



Open Science Grid

The OSG Accounting System: GRATIA

by

*Philippe Canal (FNAL), Chris Green (FNAL),
Jeff Mack (FNAL), Penelope Constanta (FNAL),
John Weigand (FNAL)*

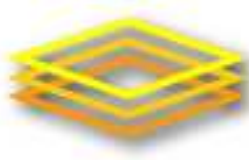
Victoria, British Columbia, Canada

CHEP 2007

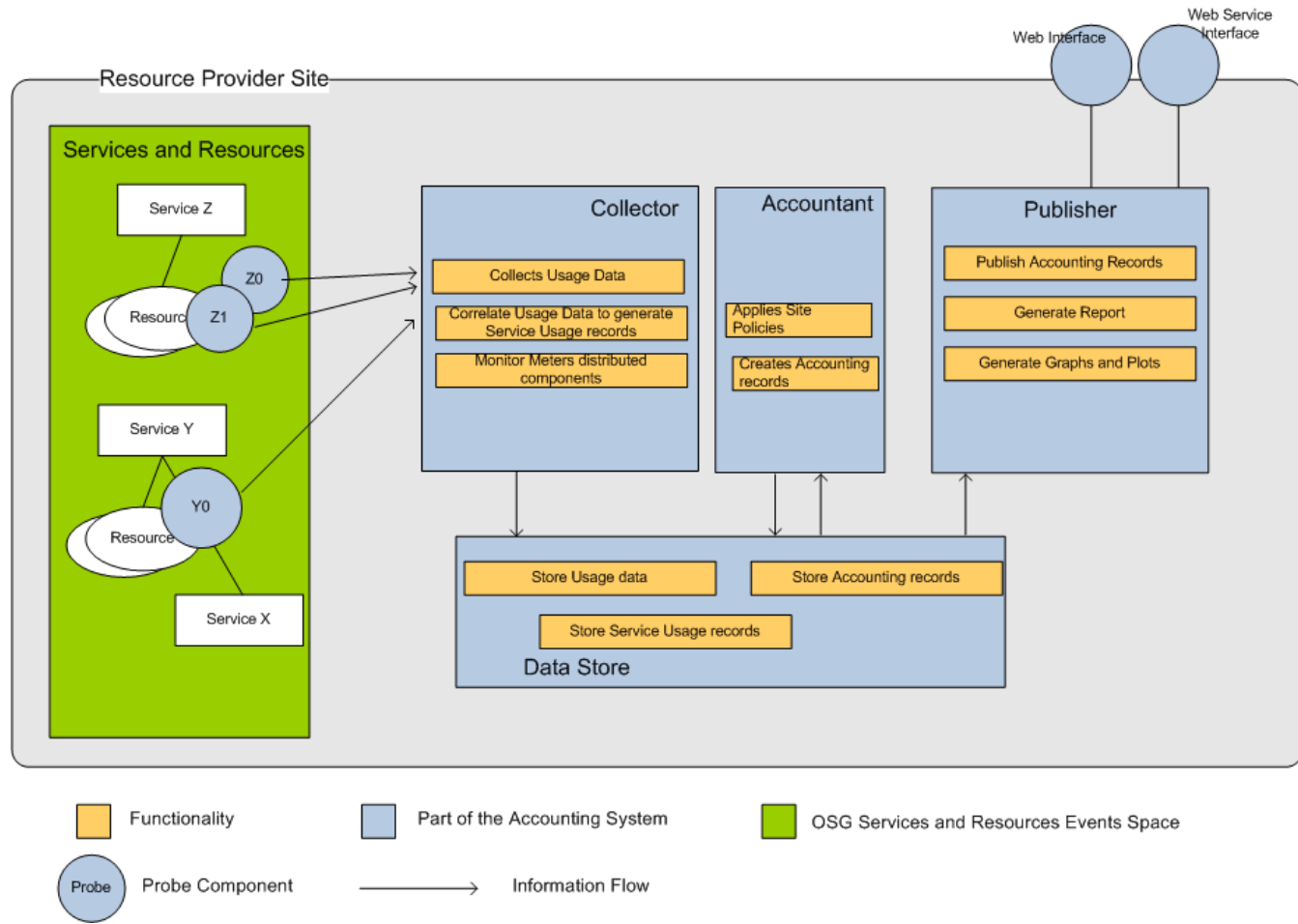


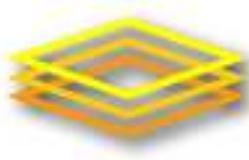
Outline

- Architecture Overview
 - probe->collector->collector
 - Implementation details (python library, JMS web server, java code, hibernate, db, birt)
- Deployment
 - list current and planned probes
 - boast the number of sites Gratia probes are installed on
 - ready for multi-tier infrastructure but single database not yet a performance issue (and won't be a reliability issue due to fail-over mechanism and db backup strategy).
 - display some numbers from OSG
- Collaboration/Interoperability
- Lessons Learnt
 - Easily extended
 - BIRT
 - Reliability
 - Database schema(s)

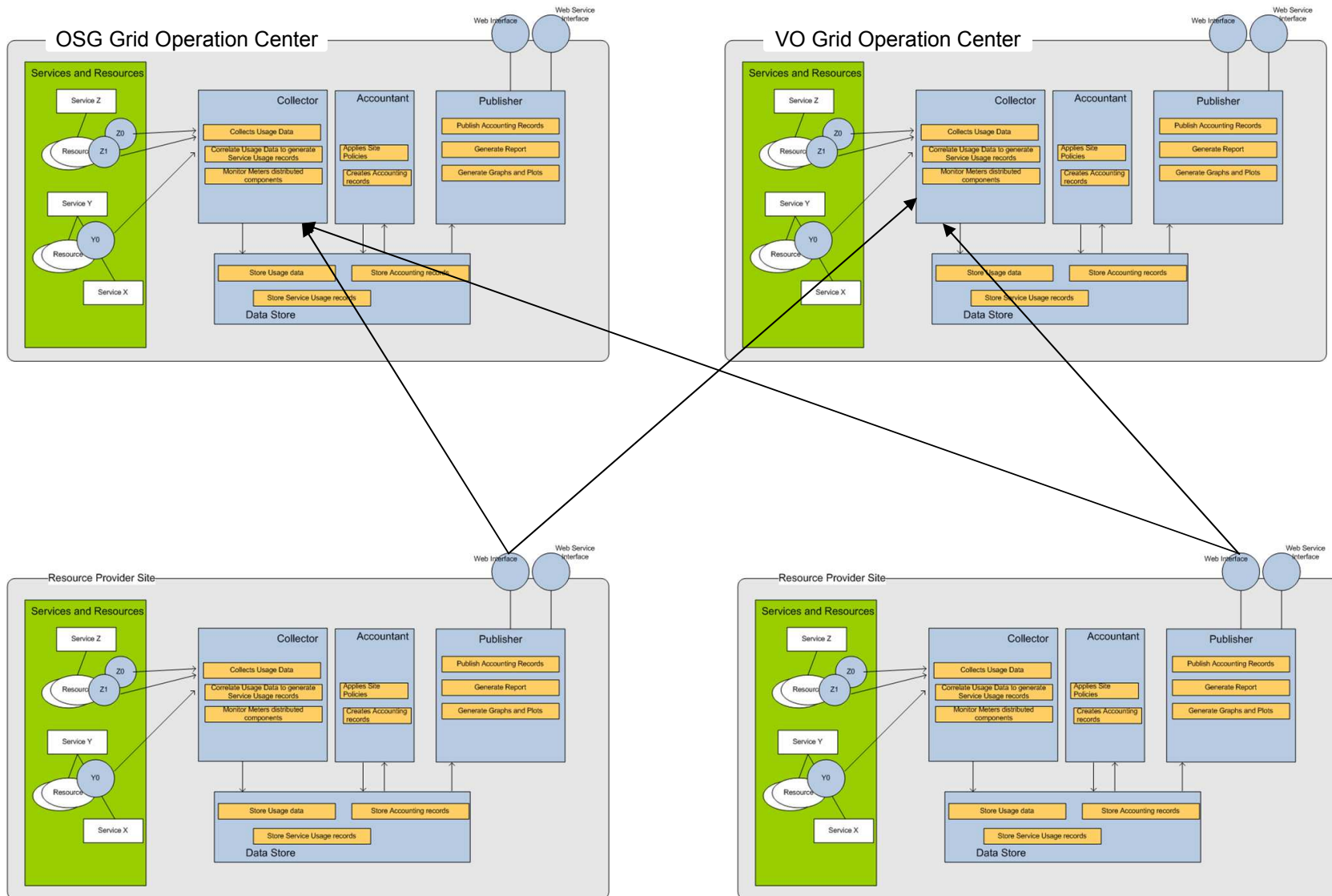


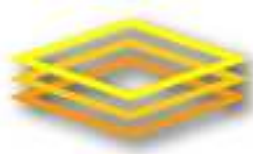
Architecture Overview





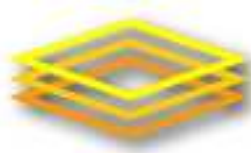
Architecture Overview





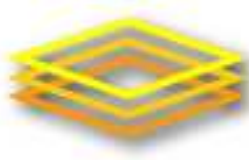
Implementation Summary

- Probe
 - Gratia Probe library implemented in Python
- Collector
 - Tomcat
 - JMS Queue
 - XML -> Java Object
 - Policies and fixes applied to record to standardize it
 - Fill in missing VO, Fill in missing StartTime, etc.
- Database
 - Hibernate used to store java object in database
 - MySQL.

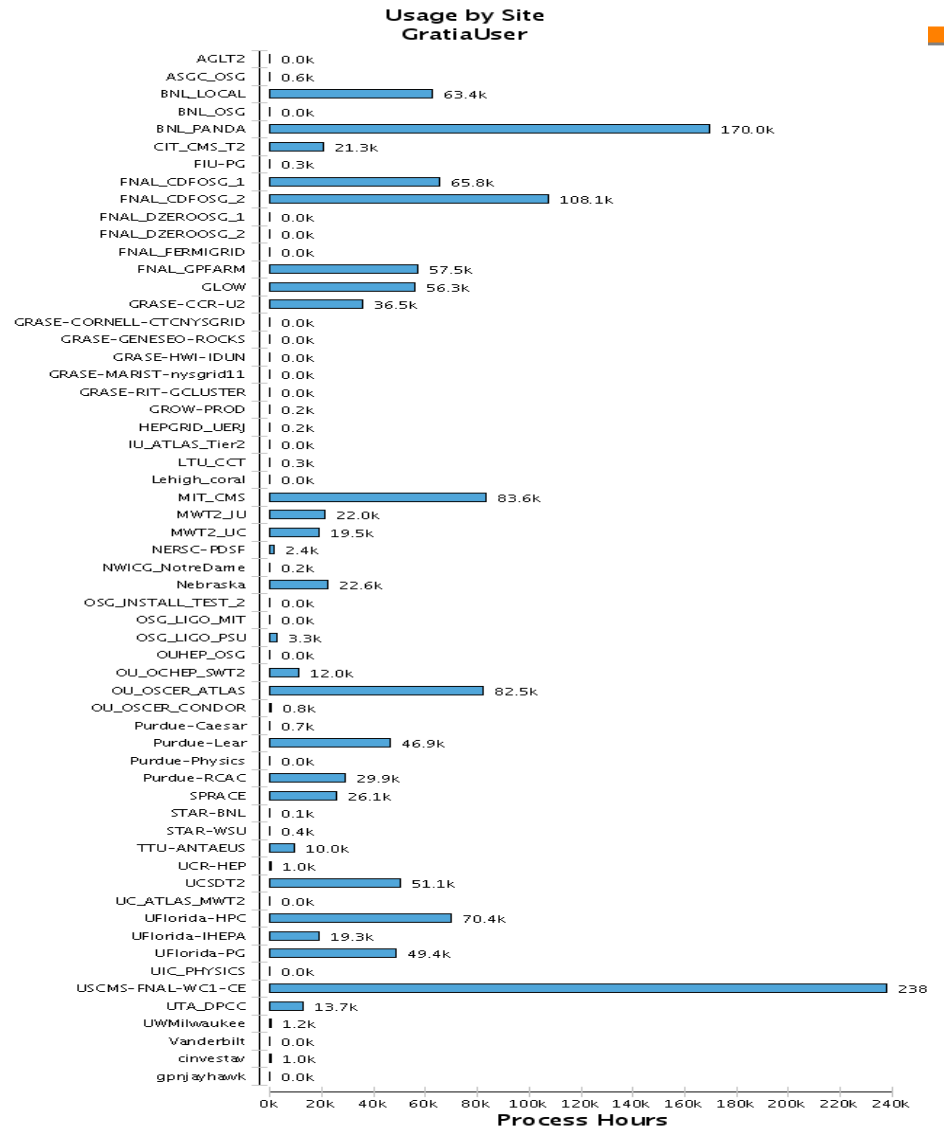
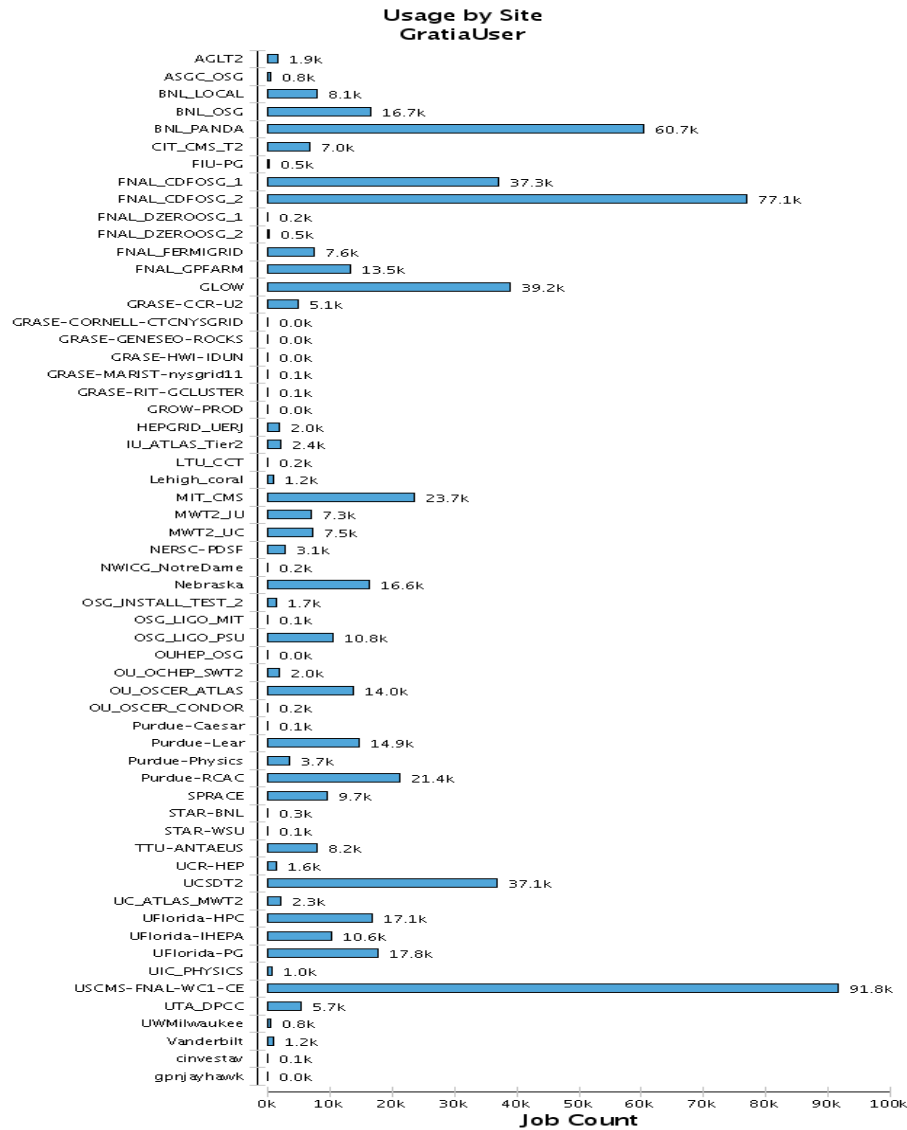


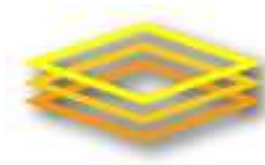
Deployment

- Services currently supported:
 - Condor 6.8 when submitted through OSG Gram/Globus
 - Condor 6.9, PBS, LSF, Sun Grid Engine
 - DCache
- Coming soon:
 - [gridftp](#), [storage](#).
- 2/3 of the OSG Sites are now reporting to Gratia
 - [Until OSG 0.8.0 deployment has been voluntary](#)
- We typically get 5000 jobs an hour.
 - [Capacity with current hardware is at least 300,000 records an hour.](#)
- Code is ready to be deployed using multi-tier infrastructure. So far single database not yet a performance issue
 - [It is not a reliability issue due to fail-over mechanism and db backup strategy](#)



OSG By The Numbers

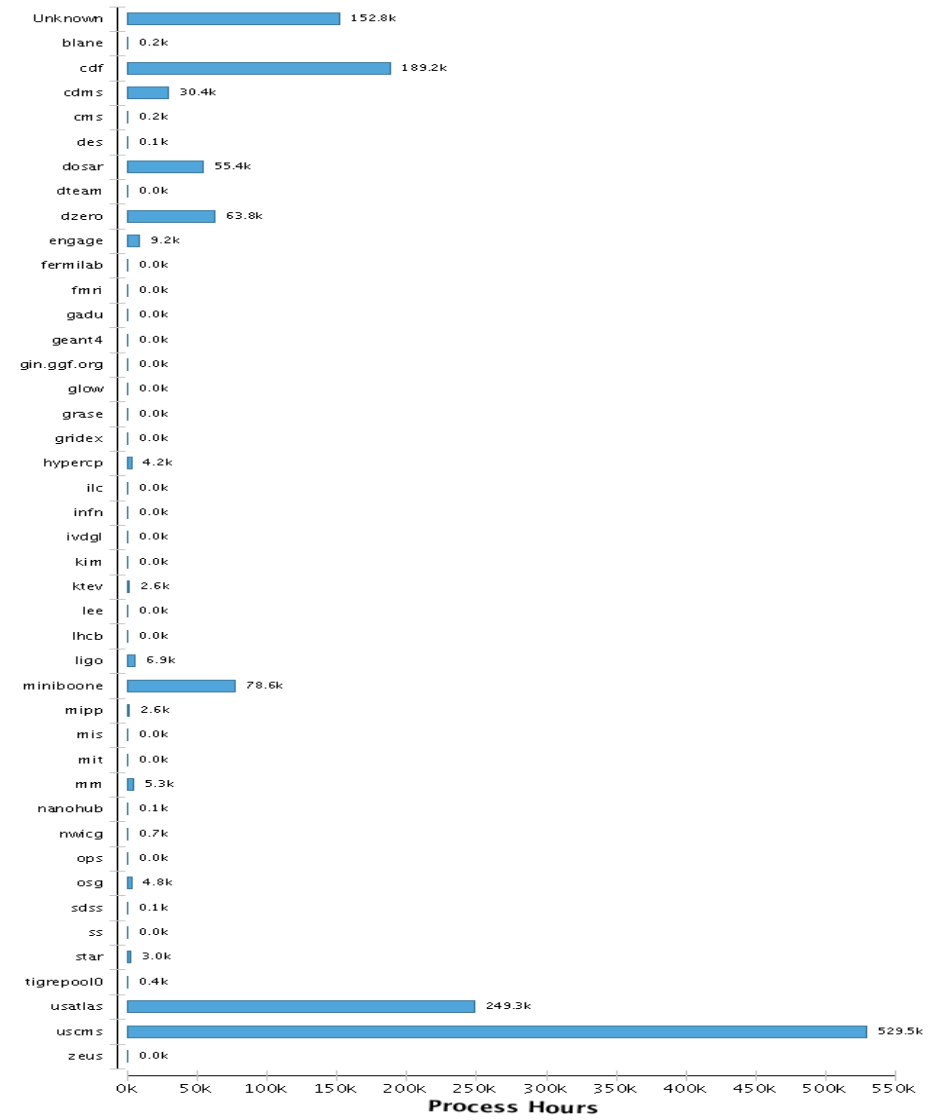
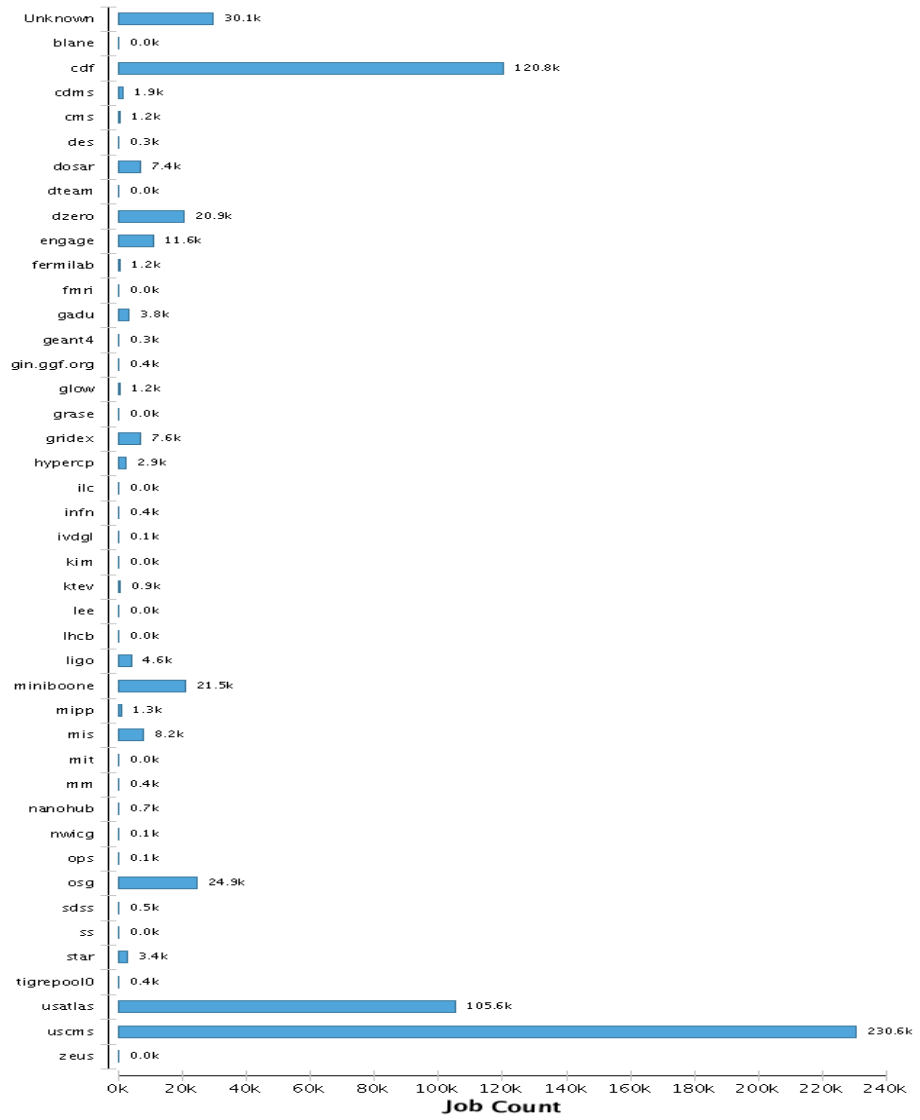


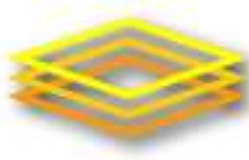


OSG By The Numbers

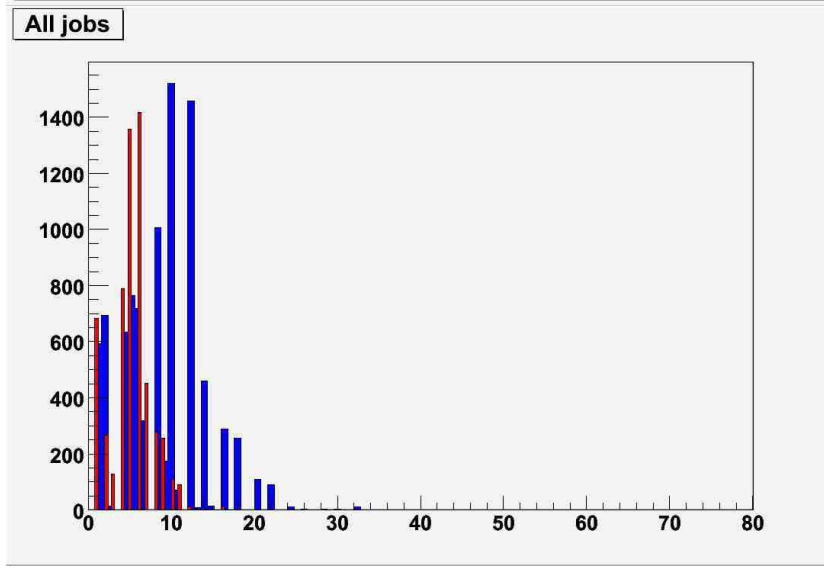
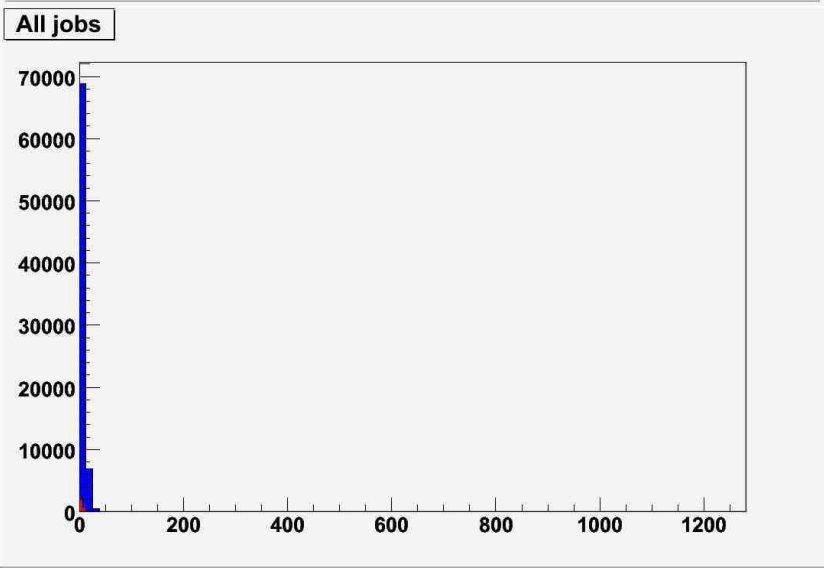
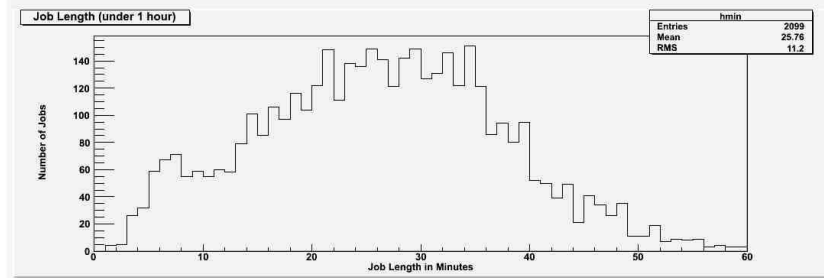
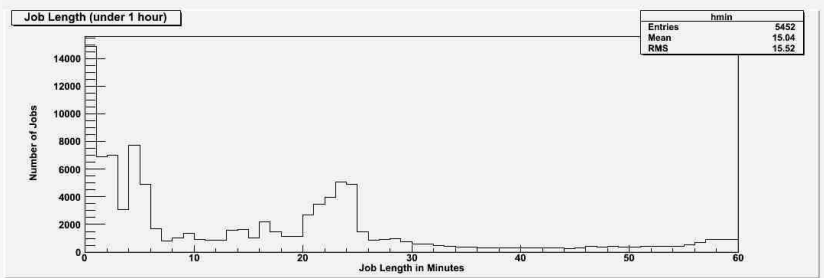
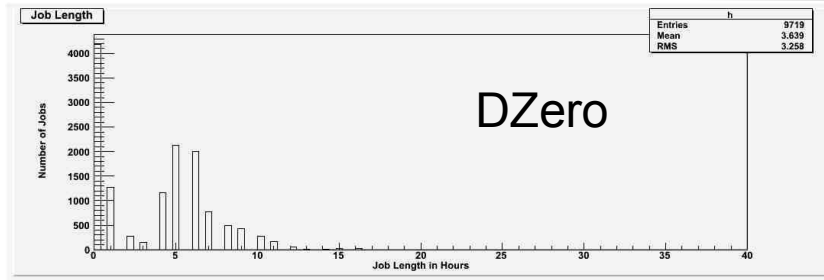
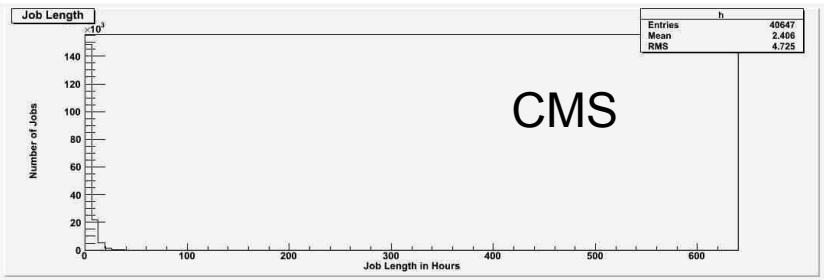
Usage by VO
GratiaUser

Usage by VO
GratiaUser





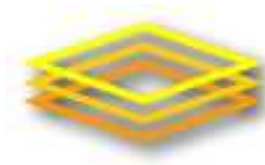
OSG By The Numbers





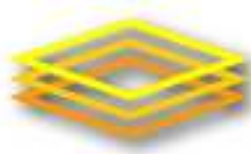
Collaboration/Interoperability

- On behalf on the ATLAS and CMS tier 1 and tier 2 that are part of OSG, OSG uploads usage information from Gratia into the APEL database.
- The update is done daily but refresh the information for the previous month until the 15th (to catch up with record that take more than 24 hours to reach the database).
- The usage information is aggregated per day, per VO, per Site.
- The data is uploaded using 'direct' sql insert into an OSG table which is then copy into the main APEL schema.
- This has allowed for 'restating' of information.
- Currently the normalization factor is 'estimated' from the composition of the farm (when known).




Lessons Learnt

- Easily extended
- BIRT
- Reliability
- Database Schema(s)




Easily Extended

- Whole <input> infrastructure was extended to a new type of record in a few days.
- Re-used Gratia infrastructure code to implement the OSG Resource and Service Validation
- Implemented:
 - New type of xml message
 - Update Probe library to add one more record class
 - Corresponding Java classes
 - Corresponding Hibernate schema file



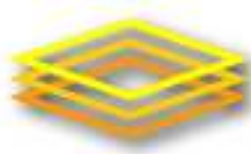
BIRT

- BIRT was selected because it is or has:
 - Open Source
 - Graphical Designer integrated in Eclipse
 - BIRT is a top level Eclipse project since September 2004.
 - Good integration with Tomcat
 - Object-oriented Reporting – Libraries and templates
 - Reusable Styles – Commonly used layout and formatting choices can be defined in reusable styles that can be applied to multiple reports
 - Extensible
 - Scripting support - Custom scripting to incorporate complex business logic and data access into report designs
 - Complex report layout - APIs permit the introduction of new visual components into reports and new attributes for existing components
 - Flexible charting - Pluggable architecture for incorporating custom charts and new graphic formats into the charting engine



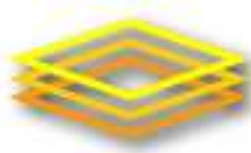
BIRT

- Printed and online documentation does not provide adequate information to enable more complex applications to be developed without a lot of time wasted using the “trial and error” approach.
- Integrating the BIRT runtime engine into a custom web application is difficult because of poor documentation and packaging.
- Upgrading reports to new versions are not backward compatible and conversion method frequently breaks the reports.
- Eclipse/BIRT development environment does not always reflect report design changes when previewing reports.
- Bug reporting process is inefficient and too slow.



Reliability

- One of the primary goals in the implementation of Gratia.
- Implementation:
 1. The Probe library caches the XML messages locally when the communication with the Collector fails
 - Allowed seamless server upgrades
 - Recovery from probe misconfiguration.
 2. The Collector caches the XML messages locally when the communication with the back-end database fails.
 3. The Collector keep a local copy of the processed messages for a specified (configurable) time period.
 4. The Back-end database is regularly backed-up.



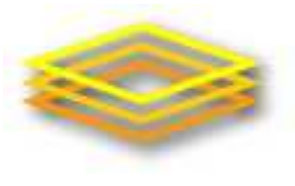
Database Schema(s)

- Representing OGF XML Usage Record
- Goals: Simple, Flexible, Able to cope with (unexpected) changes.
- First Implementation:
 - Single Table for the one-of elements
 - Auxiliary Tables for the many-of elements
 - Extra columns to store original XML record and 'unused' part of XML record.
 - Advantages: Queries were simple and straightforward
 - Disadvantages: quite slow
- First Improvement: (1000x)
 - Use summary table (per day aggregation)
 - Fast enough for usual queries.
 - Not 'rich' enough (many query still requires access to the per job information)
- Second Improvement: (50x)
 - Split single (very large) table into 3 tables
 - In particular separate the XML copies into their own tables
- Future Improvements:
 - Use MySQL horizontal partitioning (MySQL 5.1)
 - Use Foreign keys for 'repeated' strings (the OGF XML UR contains a lots of descriptions fields).

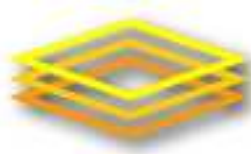


Conclusion

- Future Plans
 - Storing the Usage as measured by the VO software (Panda, etc.) to allow for easy comparison.
 - New services probes, improve normalization factors, verify data quality.
 - Better reporting.
- More Information
 - [Project Charter](#), Requirements and Design Documents
 - [OSG Accounting Twiki](#) page and
 - [Mailing list](#): osg-accounting@openscience.org
- Any Questions, Comments, etc?

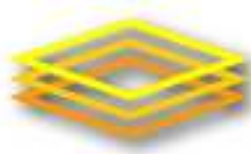


SPARE SLIDES



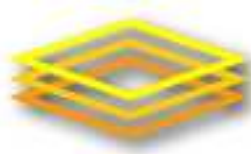
Goals

- Track services and resources usage *per grid user* after the fact
 - Focus on *quality, integrity and security* of the information
- Accounting Information *easily* available to people (web interface) and to applications (Web Services)
- Build a system that is simple to manage (install, configure and upgrade) and to extends (well defined APIs)
- Based on well proven and standard (industrial strength) technologies
- However we do *not* cover (but keep in mind)
 - User charging system,
 - Resources or services pricing
 - Support for an economic model for resource allocation



System Properties

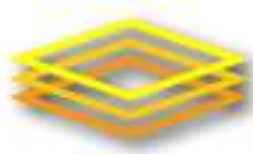
- Interoperability
 - The Accounting System should leverage existing standards to maximize interoperability with other Grids and Accounting Services.
- Fault Tolerance
 - Reduce and flag data loss.
 - Resilient to communication failures over LAN and WAN.
 - resilient to the failure of one of its component.
- Security
 - Guarantees integrity and non-repudiation of the accounting records at the site level.
 - Uses secure communication channels (mutual authentication, message integrity, confidentiality) and access control lists.
- Scalability and Performance
 - Not really an issue
- Other
 - leverage existing tools and infrastructures to solve related problems.



For Example: Monitoring at SLAC

What do we monitor:

- Network
 - Switches, routers status Internet Mbytes/sec in/out
- Computer Clusters
 - Batch systems, NFS and AFS servers, databases servers
- Storage Space
 - Disks usage, HPPS
- Some metrics we use:
 - CPU utilization, Memory
 - Disk usage, Disk I/O
 - Various Networking metrics (Mbytes in/out of switches, routers, servers...)
 - Some primitive job submission results (LSF)
- We use a lot of monitoring tools and infrastructure: Ganglia, Nagios, OpenView, SNTP tools, Monalisa...



Design Direction

- Open: we give APIs
- Distributed: Meters are distributed objects
- Based on open source standard technologies: Web Services, Java Platform, Tomcat, Axis, Hibernate
- Same idea as GUMS and JClarens: the service is an independent Tomcat Application
- Insure interoperability with OSG partners (LCG, TeraGrid...)