# Gratia: New Challenges in Grid Accounting.

**Philippe Canal**

Fermilab, Batavia, IL, USA.

pcanal@fnal.gov

**Abstract**. Gratia originated as an accounting system for batch systems and Linux process accounting. In production since 2006 at FNAL, it was adopted by the Open Science Grid as a distributed, grid-wide accounting system in 2007. Since adoption Gratia's next challenge has been to adapt to an explosive increase in data volume and to handle several new categories of accounting data. Gratia now accounts for regular grid jobs, file transfers, glide-in jobs, and the state of grid services. We show that Gratia gives access to a thorough picture of the OSG and discuss the complexity caused by newer grid techniques such as pilot jobs, job forwarding, and backfill.

## 1. Introduction

Grid infrastructures have continually developed and grown in the last few years. One important component of the Grid ecosystem is the ability to know how the resources have been used and by whom. In particular this can give the owners and sponsors of a large Grid an understanding of the successes and missed opportunities.

The Gratia system[1] is designed to be a Robust, scalable, trustable, dependable grid accounting service. In particular it has a strong emphasis on avoiding data loss rather than providing live but approximate information. It consists of many probes operating on and uploading data from remote locations to a network of one or more collector-reporting systems. This data can be about batch jobs, grid transfers, storage allocation, site availability tests or process accounting. The primary focus of the Gratia system is to provide an accounting of jobs, transfers and services executed on the Open Science Grid[2]. After outlining the Gratia Infrastructures, we describe a few of the challenges caused by novel techniques introduced in the computing Grid, including pilot jobs and backfill jobs.

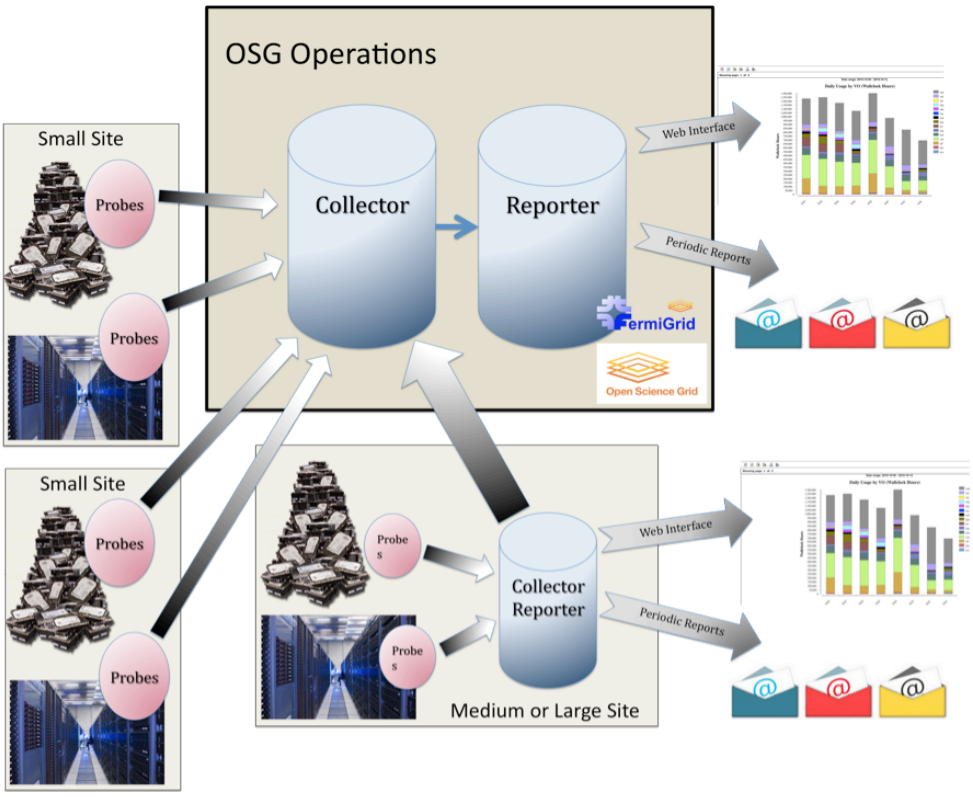
## 2. Gratia System Overview

The Gratia infrastructure is designed to be hieratical. On the lowest level, sensors called 'Probes' look at a service or resource; for example a computing batch system. They extract the information about each usage, each job in the case of a batch system, and push it to an accumulator, called a 'Collector'. The collector, after some data validation stores the information in permanent storage, for example a MySQL database. Often collocated with a Collector is a 'Reporter', which gives access to the data through a web interface, providing textual and graphical representation of the data.

A Collector can in turn push its data or a subset of its data to another Collector. 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. An organization can leverage this hierarchy to create a copy of the usage by its members across the whole Grid by having the central Collector copying the data to the Organization's private Collector.

One of the major focuses is reliability of the system in particular with the use of message caching, when the communication between two elements fails, and a database back-up strategy. Several security mechanisms are used to ensure integrity and non-repudiation of accounting records, as well as to establish secure communication channels between the system's elements.

The information is passed from one component to the other using the XML Usage Record standard[3] published by the Open Grid Forum, a record exchange format defined to facilitate information sharing among grid sites, especially about job accounting.




Example of a Set of probes and Collectors

## 2.1. Probes

A Gratia Probe extracts information about a service or resource, formats it to respect the Open Grid Forum Usage Record XML format and uploads it to a Gratia Collector. The Probe makes all possible attempts to preserve the record. The probe saves a backup of each record under a configurable 'WorkingFolder' directory before any upload is attempted. In the event of a successful upload the backup file is removed. In the event of a failed backup save (for example the amount of disk space used is too large), upload is still attempted. In case of prolonged outage of the Collector, the impact of the storing of the records in minimized by compressing the outstanding records when their numbers get large (this number is configurable) and insuring that the Probes stop storing incoming records before the disk partition is really full. During the next probe run, backups retained due to failed upload attempts will again be uploaded, with the file being removed on successful upload and retained on failure. The probes also regularly upload information on the current size of their backlog.

A probe is configurable via a simple text file (Probe) that set: the destination of the record, the frequency and size of the uploads, the retention policies and location of log files, the maximum amount of files and disk space allowed to be used before the disk partition is considered full. A probe can also be configured to upload or to not upload local records; for example in the case of a batch

system where jobs can be submitted both from a local machine or from a Grid, you can set the Probe to only report about the Grid jobs.

The Connection to the collector is done via either an http or an https post. The Collector can reject connection based on both the hostname where the Probe is run and its certificate (in the https case). The size of each message uploaded to the Collector is configurable and defaults to 100 records per messages.

Currently python libraries exists for the following type of accounting

Storage Elements:

- Static description
- State in time (TB used by VO)
- Accounting of transfers

Compute Elements:

- Static description
- State in time (jobs running)
- Accounting of completed jobs

Metrics:

- Keep history of the monitoring information

Currently a probe exists for each of the following resources and services.

Job Schedulers:

- Wisconsin's Condor workload management system
- Works' Portable Batch System (PBS)
- Platform Computing's Load Sharing Facility (LSF)
- Torque (open source)
- Sun/Oracle Grid Engine job scheduler
- Berkeley Open Infrastructure for Network Computing (BOINC)
- GLEXEC

Data Storage:

- Hadoop
- Scalla/Xrootd
- DCache
- GridFTP

Services:

- BDII
- RSV (Resource and Service Validation)

## 2.2. Collectors

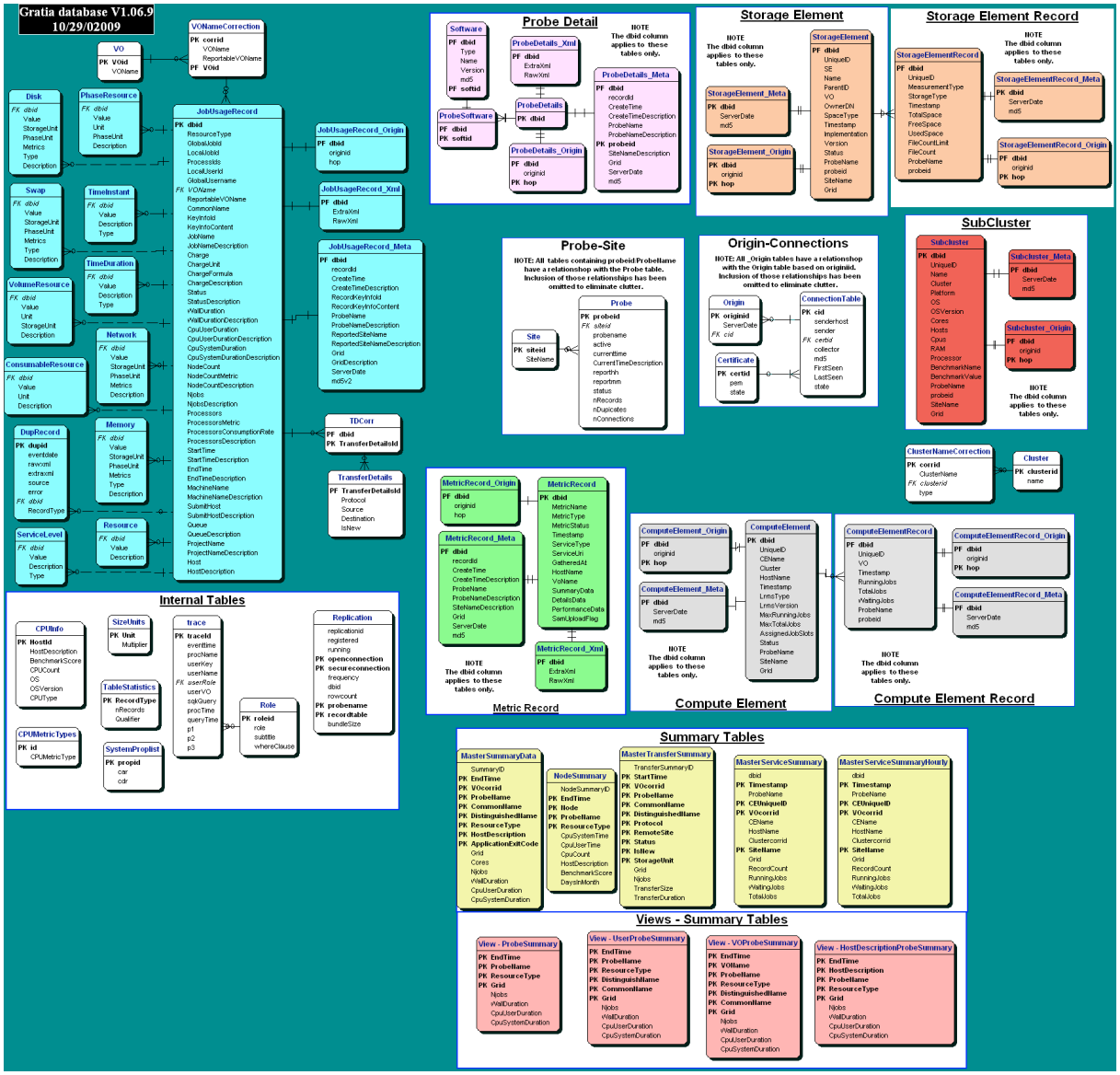
A collector receives usage record information either directly from probes or via another collector. A collector can replicate its information to another collector using the Global Grid Forum XML usage record standard to upload the information. It stores the information in raw form (individual records) for a customizable amount of time (typically 3 months) and creates daily summaries of the information to keep long term and to be used for interactive displays.

### *2.2.1. Functionality: Collectors are a set of Java Servlet, JavaServer pages and Java libraries whose main purpose is to receive, validate, summarize and store usage records.*

When a Collector receives connection from a Probe or from another Collector, it first checks whether the sender has a valid certificate and whether the sender is authorized to upload data to this collector. Both the requirement for a certificate and the check for the authorization can be disabled. The collector has also the optional ability to keep track of the origin of the records. When a Collector who is collecting this origin information pushes its own records to another Collector, this provenance

information is also sent. Therefore the last Collector in the chain has a trace of every single stop in the records' journey.

Once the sender has been authorized, the incoming records are stored immediately on the local file and the Collector reports to the sender that it has taken custody of the records. In a separate thread, the Collector then starts processing each message and analyzes the records. During the analysis, the Collector will fill out any required information that can be derived from other datum, for example the time of the end of the job when the start time and the duration are provided. And it will also fill out any related meta-data table, such as, Virtual Organization names, Site names, and information about the Probes.




Schema Of The Gratia Backend Database.

The Collector also checks whether the record is a duplicate of any already stored record and rejects it unless there is additional information in the new records. For example, the new records might contain the name of the Virtual Organization to which the user belongs while the older records did not.

Finally the record is stored in the backend database (currently only MySQL is supported) and the disk file corresponding to the record is deleted. At the same time, if the type of record supports it, the table summarizing those records is updated.

The Gratia Collector keeps an extended set of status information keeping track of when each probe last reported, which version of the Gratia library it was using and how much backlog, if any, the probe still needs to report. The Collector also keeps historical information about its own performance including the time left to catching up with any backlog that may have accumulated during a downtime.

### 2.3. Reporters

2.3.1. *Web Interface*[17]. Along side the Collector, a Reporter service can be installed. This Reporter provides both graphical and textual access to the data. The graphical results, built using the BIRT library, provides a view of the daily usage filtered or grouped by either Site, Virtual Organization or individual user (and many combination there of). The date range seen is configurable. Each plot provides click-through to access more details and allows the downloading of the textual version of the data in comma separated files or of the graphical version of the data in Excel, Postscript, PDF, Microsoft Word and Microsoft PowerPoint formats.

2.3.2. *Emails Reports*[18]. A series of textual email reports are also available. These email reports can be used to push aggregated and summary information to interested parties including end users, site administrator, site and virtual organization management and stakeholders. These reports include information on the job success rate or the CPU efficiency at a Site and/or for a specific VO, global overview of the use of the whole Grid or a specific, etc.

2.3.3. *External Reporters*. A few third parties have developed additional display to customize the set of information presented. In particular, Nebraska University at Lincoln has developed a different set of graphical reports[19] that includes summary for only the CMS or the ATLAS tier 2 sites, for the opportunistic usage (computational hours done on hardware not owned by the organization using it) of each Virtual Organization, information about the amount of data transfer done across the grid. They also developed an automatic updated display showing the activity of the Grid for both jobs and data transfer over a given time period (24 hours, 30 days or a year).

## 3. New Challenges

A pilot is a grid job that is submitted through the usual Grid infrastructure. Once the pilot job has been started at a specific resource rather than executing a single job, it requests back from the submitter's infrastructure a series of actual jobs potential from different end users and will execute them in sequence.

Pilot jobs are becoming more and more common and pose interesting challenges for accounting. As the pilot job is submitted through the normal channels, it is already fully accounted for via the usual batch probe. However the batch probe cannot report any details on how the pilot job was shared amongst individuals and virtual organizations. To solve the problem, the ideal solution would for a pilot job probe to reports both the internal information and enough data to correlate to the main 'batch' job. However, the information is not always available to the pilot job probe. In addition the collector would usually receive the information about the tasks executed by the pilot job before the information about the pilot job as a whole making the correlation more challenging. In addition, the question on how to properly display this information is unresolved. Backfill jobs, where a special module/helper on the batch system is used to request new workload when some of the batch nodes are idle, present similar type of issues.

Job Forwarding causes yet another set of issues. Both the forwarding and the forwarded-to sites will report some of forwarded job's resource usage. The forwarding site will record it as a job using no processing time (or almost no processing time) and thus would look like a very inefficient job. The forwarded-to site will properly record the job processing information but is currently unable to record that the job itself will also be partially included in the usage report of the forwarding job. Properly recording those types of jobs will require both help from the batch system (to mark those jobs specifically in the logs) and help from the presenter layers to show data with or without those jobs. Recording the forwarding job is important as it does consume some of the resources of the forwarding site, albeit only on the head node.

## 4. Conclusion

Gratia offers a complete and mature accounting solution for both medium and large size Grids. The Gratia probes covers a very wide range of services and resource types and is easily extendable. Both the Collector and the Probes are very efficient and stable. The Gratia Collector is capable of handling at least 200000 records an hour with a single collector making it ideal for large scale Grids. It has been in operation in the Open Science Grid since late 2005 with only minor issues and has been deployed in more than 90 Grid sites.

## References


[1] Gratia Web home https://twiki.grid.iu.edu/bin/view/Accounting/WebHome

[2] Pordes, R., et al. 'The Open Science Grid–its status and implementation architecture' (2007), presented at International Conference on Computing in High Energy and Nuclear Physics (CHEP 07), September 2-7, Victoria, BC, Canada.

[3] Mach, R., et al. 'Open Grid Forum Usage Record Standard, version 1.0', https://forge.gridforum.org/sf/projects/ur-wg

[4] Condor workload management system, http://www.cs.wisc.edu/condor.

[5] Works' Portable Batch System (PBS), http://www.pbsworks.com.

[6] Platform Computing's Load Sharing Facility (LSF), http://www.platform.com/workload-management/high-performance-computing

[7] Torque Resource Manager (open source), http://www.clusterresources.com/products/torque-resource-manager.php.

[8] Sun/Oracle Grid Engine job scheduler, http://www.oracle.com/us/products/tools/oracle-grid-engine-075549.html.

[9] Berkeley Open Infrastructure for Network Computing (BOINC), http://boinc.berkeley.edu.

[10] Groep D., et al. 'gLExec: gluing grid computing to the Unix world', presented at International Conference on Computing in High Energy and Nuclear Physics (CHEP 07), September 2-7, Victoria, BC, Canada.

[11] Hadoop, http://hadoop.apache.org.

[12] Scalla/Xrootd, http://xrootd.slac.stanford.edu.

[13] DCache, www.dcache.org.

[14] GridFTP, http://www.globus.org/toolkit/data/gridftp.

[15] BDII, http://is.grid.iu.edu/documentation.html.

[16] RSV (Resource and Service Validation), https://twiki.grid.iu.edu/bin/view/MonitoringInformation/RSV.

[17] Gratia Reporting Interfaces, https://twiki.grid.iu.edu/bin/view/Accounting/WebHome#GUI_reporting_interfaces.

[18] Gratia Email Report, http://twiki.grid.iu.edu/bin/view/Accounting/ReportsDescription, https://twiki.grid.iu.edu/bin/view/Accounting/EmailReportsPerlInfrastructure.

[19] OSG Overview page, http://t2.unl.edu/gratia.