WLCG Monitoring Consolidation Project

October 2013

Version 0.0; 23/10/13

Pablo Saiz [a)], Pedro Andrade [a)], Julia Andreeva [a)], Marian Babik [a)], Latchezar Betev [a) 1)], Lionel Cons [a)], Josep Flix [b)], Alessandra Forti [c)2)], Maria Girone [a)], Costin Grigoras [a) 1)], Edward Karavakis [a)], Maarten Litmath [a)], Nicolo Magini [a)], Luca Magnoni [a)], Stefan Roiser [a)], Andrea Sciaba [a)]

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

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

Acknowledgements

Some other people

Executive Summary

Table of Contents

1 Introduction

The mandate of the WLCG Monitoring Consolidation Project is to do 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.

The goals are to:

- Reduce the complexity of the system
- Ensure simplified and more effective operations, support and service management
- Encourage an efficient deployment strategy, with a common development process
- Unify, where possible, the implementation of the monitoring components

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. Before this project was created, there were 18 FTE dedicated to Grid monitoring.

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

Some of the tasks of this project include:

- Gather requirements from experiments, regarding the remote testing of the distributed sites and services.
- Review all the existing monitoring applications supported by IT-SDC, define priorities and agree on the further development and support model for applications which would require development effort for customization
- Understanding needs of the sites, famous question 'I like to see at a single display how my site is working for all LHC VOs'
- Review of the test submission part. Should we continue to have several ways for test submission: one for stress testing, another one for regular tests (Hammer Cloud/Nagios/JobPilot)?
- Define an overall architecture
- Use, where possible, a common framework and common implementations for all the components
- Create a single web UI to WLCG monitoring applications

More information can be found at [1]

2 Timetable

The project has been divided in two phases:

- The first phase, from July-October 2013, should do the analysis of the current status and agree on a plan to follow during the second phase.
- The second phase, from October 2013 to July 2014 should focus on implementing and deploying the plan defined during the first stage

This document is being created at the end of the first phase, and it should be the guideline for the direction of the second phase.

3 Actions taken so far

The group has been very active during these months. There were fortnightly meetings with status reports and during those meetings the architecture of the new system was presented and discussed.

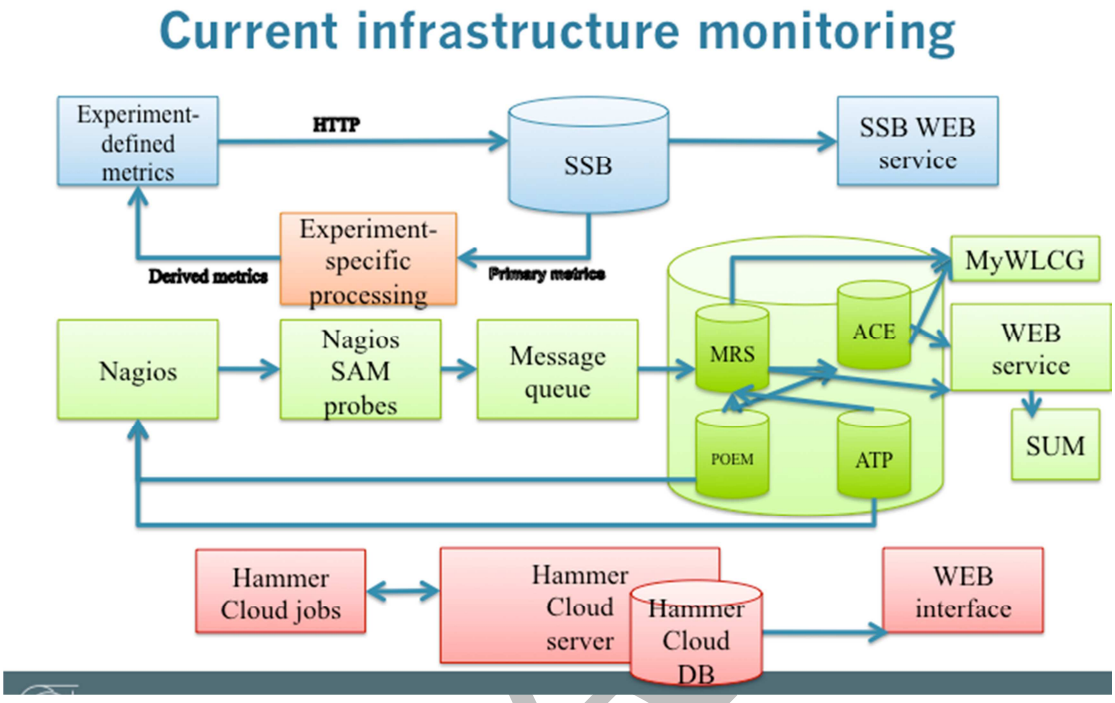
A review of the monitoring applications supported by IT-SDC was presented, and the experiments gave their feedback on the usage of the application. This can be seen in the Apendix A of this document. This review already identified a couple of applications that were not necessary to maintain anymore. These applications are listed on chapter five of this document1

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

After that, the group worked on layered design that could be applied to the monitoring tools. The design will be explained in more detail in the next section.

4 Current schema

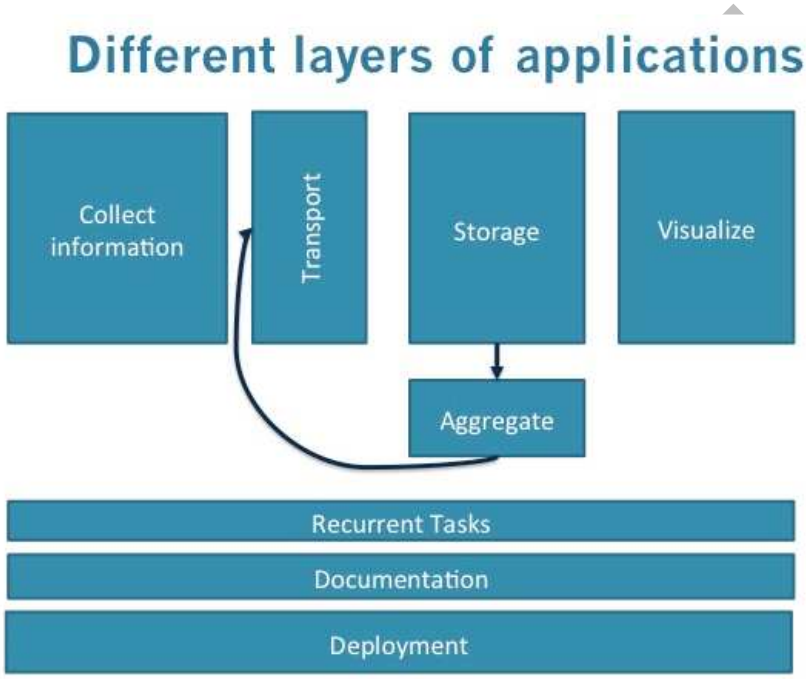
The current schema of the monitoring applications on the Site and Services infrastructure can be seen in figure 1



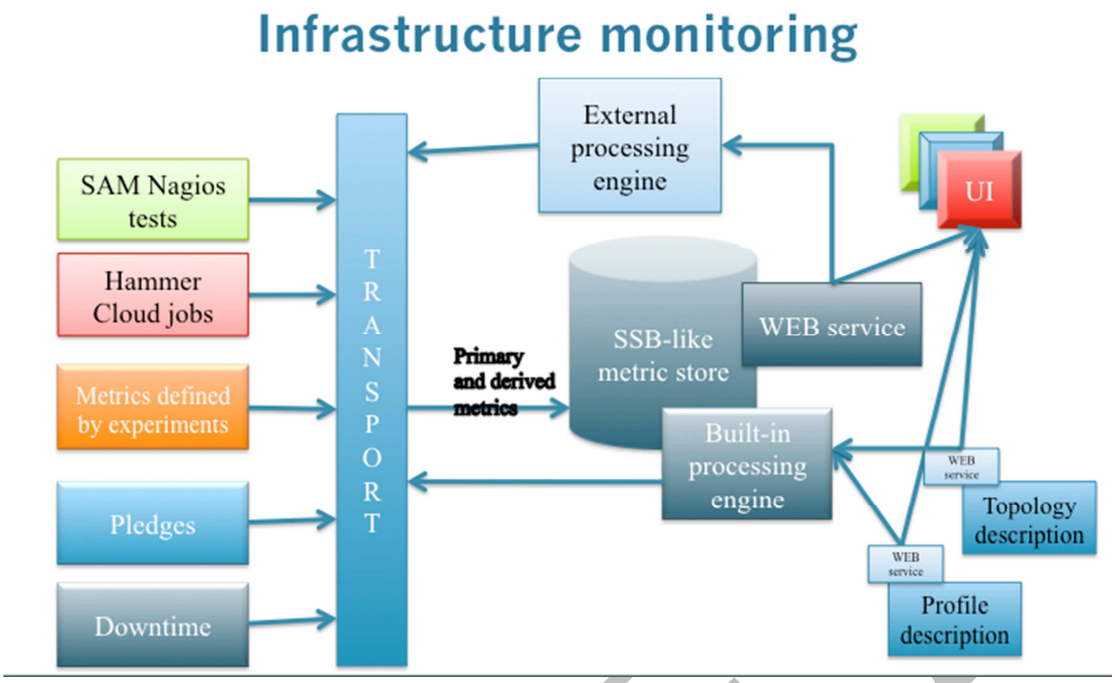
5 Proposed schema

The design proposed for the applications is depicted in Figure 1. 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 designed offers a great level of flexibility. 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 API.



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



The rest of this section will go through each of the layers, identifying the areas that need more work. Whenever an action has been identified, there will be a reference to the next section, where all the actions to be taken have been listed.

5.1 Deployment

The deployment should take advantage of the new developments done in the Agile Infrastructure group. Before this working group was created, 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…

It would be advisable to use a standard way for the management of the entire cluster, follow 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, relaying on the Agile Infrastructure [2.a].

Excluding the probes, 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 [2.b].

Another topic related to the deployment is the way applications are built. 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 would be better if all the applications supported by the team use the same procedure, and even better if that procedure was aligned with the rest of IT. The

CERN Koji is the clear candidate, and Bamboo can also be considered as a continuous integration tool [2.c].

The last topic on this layer is the source control mechanism. At the moment, there are several svn repositories that hold different applications. It would be better to concentrate them into a common repository, and use this opportunity also to move to more recent version control tools like git [2.d].

5.2 Documentation

At the moment, the group uses four different tools: confluence and wikis for end user documentation, and sphinx and distutils for the source code documentation. 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 wiki or drupal will be enough for this cases [3.a].

The number of ticketing tools should also be revisited. When the group started, there were three savannah projects, five savannah projects, two ggus support units and two service-now support units. It has to be taken into account that savannah support is supposed to stop by the end of the year. The GGUS support units should be merged [3.b], the snow support as well [3.c], the savannah items that need to be kept have to be moved to jira [3.d], and there should be a common jira category [3.e] for the monitoring tools maintained by the group [3.f].

5.3 Recurrent Tasks

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

5.4 Probes

Before the group was created, the standard way of running probes was to configure them in nagios. The four experiments have already defined a set of probes that test the functionality of some of the services. This approach could be complemented with information gathered from other sources. ALICE, ATLAS and LHCb are already looking into feeding their own metrics.

The consensus was that nagios should stay as an optional way of submitting probes [4.a]. There is also a need for new probes, in particular for the submission through Condorg [4.b] and CREAMCE [4.c] directly.

At the same time, there should be a way of incorporating metrics produced by other sources [4.d], like hammercloud, DIRAC or MonALISA. This is straightforward thanks to the layered designed introduced at the beginning of this section. If the format in which the information has to be passed to the transport is well defined, any application that follows that format can publish metrics into the system. This would allow even the submission of metrics from the standard workflow of the experiment, instead of depending on tests submitted through different channels.

5.5 Transport

This layer decouples the information providers and the storage. Thanks to it, the information can be consumed by multiple components, and it also provides fault-tolerance and scaling mechanisms. At the moment, there are three different transport modes used:

- Messaging
- HTTP put/get
- MonALISA UDP

Each mode is best suited for a different use case, and the current approach seems to be adequate for the foreseeable future. There are new versions of the libraries that should be included [5.a]. It is important to keep in contact with the CERN messaging team [5.b] and follow their recommendations.

((((Evaluate Flume [5.c] ???))))

5.6 Storage

This is one of the areas were the current system can be improved the most. There are too many different schemas, even if the type of data that has to be stored is very similar. Moreover, some of the applications interact with each other using directly the database tables, instead of using well-defined API. This creates too many dependencies between the applications, and makes it difficult to integrate.

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 components and evaluate other alternatives.

5.6.1 Common schema

SSB and SAM solve the same issue: given a set of instances and metrics, record their status over time. The approach used on 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[6.a]. 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.

5.6.2 DB backends

In parallel to this activity, there was also an investigation for ORACLE alternatives, in particular of no-sql solutions. At the same time that this investigation was done, the group evaluated hadoop 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 [6.c] and the other with ElasticSearch [6.b], 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. Therefore, the WLCG monitoring consolidation group suggested to evaluate again ElasticSearch for DDM and Job Accounting once it offers the multiple grouping [6.d], while at the same time, it can be already considered for the SSB use case [6.e]. More details about this evaluation can be found in the Apendix B.

(((Use hdfs for archival? [6.f])))

5.7 Aggregation

The goal of the aggregation layer is to combine several metrics and produce new ones. This can be for instance the combination of the status of several tests to see if a particular service is working, or the combination of the status of different services to see if a site has at least one working instance of a service. The output of the aggregation layer should be feed back into the system as a new metric. This way, it can reuse the same visualization layer as the basic metrics, 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 doing in the SSB with their 'Site readiness' exercises. The goal now is that the WLCG monitoring group will offer some aggregations that will create the site availabilities and reliabilities [7.a].

5.8 Visualization

For the visualization, it was encouraged to have a separation between the server side and the client side [8.b]. 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 [8.c].

The client part would be composed of javascript libraries, following a Model-View-Controller paradigm [8.d]. Some monitoring applications already use xbrowse for this (an MVC framework built on top of jquery). For plots, highcharts is the obvious choice.

In parallel with that direction, the Agile Monitoring representative recommended using Kibana [8.e], a web frontend for Elasticsearch.

6 Next steps

Now that the functionality that is needed has been identified, and that the design has also been

These are the recommended actions that have been identified by the group:

1. Application level:
    a. Stop MyWLCG job trends
    b. Stop CMS Datasets
    c. Stop ALICE FTD Efficiency
    d. Stop MyWLCG FTS Transfers
    e. Stop T0/T1 SAM Siteview
    f. Stop SAM Tree Map
2. Deployment level:
    a. Migrate all the hosts used by the group to the Agile Infrastructure (Puppet, Foreman, Hiera)
    b. All the machines maintained by IT-SDC should be ported to SLC6.
    c. Use standard building tools like the CERN Koji and Bamboo instead of the Dashboard Build System and the EGI Koji.
    d. Use the CERN provided git repository
3. Documentation
    a. New user/service documentation should be done in a standard tool (wiki??)
    b. Combine the GGUS support units
    c. Combine the snow support units
    d. Migrate the Dashboard savannah to Jira
    e. Create a Jira category 'WLCG Monitoring'
    f. Create Jira projects for the different activities, including also a Jira project for the common framework.
4. Probes
    a. Nagios becomes an optional component. Each VO can decide if they want to keep it.
    b. Provide a CONDORG probe
    c. Provide a CREAM-CE probe
    d. Provide a mechanism for direct publication of metrics, like hammercloud or experiment specific metrics.
5. Transport
    a. Include the latest libraries provided by the messaging team
    b. Stay in touch with the latest development from the messaging team
    c. (Evaluate Flume?)
6. Storage
    a. Combine the schema of SAM and SSB
    b. Deploy an ElasticSearch cluster for evaluation
    c. Deploy a Hadoop Cluster
    d. Evaluate grouping with ElasticSearch 1.0, once it becomes available
    e. Integrate ElasticSearch as an alternative for the SSB schema

   f. Evaluate hdfs as an archival mechanism
7. Aggregation
   a. Implement the Availability and Reliability concepts as aggregation, feeding the output as new metrics.
8. Visualization
   a. Provide SUM-like functionality on top of the common SAM/SSB schema
   b. Divide the visualization on server side (html skeleton+json API) and client (javascript)
   c. Incorporate server side caching, with memcache or varnish
   d. Adopt an MVC approach for the client side
   e. Install Kibana in the development cluster and evaluate it

7 Transition period

The work on the new WLCG Monitoring applications should start already, creating a parallel infrastructure. This parallel infrastructure will start being a prototype of how the system will look like, and after fast-cycle iterations with the end users, should lead to a production quality system before the end of the summer of 2014. At that point, and once all the interested parts are satisfied with the new schema, the switch can be done and the old infrastructure can be decommissioned.

Having these two infrastructures implies that there will be more effort needed during that time. At the same time, it will allow a better comparison of the two systems.

It is worth pointing out that there are also several commitments of the current SAM system to EGI that should be maintained until the end of April 2014.

8 The future

Here we should put at comparison of the effort needed to maintain the structure right now, and the effort that will be needed once the new system is in place

9 References

1. https://twiki.cern.ch/twiki/bin/view/LCG/WLCGMonitoringConsolidation
2. http://wlcg-mon.cern.ch/

10 Apendix A: Usage of monitoring applications

- Used
- nice to have
- not used

Category	ALICE	ATLAS	CMS	LHCb	Comments
Job Monitoring					
Current view		Interactive View Task Monitoring	Interactive view 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

11 Apendix 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

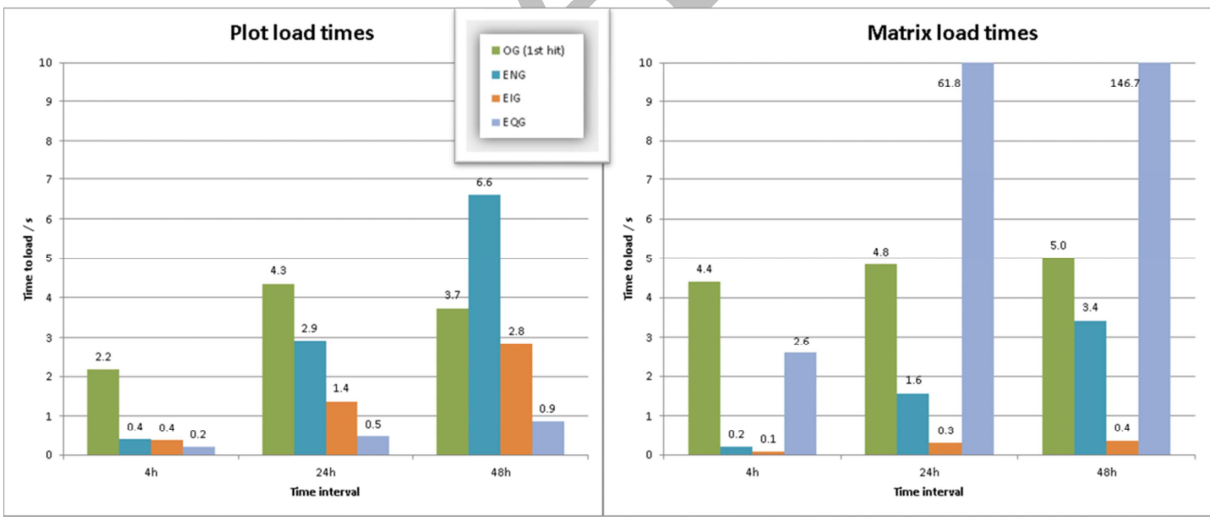


Figure 2. Comparison between Oracle 1st hit (un-cached result) and multiple grouping methods of Elasticsearch.