by multiple fields, many workarounds were investigated, resulting in the following options:

- Oracle Grouping (OG). Query using 'group by' for user selected grouping fields
- ElasticSearch No Grouping (ENG). Query for all data and then perform the grouping in the web action from the Python side
- ElasticSearch Index Grouping (EIG). Add a single field in index with all possible grouping fields concatenated as strings
- ElasticSearch Query Grouping (EQG). Query to list number of distinct combinations of selected grouping fields and then query that many times, filtering by distinct combinations



Comparison between Oracle 1st hit (un-cached result) and multiple grouping methods of ElasticSearch.