Migration of Gratia Active Archive Project Accounting to GRACC.

Mariano Basile, Supervisor:Kevin Retzke

Contents

Abstract

Fermilab provides a custodial active archive for customers, either on-site and not, for long-term storage of tens of petabytes of scientific data.
The archive facility at Fermilab is capable of providing access to these data over a 100 Gb/s network.
Tape storage is assumed to be the permanent means of custodial for data. Customers can also opt for a dedicated disk cache integrated with the tape storage.
The Active Archive Facility (aka AAF) is indeed an hierarchical storage system consisting of tape storage, called Enstore, with a front-end disk cache called dCache.
Customers are charged yearly according to a cost model that is based on their projected and actual usage.
Daily and monthly usage metrics that these charges are based on are available on the Customers' Active Archive web pages.
These usage metrics have been provided so far by means of the Gratia accounting system.
Because of the Gratia shutdown the new batch system accounting/monitoring service GRACC is being deployed.
My internship was aimed at helping the retire of Gratia sooner in order to reduce the operations and maintenance overhead required for its upkeep. Key responsibilities, after a two weeks training phase, have included the AAF data migration from the old accounting service and the AAF porting to GRACC.

1 *Fermilab Active Archive Facility*

1.1 *AAF Overview*

The Active Archive Facility is an hierarchical storage system consisting of tape storage, called Enstore, with a front-end disk cache called dCache.
Services are provided for both on-site direct access to files on tape, or on and off site access of files through a disk cached front end to the tape storage.

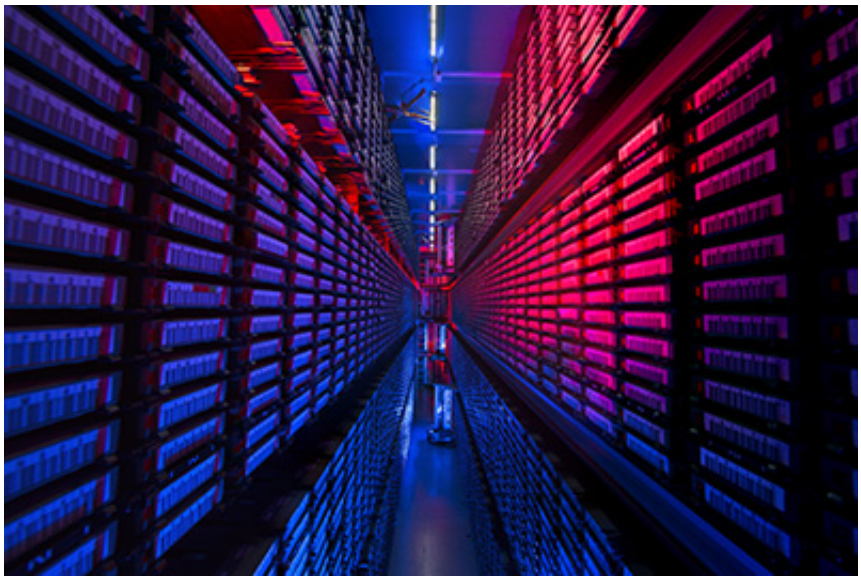


Figure 1: Fermilab Active Archive Facility

1.1.1 *The tape storage system: Enstore*

The tape based storage system has a capacity of tens of Petabytes and an aggregate I/O bandwidth of multi GB/s.
This tape storage system is the Fermilab primary data store for scientific data sets.
When a request to read or write is made to the AAF, a tape must be mounted and threaded to its load point, then a seek to the location of the file on tape is made.
The mount time depends on the load of the system – the quickest time to load is about 45 seconds. A seek from the beginning of tape takes about 50 seconds on average and a seek from one location on tape to another random location is about 33 seconds. It is best to read or write as many files as possible while the tape is mounted.
For writes, this means supply files at a high enough frequency that a number of them will be written per mount.
For read accesses, files written about the same time will generally reside on the same tape, so accessing multiple files from around the same time will yield the best performance.

1.1.2 *The disk cache system: dCache*

dCache is a disk caching software developed by Fermilab. It can be either used independently for high performance volatile storage, or in conjunction with Enstore as the high speed front end of a tape backed hierarchical storage system.
In the latter use, dCache decouples the low latency, high speed disk access over the network from the high latency sequential access of tapes.
The cache provides high performance access to frequently accessed files.
Whether the file already exists in the disk cache, or needs to be first retrieved from tape is transparent to the user.
Fermilab dCache systems use raided disk in redundant configurations to reliably store users' files.
Files in dCache can be accessed with several different protocols.
Local users can access data through kerberized FTP, GridFTP, Webdav, and NFSV4.1.
Users needing to access files on tape from off-site computers must do so through dCache.

1.2 *Usage Information and Cost*

The Active Archive Facility (aka AAF) service is provided through a Strategic Partnership Project (SPP) agreement between the customer and Fermilab.
This agreement consists of a set of Terms and Conditions, and a Statement of Work which specifies the expected usage by the customer, estimated charges for each year, and agreed responsibilities of each party.
Customers are charged each year in advance.Yearly storage charges are based on several quantities:

- The total accumulated amount of tape storage used at the end of the year (in the order of Petabytes).
- Amount of data read/written from/to disk;
- Amount of data read/written from/to tape;
- No. of tape mounts and tape drive hours used;

Tape drive-hours are measured from when a tape gets mounted in a drive, transfers to/from the tape are completed, to when the tape dismounted.
At the beginning of the each billing year, the customer supplies information on the estimated storage and drive utilization they expect to use, and are billed in advance for the year.
If funds run out during the year, more must be supplied to be able to store additional data.
If there are leftover funds at the end of the year, they are carried over into the next year.
Each customer can monitor its own usage by means of active archive web pages.
These .html pages are updated daily, and history of monthly summaries are available.
Different time scales are used when presenting data to customers: week to date, past seven days, month to date, last thirty days, from start of contract (2015-01-01).
AAF usage metrics regarding the amount of data read/written from/to disks and no. of tape mounts and tape drive hours used are also presented in a time series fashion.
The following is an example of daily usage report on 22th Septmember:

Current Usage Summary (updated daily)

Data as of: Fri Sep 22 01:00:01 2017

Total storage on tape: 1,902,515.95 GB

	Week to Date	Past Seven Days	Month to Date	Past 30 Days	From 1/1/2015
GB written to disk	10,173.57	29,121.35	106,063.85	149,705.32	4,206,511.97
GB read from disk	31,034.77	97,671.42	421,372.24	693,225.98	34,460,887.35
GB written to tape	6,673.86	7,436.95	24,813.77	39,845.23	1,251,005.88
GB read from tape	515.67	633.79	16,074.81	41,328.57	1,044,252.80
Tape drive hours used	22	29	176	391	13,257
No. of tape mounts	84	140	740	1,651	75,976

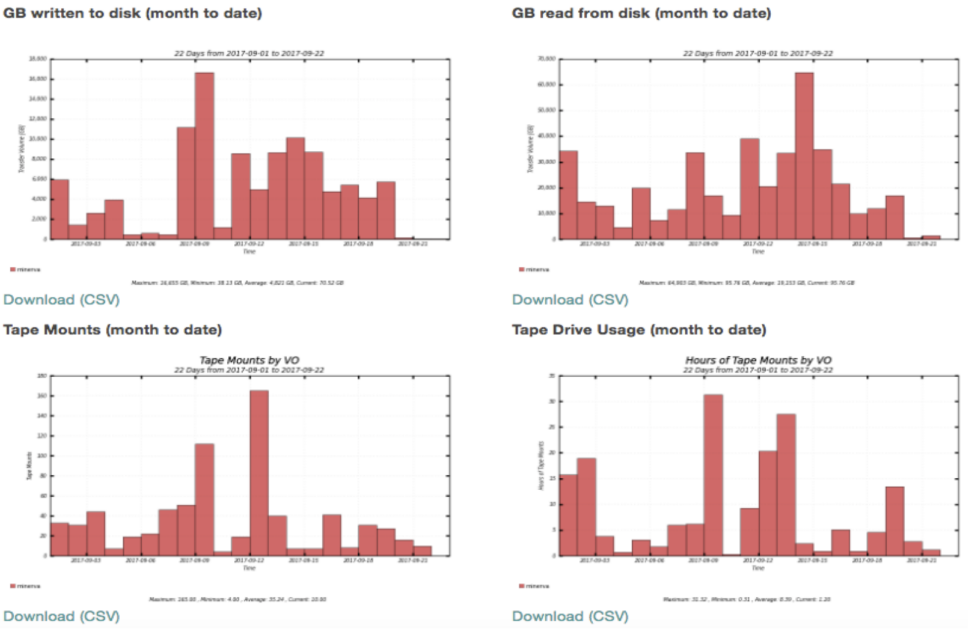


Figure 2: Example of daily usage metrics on the 22th September for Minerva experiment

Monthly usage metrics are provided both in the form of html pages, for last month statistics, and in PDF format for other months dating back in time.

Previous Month's Usage Summary

Data as of: Fri Sep 1 00:00:00 2017

	Week to Date	Past Seven Days	Month to Date	Past 30 Days	From Start of Contract
GB on tape (total)	0	0	0	0	1,879,866.86
GB written to AAF	22,638.26	40,029.56	80,892.38	79,502.76	4,100,448.11
GB read from AAF	87,921.64	173,743.48	624,754.94	610,025.07	34,039,515.11
GB written to tape	9,647.62	13,812.70	22,364.17	22,210.32	1,226,192.10
GB read from tape	2,930.24	5,519.41	65,313.76	60,838.75	1,028,177.99
Tape drive hours used	75	129	422	406	13,079
No. of tape mounts	444	647	2,032	1,984	75,218

Monthly History

minerva-2017-08
minerva-2017-07
minerva-2017-06
minerva-2017-05
minerva-2017-04
minerva-2017-02
minerva-2017-01
minerva-2016-12
minerva-2016-11
minerva-2016-10

Figure 3: Example of monthly usage metrics for August 2017 (and back) for Minerva experiment

2 *The need of a migration*

What we have not said so far is how the usage metrics are obtained. This task has been accomplished so far by means of the GRATIA accounting system. The GRATIA system has been developed over a decade ago as an accounting system for batch systems and Linux process at FNAL. Starting from 2007 Gratia has been adopted

by the Open Science Grid (OSG) as a distributed, grid-wide accounting system. At that point GRATIA has started showing its limitation:

- Due to the evolution in the scientific computing the requirements regarding what types of usage should be tracked, and what information should be stored have changed over time. Because of an inflexible record models and storage GRATIA has struggled to keep up with the changes.
- Additionally, the GRATIA service architecture has struggled to scale as the OSG usage and hence record rate has increased.
- Also the user interface and visualization tools have fallen behind the state- of-the-art, due to large effort being required to build interfaces with the Gratia system.

For these reasons and more in the early 2016 OSG and Fermilab decided to investigate re-designing GRATIA to provide a more flexible architecture, data storage format and easier integration with open-source data exploration and visualization tools.
The investigation settled on a microservice-based architecture called GRACC (Grid ACCounting).
Since the GRATIA shutdown, (also) AAF was required to be migrated to GRACC. In order to understand how the AAF migration to GRACC has been realized it is worth describing very briefly both the old accounting system (GRATIA) and the new accounting/monitoring system GRACC on which I have focused during the first two weeks of my internship.

3 *GRATIA*

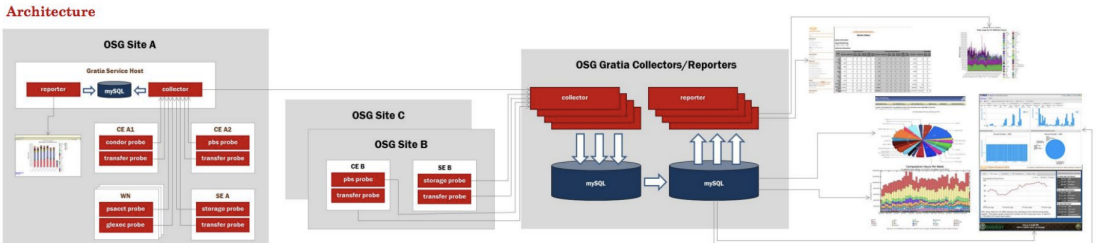


Figure 4: Gratia Architecture

3.1 *Gratia Probes*

The Gratia infrastructure is designed to be hierarchical. On the lowest level "*probes*" running on remote sites look at a service or resource.Probes usually consist of:

- A python script (the probe itself)
- A probeConfigfile
- A cron job which schedules the probe's execution

Probes send collected data to an accumulator, called a "*collector*" specified inside the probeConf file by using the Gratia API. Collected data include information about: batch jobs and glide-in jobs, grid transfers, storage usage and allocation, cloud accounting, grid services availability, etc.

3.2 *Gratia Collectors & Reporters*

A collector receives usage record information either directly from probes or via another collector. A collector stores the received raw records and creates daily summaries of them to keep long term and to be used for interactive displays.
After some data validation it stores the information in permanent storage which is a MySQL database.
Gratia supports hierarchical collectors' structure and permits forwarding and filtering between collectors.
Typical use of this hierarchy is for a site to have the probes looking at its resource reporting the information to a local collector and to have the local collector push the data up to the central Grid collector.
The information is passed from one component to the other using a record exchange format defined to facilitate information sharing among grid sites.
Often collocated with a collector there's a reporter, which gives access to the data through a web user interface.

4 *GRACC: New Generation of the OSG Accounting*

GRATIA limitations described in section *2* steered the GRACC investigation towards a loosely-coupled architecture composed of small set of existing technologies connected through a message exchange with a flexible data schema and interchange format.
Furthermore, based on existing technical experience within the OSG and at Fermilab and elsewhere, the choice was clear to utilize several key components: Elasticsearch as a data storage and query platform, RabbitMQ for data exchange, and Grafana for the primary user interface, supplemented by Kibana for ad-hoc analytics.

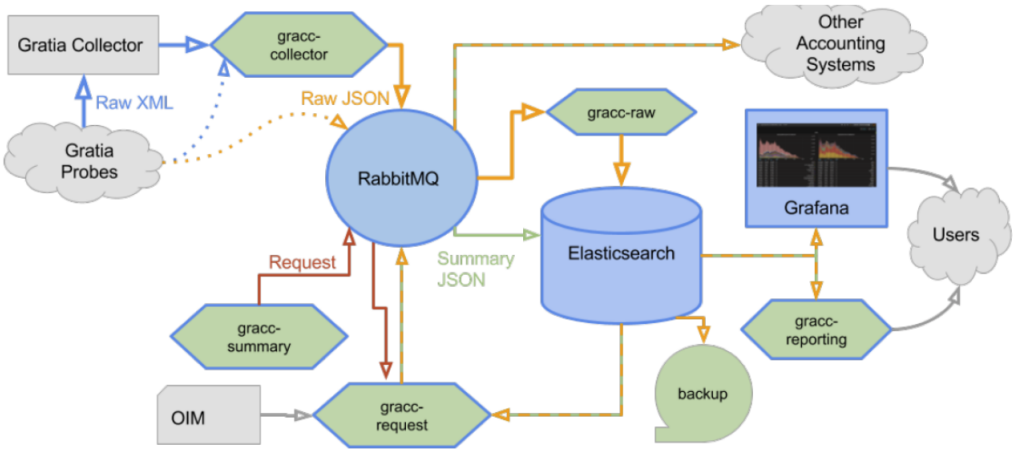


Figure 5: Diagram of the GRACC microservice architecture. Included are the existing Gratia probes and collectors, GRACC services, inter-process message passing, record storage, and visualization.

4.1 *Record Collection*

Origin usage record generation for GRACC is handled by the existing Gratia probes.
It is envisioned that in the future the probes will be updated to send records directly to the GRACC message exchange.
For the near-term, the gracc-collector service was developed as a transitory endpoint that is compatible with existing Gratia collectors and probes.

4.2 *Communication*

All inter-process communication is handled through a RabbitMQ broker, which was chosen over the several major competing message systems that would meet the needs of GRACC primarily because the OSG had an existing deployment at its Grid Operations Center.

4.3 *Record Processing & Summarization*

Raw usage records that are sent to the raw RabbitMQ exchange have to be stored in the Elasticsearch database, after undergoing some processing. Summary records undergo similar processing. *Logstash* is used for both, as its pipeline architecture and large plugin library allows simple configuration of record processing as well as robust communication with both RabbitMQ and Elasticsearch in addition to any other components that may be included in GRACC in the future.

4.4 *Record Storage*

Elasticsearch is used for record storage, as it provides a fault-tolerant distributed architecture, flexible schema, partitionable indices, and powerful search and aggregations.
Each stream of raw records is stored in a time based index pattern, where each index stores the records for one month. Likewise summary records are stored in yearly indices.
The use of time-based indices allows for faster queries since queries only have to be run against the indices that span the time range of interest.

4.5 *Interactive Visualization*

The primary visualization interface is Grafana, an open-source monitoring dashboard application, which supports a rich ecosystem of data sources and graph plugins, and provides a powerful interface to create and share dashboards.
Kibana is also made available for in-depth analytics and data exploration.

5 *Migration of Active Archive Project to GRACC*

5.1 *AAF in a nutshell*

Once both GRATIA and GRACC have been analyzed I've started analyzing the active archive facility. An overview diagram of the components of this system is provided in the following figure:

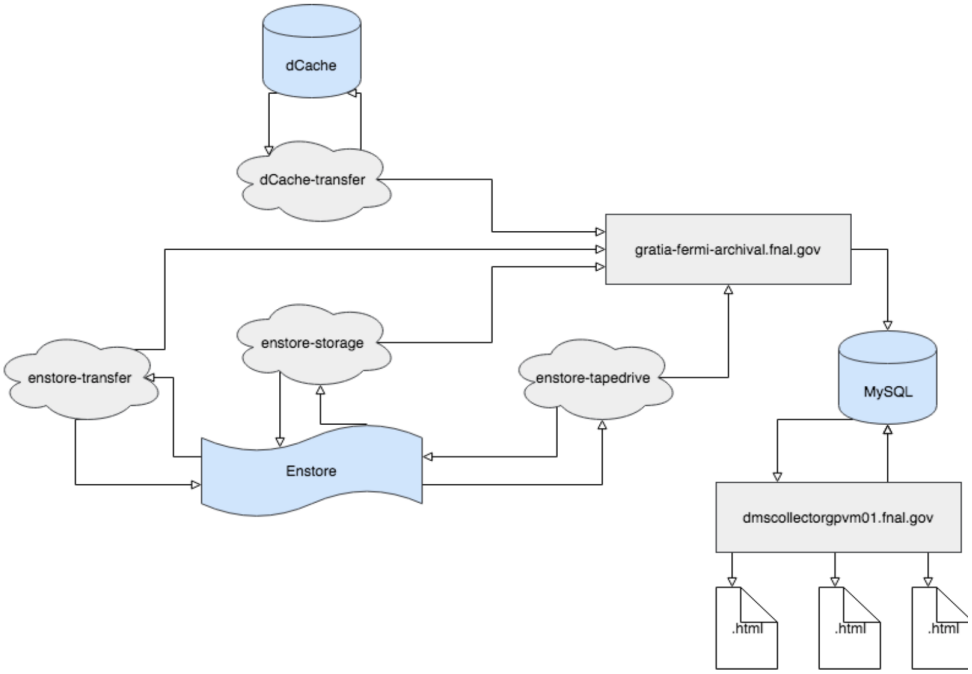


Figure 6: Active Archive Facility Architecture

There are four main sets of GRATIA probes:

- *dCache-transfer*: collect transfer information from/to the dCache system(either reads or writes).
- *enstore-transfer*: collect transfer information from/to the Enstore system (either reads or writes).
- *enstore-storage*: collect information about the amount of data stored in the Enstore system (data stored on tapes).
- *enstore-tapedrive*: collect information about the no. of tape mounts and tape drive hours used.

Once data have been collected by probes, data are pushed into the Gratia Collector *"gratia-fermi-archival.fnal.gov"*.

As every collector after some data validation it stores the information in permanent storage which is a MySQL database (*gratiadb03.fnal.gov*).
On another machine *dmscollectorgpvm01.fnal.gov* different cronjobs periodically trigger the execution of some python scripts that fetch the data from the MySQL database and generate the customers' active archive web pages.
These scripts are named "*AAFReportGenerator.py*" and "*AAFHTMLReport.py*".

5.2 *Dev Environment Setup*

Before proceeding with the data migration, as it is common in such cases, a dev-environment has been set up. It consists of a remote Fermicloud virtual machine running Scientific Linux (release 7.3 Nitrogen).
The complete GRACC stack was required at that point. To this aim I've used docker (v1.12.6) with which I've started facing up also during my second internship week.

5.3 *Data Migration*

The MySQL database in which data are stored by the collector and from which data are retrieved by the python scripts is shown in figure 7:

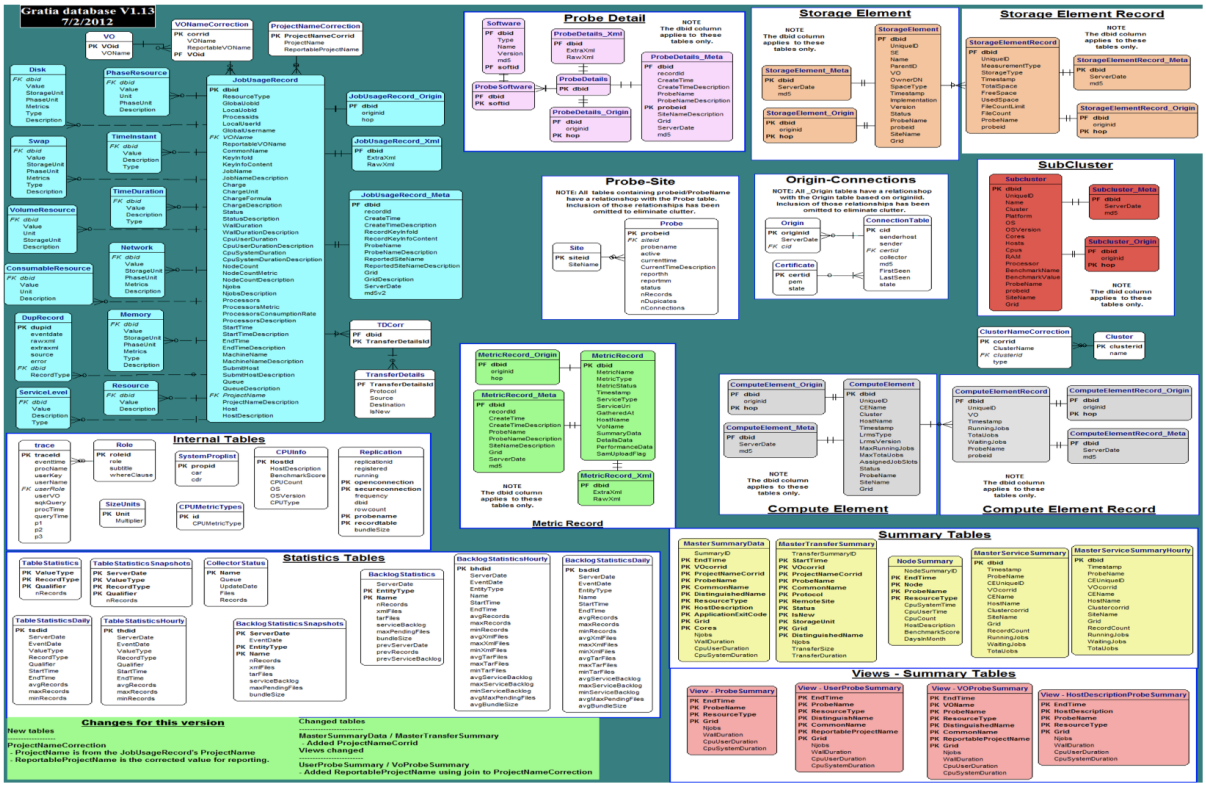
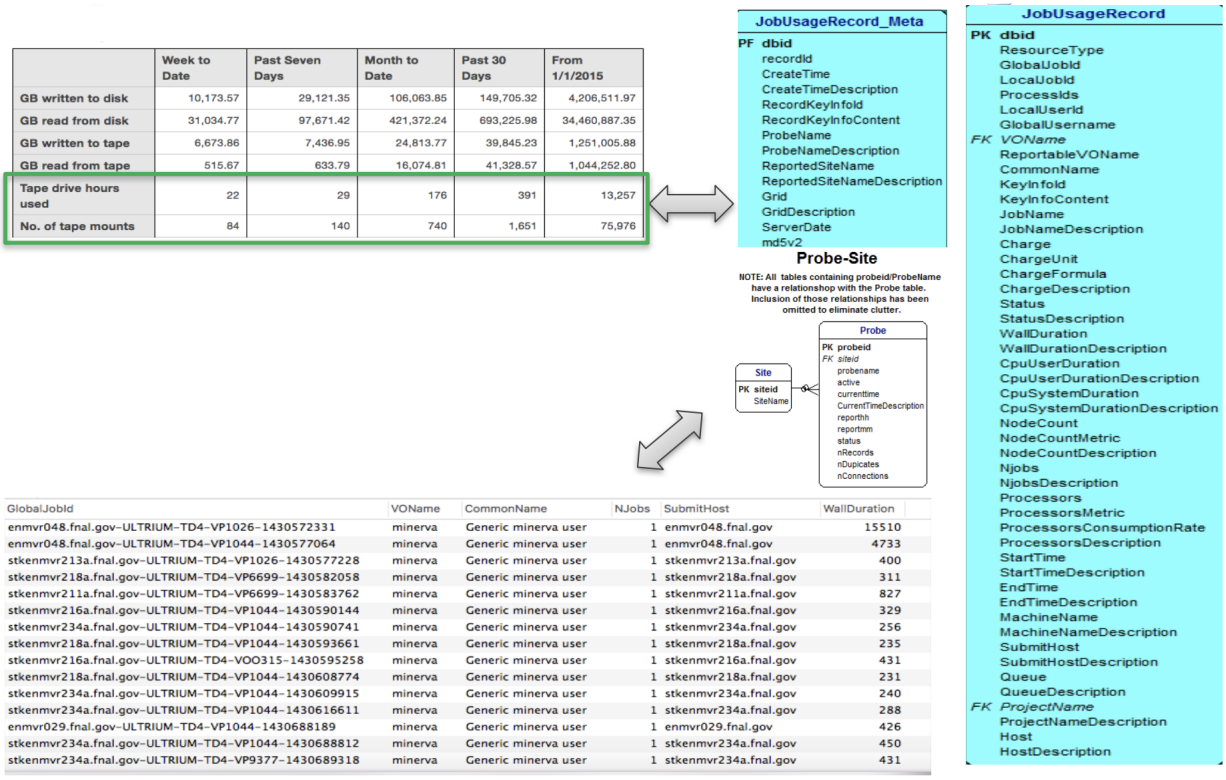


Figure 7: Gratia AAF Schema

It was necessary to find out which records were fetched in order to proceed with the records migration. Python code defined in "*AAFHTMLReport.py*" has been inspected for this aim.
The involved tables containing the records to be migrated are shown in the following sections.

5.3.1 *How to: Obtain AAF Usage Metric regarding the no.of tape mounts and tape drive hours used*



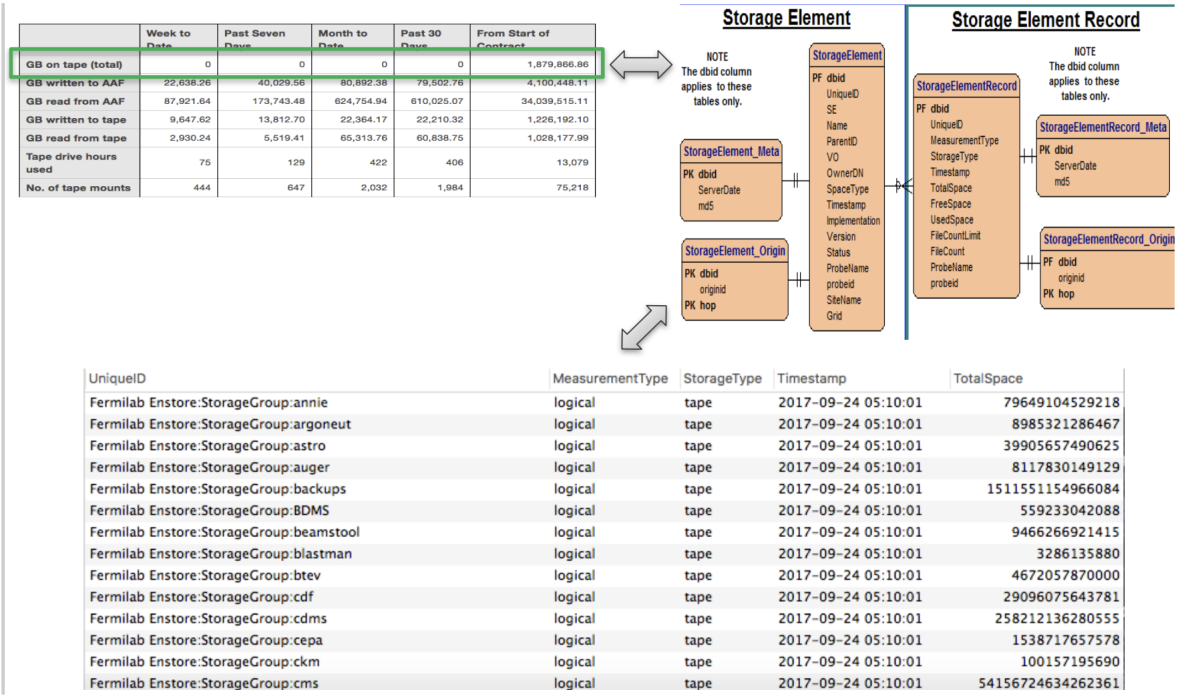
	Week to Date	Past Seven Days	Month to Date	Past 30 Days	From 1/1/2015
GB written to disk	10,173.57	29,121.35	106,063.85	149,705.32	4,206,511.97
GB read from disk	31,034.77	97,671.42	421,372.24	693,225.98	34,460,887.35
GB written to tape	6,673.86	7,436.95	24,813.77	39,845.23	1,251,005.88
GB read from tape	515.67	633.79	16,074.81	41,328.57	1,044,252.80
Tape drive hours used	22	29	176	391	13,257
No. of tape mounts	84	140	740	1,651	75,976

GlobalJobId	VOName	CommonName	NJobs	SubmitHost	WallDuration
enmvr048.fnal.gov-ULTRIUM-TD4-VP1026-1430572331	minerva	Generic minerva user	1	enmvr048.fnal.gov	15510
enmvr048.fnal.gov-ULTRIUM-TD4-VP1044-1430577064	minerva	Generic minerva user	1	enmvr048.fnal.gov	4733
stkenmvr213a.fnal.gov-ULTRIUM-TD4-VP1026-1430577228	minerva	Generic minerva user	1	stkenmvr213a.fnal.gov	400
stkenmvr218a.fnal.gov-ULTRIUM-TD4-VP6699-1430582058	minerva	Generic minerva user	1	stkenmvr218a.fnal.gov	311
stkenmvr211a.fnal.gov-ULTRIUM-TD4-VP6699-1430583762	minerva	Generic minerva user	1	stkenmvr211a.fnal.gov	827
stkenmvr216a.fnal.gov-ULTRIUM-TD4-VP1044-1430590144	minerva	Generic minerva user	1	stkenmvr216a.fnal.gov	329
stkenmvr234a.fnal.gov-ULTRIUM-TD4-VP1044-1430590741	minerva	Generic minerva user	1	stkenmvr234a.fnal.gov	256
stkenmvr218a.fnal.gov-ULTRIUM-TD4-VP1044-1430593661	minerva	Generic minerva user	1	stkenmvr218a.fnal.gov	235
stkenmvr216a.fnal.gov-ULTRIUM-TD4-VOO315-1430595258	minerva	Generic minerva user	1	stkenmvr216a.fnal.gov	431
stkenmvr218a.fnal.gov-ULTRIUM-TD4-VP1044-1430608774	minerva	Generic minerva user	1	stkenmvr218a.fnal.gov	231
stkenmvr234a.fnal.gov-ULTRIUM-TD4-VP1044-1430609915	minerva	Generic minerva user	1	stkenmvr234a.fnal.gov	240
stkenmvr234a.fnal.gov-ULTRIUM-TD4-VP1044-1430616611	minerva	Generic minerva user	1	stkenmvr234a.fnal.gov	288
enmvr029.fnal.gov-ULTRIUM-TD4-VP1044-1430688189	minerva	Generic minerva user	1	enmvr029.fnal.gov	426
stkenmvr234a.fnal.gov-ULTRIUM-TD4-VP1044-1430688812	minerva	Generic minerva user	1	stkenmvr234a.fnal.gov	450
stkenmvr234a.fnal.gov-ULTRIUM-TD4-VP9377-1430689318	minerva	Generic minerva user	1	stkenmvr234a.fnal.gov	431

Figure 8: Records required to be migrated for AAF usage metrics: *No.of tape mounts and tape drive hours used*

AAF usage metrics regarding the *no.of tape mounts and tape drive hours used* are obtained by querying the *JobUsageRecord, JobUsageRecord_Meta and ProbeName* tables. Records inside these tables are then required to be migrated.

5.3.2 *How to: Obtain AAF Usage Metric regarding the total amount of data on tapes*



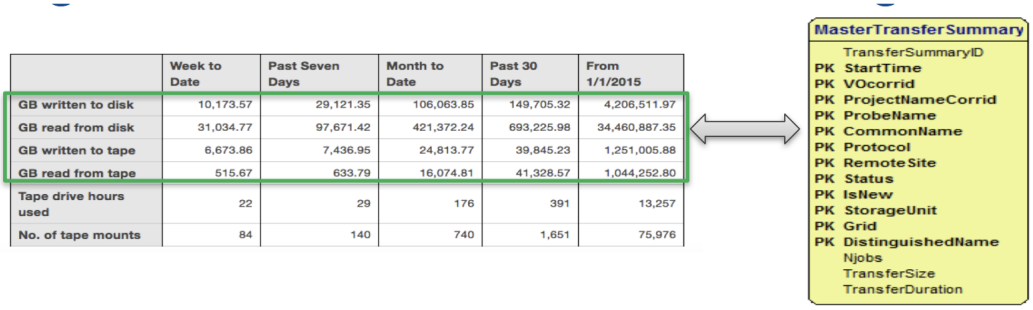
	Week to Date	Past Seven Days	Month to Date	Past 30 Days	From Start of Contract
GB on tape (total)	0	0	0	0	1,879,866.86
GB written to AAF	22,638.26	40,029.56	80,892.38	79,502.76	4,100,448.11
GB read from AAF	87,921.64	173,743.48	624,754.94	610,025.07	34,039,515.11
GB written to tape	9,647.62	13,812.70	22,364.17	22,210.32	1,226,192.10
GB read from tape	2,930.24	5,519.41	65,313.76	60,838.75	1,028,177.99
Tape drive hours used	75	129	422	406	13,079
No. of tape mounts	444	647	2,032	1,984	75,218

UniqueID	MeasurementType	StorageType	Timestamp	TotalSpace
Fermilab Enstore:StorageGroup:annie	logical	tape	2017-09-24 05:10:01	79649104529218
Fermilab Enstore:StorageGroup:argoneut	logical	tape	2017-09-24 05:10:01	8985321286467
Fermilab Enstore:StorageGroup:astro	logical	tape	2017-09-24 05:10:01	39905657490625
Fermilab Enstore:StorageGroup:auger	logical	tape	2017-09-24 05:10:01	8117830149129
Fermilab Enstore:StorageGroup:backups	logical	tape	2017-09-24 05:10:01	1511551154966084
Fermilab Enstore:StorageGroup:BDMS	logical	tape	2017-09-24 05:10:01	559233042088
Fermilab Enstore:StorageGroup:beamstool	logical	tape	2017-09-24 05:10:01	9466266921415
Fermilab Enstore:StorageGroup:blastman	logical	tape	2017-09-24 05:10:01	3286135880
Fermilab Enstore:StorageGroup:btev	logical	tape	2017-09-24 05:10:01	4672057870000
Fermilab Enstore:StorageGroup:cdf	logical	tape	2017-09-24 05:10:01	29096075643781
Fermilab Enstore:StorageGroup:cdms	logical	tape	2017-09-24 05:10:01	258212136280555
Fermilab Enstore:StorageGroup:cepa	logical	tape	2017-09-24 05:10:01	1538717657578
Fermilab Enstore:StorageGroup:ckm	logical	tape	2017-09-24 05:10:01	100157195690
Fermilab Enstore:StorageGroup:cms	logical	tape	2017-09-24 05:10:01	54156724634262361

Figure 9: Records required to be migrated for AAF usage metrics: *Total amount of data on tapes*

AAF usage metrics regarding the *total amount of data on tapes* are obtained by querying the *StorageElement and StorageElementRecord* tables. Records inside these tables are required to be migrated too.

5.3.3 *How to: Obtain AAF Usage Metric regarding the amount of data read/written from/to disks/tapes*



	Week to Date	Past Seven Days	Month to Date	Past 30 Days	From 1/1/2015
GB written to disk	10,173.57	29,121.35	106,063.85	149,705.32	4,206,511.97
GB read from disk	31,034.77	97,671.42	421,372.24	693,225.98	34,460,887.35
GB written to tape	6,673.86	7,436.95	24,813.77	39,845.23	1,251,005.88
GB read from tape	515.67	633.79	16,074.81	41,328.57	1,044,252.80
Tape drive hours used	22	29	176	391	13,257
No. of tape mounts	84	140	740	1,651	75,976

Figure 10: Records required to be migrated for AAF usage metrics: *Amount of data read/written from/to disks/tapes*

AAF usage metrics regarding the *amount of data read/written from/to disks/tapes* are obtained by querying the *MasterTransferSummary* table.
MasterTransferSummary records are obtained by means of a MySQL stored procedure.
The stored procedure generates summary records from corresponding *JobUsageRecord* so that to reduces the number of records by two to three orders of magnitude.
This enable faster analytics of these records over different time periods.
Also records inside this table are required to be migrated.

5.3.4 *Migration Plan*

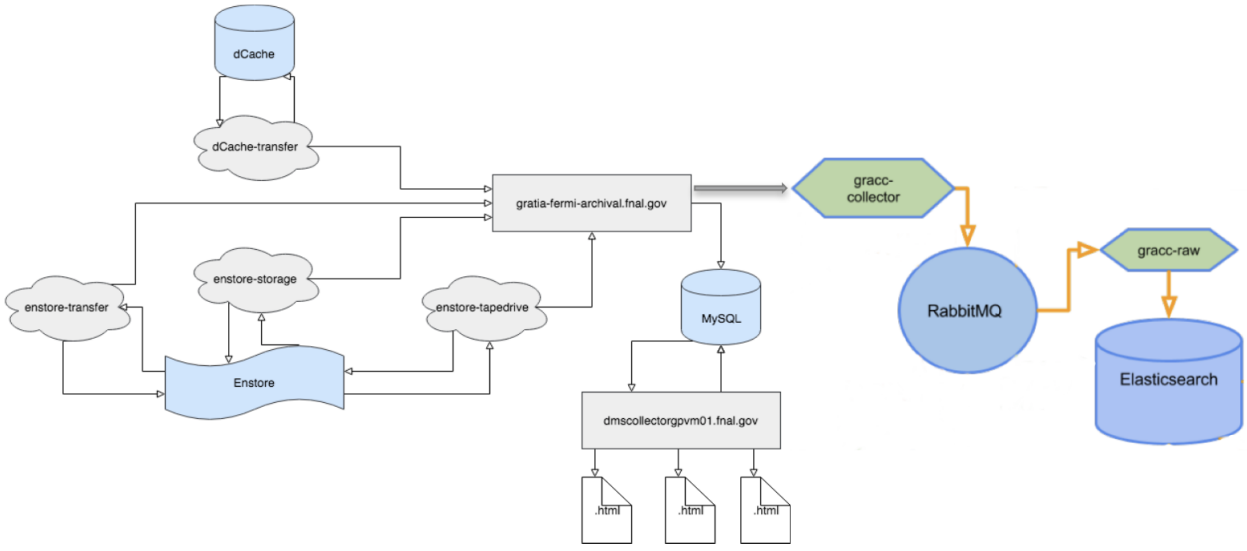


Figure 11: *Migration Plan*

As already described in section *3.2* Gratia supports hierarchical collectors' structure and permits forwarding and filtering between collectors. Moreover as then described in section *4.1* the gracc-collector service was developed as a transitory endpoint that is compatible with existing Gratia collectors and probes.
Relying on these two concepts the migration plan has been designed.

5.3.5 *JobUsageRecord, StorageElement and StorageElementRecord migration*

JobUsageRecord, StorageElement and StorageElementRecord were forwarded by:

- enabling *JobUsageRecord Replication, StorageElement Replication, StorageElementRecord Replication in the GRATIA Collector "gratia-fermi-archival.fnal.gov".*
- choosing as remote endpoint the GRACC collector running on the Fermicloud VM.
- choosing as ProbeName "*All*" to let all records being migrated.
- choosing a record dbid to start the forwarding.

Some few words have to said about records stored inside the *JobUsageRecord* table.
So far more than 1 billion records have been stored and these are either records gathered by the *enstore-tapedrive*, *dCache-transfer and enstore-transfer* probes.
Due to limitations in the GRATIA forwarding mechanism it was unfeasible migrating all these records in a reasonable time (years were required).
This was not an issue since:

- AAF usage metrics regarding the *amount of data read/written from/to disks/tapes* are obtained by means of the *MasterTransferSummary* table, i.e. MasterTransferSummary records are the ones that needed to be migrated to GRACC.
- AAF Usage Metric regarding the *no.of tape mounts and tape drive hours used* are obtained by performing aggregations on records gathered by the *enstore-tapedrive probe.* Since the resulting number of records is minimal with respect to the total, despite the limitations in the GRATIA forwarding mechanism, it was possible having the whole set of them in GRACC by simply defined another ProbeName "*enstore-tapedrive:fndca2a.fnal.gov*".

Once records have been received by the GRACC Collector they are transformed from XML into a more flexible "flattened" JSON representation and then published to a standard RabbitMQ exchange for further processing and storage.
At that point the gracc-stash-raw is in charge of retrieving data from the RabbitMQ exchange and indexing data inside inside Elasticsearch.
Some modification were required to the original implementation of the gracc-stash-raw component in order to treat differently *JobUsageRecord and StorageElement,StorageElementRecord.*

5.3.6 *MasterTransferSummary record migration*

Only *MasterTransferSummary* record were missing at that time.
This was due to the fact that the GRATIA forwarding mechanism doesn't provides any implementation for replicating *MasterTransferSummary* records.
At that time Logstash, which I've started learning during the two weeks training phase, has been taken into account.
Logstash is basically an open source, server-side data processing pipeline that ingests data from a multitude of sources, transforms it, and then sends it to a favorite "stash".

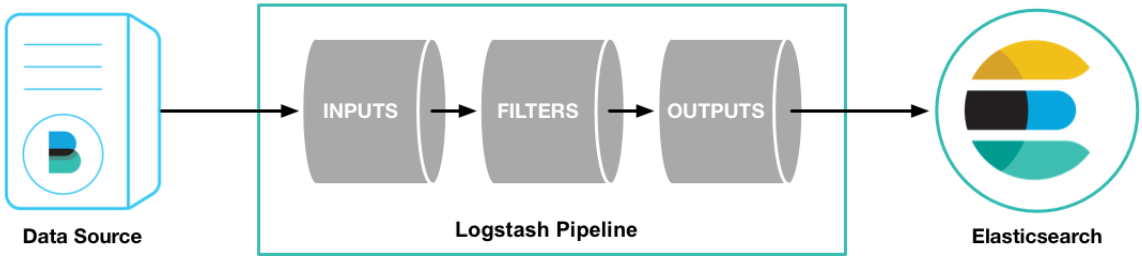


Figure 12: *Logstash Architecture*

Logstash is based on the concept of *pipeline.* A Logstash pipeline has two required elements, input and output, and one optional element, filter. The input plugins consume data from a source, the filter plugins modify the data and the output plugins write the data to a destination.
What I've done was creating a Logstash configuration pipeline that uses the *jdbc* input plugin to fetch and perform aggregations on *MasterTransferSummary* records from the *MasterTransferSummary* table and uses the

Elasticsearch output plugin to index *MasterTransferSummary* records in our favorite stash which for GRACC is (of course) Elasticsearch.

```
input {
    jdbc {
        jdbc_driver_library => "/home/aafgratia/mysql-connector-java-5.1.43/
        mysql-connector-java-5.1.43-bin.jar"
        jdbc_driver_class => "com.mysql.jdbc.Driver"
        jdbc_connection_string => "jdbc:mysql://gratiadb03.fnal.gov:3306/
        gratia_aaf"
        jdbc_user => "xxxxx"
        jdbc_password => "xxxxx"
        statement_filepath => '/home/aafgratia/logstash-5.5.1/
        gracc-aaf-transfer-summary/gratia-mts-to-gracc.sql'
        lowercase_column_names => false
        type => "JobUsageRecordSummary"
    }
}
filter {
    mutate {
        convert => {"Status" => "integer"}
    }
}
output {
   #stdout { codec => "rubydebug" }
    elasticsearch {
        hosts => [ "localhost:9200" ]
        index => "gracc.aaf-transfer.summary4-nooim"
        document_id => "%{SummaryID}"
    template_name => "gracc-summary-template"
        template => "/home/aafgratia/logstash-5.5.1/gracc-aaf-transfer-summary/
        gracc-summary-template.json"
        manage_template => true
        template_overwrite => true
    }
}
```

Figure 13: *Skeleton of the Logstash configuration pipeline*

5.4 *AAF Porting to GRACC*

Once all data have been stored inside Elasticsearch it has been necessary to implement the AAF's porting to GRACC. AAF is basically implemented as a set of python scripts along with a configuration file.
The porting has mainly involved the modification of the two scripts "*AAFReportGenerator.py*", "*AAFHTMLReport.py*", the development of a python utility class "*ElasticsearchUtils.py*" and of a bash script "*find_differences_gratia_vs_gracc.sh*" which aims to check that no differences are present between the active archive customers' html pages generated by GRATIA and GRACC.

5.4.1 Active Archive Customers' Html Pages Comparison

An initial comparison via the bash script has shown that GRACC's reports may differ wrt to Gratia's ones, as far as the AAF metrics regarding the *amount of data read/written from/to disk/tape* was concerned.
Issue was due to the fact that in Gratia the *"TransferSize" field* in the *MasterTransferSummary* table, which is used to account the total amount of data read/written from/to disk/tape, is of type "double" whereas in GRACC was of type "float" because no template was defined at time of migration. The solution was to explicitely define the schema mapping and apply it in the output filter of the *logstash.conf* file (see Figure *13*).

6 *GRACC SUMMARIZATION*

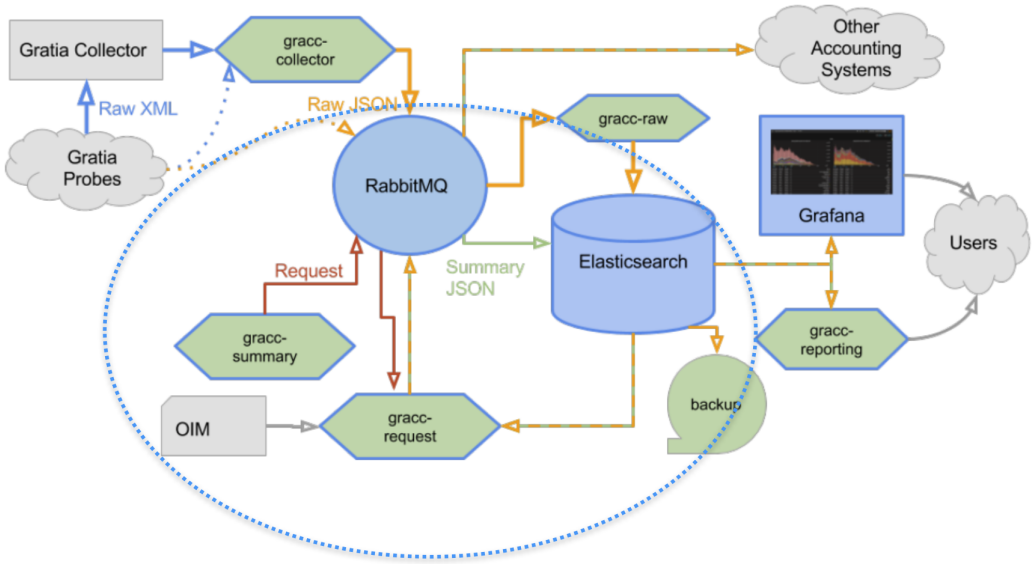


Figure 14: *Components involved in GRACC Summarization*

When *JobUsageRecord Replication* has been enabled we have let (also) raw records gathered by the *dCache-transfer and enstore-transfer* probes being forwarded to GRACC. The reason why there was the need that raw transfer records needed to be present was that in GRACC a record summarization service had to be support too (an equivalent of the *GRATIA MasterTransferSummary* stored procedure).
Record summarization is performed in GRACC by a process, the *gracc-request* a*gent*, that listens for requests, made periodically by the *gracc-summary agent,* that specify the time period to summarize and the RabbitMQ exchange to send the summarized records to.
Summarization is done via aggregation query to Elasticsearch, formatting the results into summary records, performing name corrections/mappings and further enriching the records by looking up corresponding data in OIM. Another stash, the *gracc-stash-summary.transfer* at that point starts retrieving data from the RabbitMQ exchange and indexing summary records inside Elasticsearch.

6.1 *Dev Environment Setup*

Initially in order to perform name corrections, VONames corrections have been migrated to GRACC by using logstash. Then all the components involved in the GRACC Summarization, i.e the *gracc-request agent*, the *gracc-summary agent* and the *gracc-stash-summary.transfer* have been deployed with docker.

6.2 Raw (transfer) record comparison

In order to verify that the GRACC summarization procedure outputs, when the same time interval is passed, the same *MasterTransferSummary record* migrated out of GRATIA the first step was to verify that the same set of raw (transfer) records to be summarized are present in GRACC.

Raw records forwarding has been enabled on the *gratia-fermi-archival.fnal.gov* collector on 26th August 2017. The comparison has been carried out by means of a bunch of on purpose developed python scripts. The results are shown below:

Figure 15: *Raw (transfer) record comparison from 2017-08-26 to 2017-08-27*

As it possible to see there'a deficit consisting of 11 eleven raw records. For each missing record the script also presents the time at which the transfer has begun. Querying GRATIA with these StartTimes it has been found out that the deficit actually involves:

- 10 records that have a value for the StartTime field which is greater than the one specified at replication init time but whose record dbid is smaller (and that was the reason why records were not forwarded).This is possible due to the fact that when the GRATIA collector stores the data into the MySQL database no records ordering guarantee is provided.
- Plus the record with the dbid specified at replication init time itself.

Figure 16: *Raw (transfer) record comparison from 2017-08-27*

No difference has been noticed in all the other cases.
We will present later on, which solution has been adopted because of the record missing in GRACC on 26th August.

6.3 Python summary scripts: Modification Required

The GRACC python summary scripts were already developed. By the way different modification were required prior to obtaining a fully functional GRACC summarization procedure. In particular:

- We needed to have consistent aggregations both in the MySQL SELECT used in the jdbc logstash input plugin used to migrate *MasterTransferSummary* records and in the *gracc-request agent (*in the *transfer_summary.py* script) which is in charge of generating GRACC transfers summaries.
- We also needed to have summary records (datetime fields inside these records) coming out of logstash be indexed in GRACC according to UTC date format.
- Since the time period to summarize is splitted into weeks (in order to avoid overloading the ES cluster) refinements were required in the *gracc-request agent (*in the *transfer_summary.py* script*)* and in the *gracc-summary agent* (in the *summarize.py* script*)* to avoid summarizing twice the last day of each week.

6.3.1 GRATIA vs GRACC Summarization: Comparison

It has been defined as time period to summarize the time range from 27th August 2017 to 22th Septmember 2017.
What we were expecting is that for each day in this time range the number of transfers either for dCache and Enstore, either for read(TransferType:0) and write(TransferType:1) transfers were the same.
Furthermore, we were also expecting that for each day we end up having the same total amount of data transferred. As it possible to see everything has been confirmed:

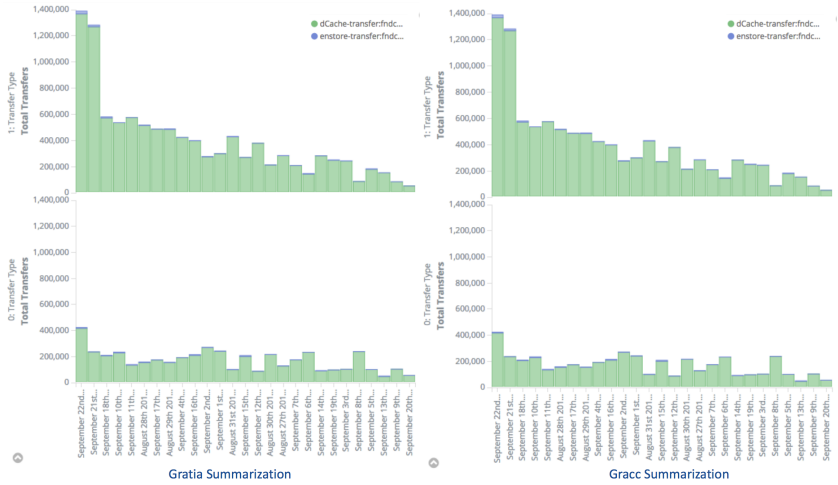


Figure 17: *GRATIA vs GRACC Summarization: Total Transfer Comparison*

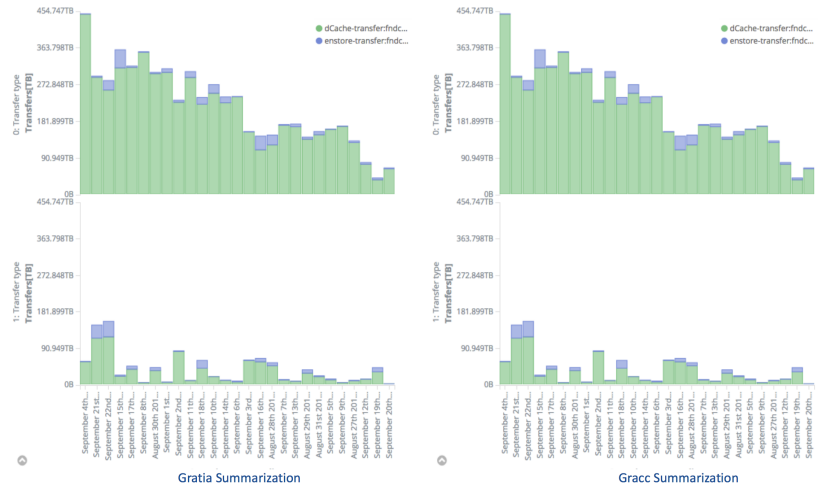


Figure 18: *GRATIA vs GRACC Summarization: Total Amount of Data Transferred*

7 *Generating GRACC* Active Archive Customers' Web Pages

What the GRACC Active Archive Customers' html pages were supposed to output is zero-difference with respect to what the GRATIA reports actually output.
Since forwarding was enabled only on the 26th of August GRACC transfer summaries were only available for a limited time period.
In order to generate GRACC reports and since we still had raw (transfer) record missing on 26th August one step was still necessary.

Basically what we needed at that time was a different "view" of the index being queried while retrieving transfers to/from disk/tape.
In order to do so Elasticsearch filtered aliases have been used.
The idea was to use migrated *MasterTransferSummary* records for retrieving transfers occurred from the 2015-01-01 (AAF start of contract) to 2017-08-26 (to overcome record missing issue).
Then the idea was to use GRACC summaries starting from 2017-08-27 to 2017-09-22.

```
        "add" :
         {
              "index" : "gracc.test.summary4-2017",
              "alias" : "gracc.test.summary4-alias",
              "filter" : {
                  "range" : {
                      "StartTime" : {
                          "lt" : "2017-09-23",
                          "gte": "2017-08-27"
                      }
                  }
              }
         }
     },
    { "add" :
         {
              "index" : "gracc.aaf-transfer.summary4-nooim",
              "alias" : "gracc.test.summary4-alias",
              "filter" : {
                  "range" : {
                      "@timestamp" : {
                          "gte" : "2015-01-01",
                          "lt":"2017-08-27"
```

Figure 19: *Elasticsearch Filtered Aliases*

Only at that time GRACC reports have been generated.

7.1 GRACC Reports: Final assessment

By means of the already developed bash script "*find_differences_gratia_vs_gracc.sh*" GRACC reports have been compared with the GRATIA ones.
The script in case any difference is present makes a folder inside the root folder "*differences*" named as the involved Virtual Organization together with a file containing the differences.
After the script has been carried out the content of the the root folder "*differences*" was still empty indicating that GRACC reports are accurate.

```
mbasile — aafgratia@fermicloud159:~/active_archive-gracc2/htdocs —
[aafgratia@fermicloud159 htdocs]$ ./find_differences_gratia_vs_gracc.sh
[aafgratia@fermicloud159 htdocs]$ ls -l differences/
total 0
[aafgratia@fermicloud159 htdocs]$
```

Figure 20: Content of the root folder "*differences*" after bash script execution

We can conclude presenting a sample reports comparison for an off-site customer (Simons) on the 22th September:

Current Usage Summary (updated daily)

Data as of: Fri Sep 22 00:00:00 2017

Total storage on tape: 1,704,937.66 GB

	Week to Date	Past Seven Days	Month to Date	Past 30 Days	From 1/1/2015
GB written to disk	37,175.85	54,148.55	84,450.01	84,450.22	1,272,525.77
GB read from disk	3.98	3.98	3.98	3.98	1,084,196.04
GB written to tape	38,955.17	55,945.49	88,438.16	88,438.38	1,397,135.47
GB read from tape	405.25	1,510.41	5,291.42	7,886.90	1,208,407.23
Tape drive hours used	127	208	324	329	12,770
No. of tape mounts	628	858	1,300	1,318	82,887

Gratia Report

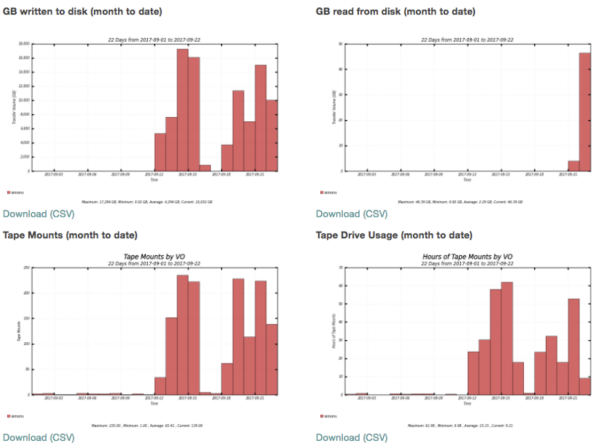
Current Usage Summary (updated daily)

Data as of: Fri Sep 22 00:00:00 2017

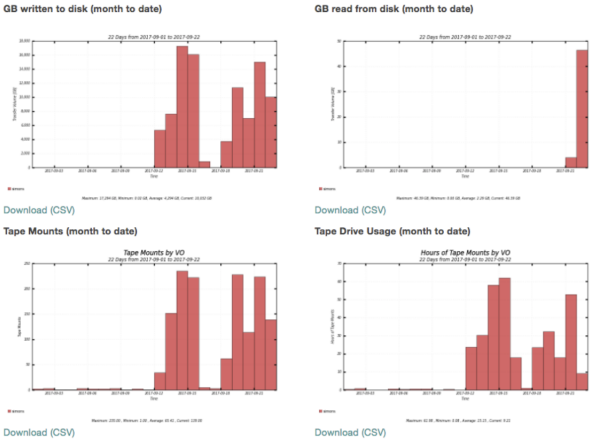
Total storage on tape: 1,704,937.66 GB

	Week to Date	Past Seven Days	Month to Date	Past 30 Days	From 1/1/2015
GB written to disk	37,175.85	54,148.55	84,450.01	84,450.22	1,272,525.77
GB read from disk	3.98	3.98	3.98	3.98	1,084,196.04
GB written to tape	38,955.17	55,945.49	88,438.16	88,438.38	1,397,135.47
GB read from tape	405.25	1,510.41	5,291.42	7,886.90	1,208,407.23
Tape drive hours used	127	208	324	329	12,770
No. of tape mounts	628	858	1,300	1,318	82,887

Gracc Report



Gratia Report



Gracc Report

Figure 21: Sample report comparison for Simons on the 22th Septmember