

# Next Generation Accounting

## Overview

Goals in 2007:

- 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 extend (well defined APIs)
- Based on well proven and standard (industrial strength) technologies - not really the case anymore. Tomcat/Java seems to be overkill for this, BIRT was a disaster, Hibernate seems to be an industry standard but bugs (e.g bulk deletion) has never been fixed.

Since 2007:

- from MySQL 5.1 to MySQL 5.6
- from Java 4 to Java 7 with very little code modification
- from hibernate 3 to hibernate 4
- get rid of BIRT
- migrating to google chart for gratiweb
- New Probes:
  - xrootd, hadoop, dcache, enstore storage
  - xrootd, hadoop, dcache, enstore transfer
  - enstore tape usage
  - ONE VM probe
  - AWS probe
  - SLURM probe
  - glideinsWMS probe
- BatchPilot, Storage, AWS, ONEVM resource type added
- ProjectName for campus was implemented
- New Reports:
  - Campus grid reports
  - Flocking reports
  - Best in class
  - and many others
- WLCG and XSEDE integration

## Current Stats

- 5 collectors (osg, osg-transfer, osg-itb, fermilab (cloud), aaf)
- 2 database servers (one is replicate)
- OSG collector:
  - 7.4 M records in SummaryData (jobs and pilot jobs)
  - Started at 2004/09
  - Stopped removing JobUsageRecord since 2015/01 - after hibernate upgrade  
Number of records now : ~380M (rate ~1M/day)
- OSG Transfer collector:
  - 41.9M records in TransferSummary
  - Started at 2008/04
  - JobUsageRecord (transfer records) -
- Size of databases:
  - gratia 962.6GB
  - gratia\_osg\_transfer 1005.2 GB
  - gratia\_itb 42.5GB
  - gratia\_fermi 11.9GB
  - gratia\_aaf 994.2GB

## Requirements

There are several requirements that are mandatory for any new proposed replacement for Gratia:

1. All historical summary data (job records and transfer) needs to be preserved
2. Data should be archived and It should be possible to extract the historical data in suitable format in order to upload to the future accounting service
3. Gratia probes should not be changed drastically and new service should be able to deal with older version of probes.

## OSG:

OSG is using gratia accounting to provide reports to:

- DOE and NSF
- WLCG and XSEDE
- Institutional PIs and resource admins
- VO managers
- ET and OSG staff (operations, user support, etc)

In order to provide these reports OSG needs the following information:

- Daily summary of job wall duration, cpu usage? per site, VO, Project, DN (with role), user, exit code for Batch and BatchPilot resources
- Daily transfer summary (size and wall duration) per storage site, vo, user, direction, exit code.
- Ability to aggregate at different levels (site versus cluster versus CE) and rename site aggregation.

## Fermilab:

Fermilab is currently relied on Gratia accounting for several projects.

- AAF Project

AAF is using Gratia to report charges to customers that based on several quantities:

- The total accumulated amount of tape storage used at the end of the year
- The amount of tape media that needs to be acquired for the year
- The amount of tape-drive hours used

The portion of the yearly cost for media is spent at the beginning of the year, and the costs associated with the total amount of data stored on tape and the drive-hours are drawn on a monthly basis. The metrics that these charges are based on are available on the Customer's Active Archive web pages. These are updated daily, and history of monthly summaries are available. In order to provide these reports AAF Project needs the following information:

- daily transfer summary (size and wall duration) per storage type (enstore and dcache), vo, user, direction, exit code.
- daily storage on tape per vo
- daily dcache pools utilization per vo
- daily duration and count of tape mounts

- HEPCloud Project

HEPCloud is extending the current Fermilab Computing Facility to transparently run on a variety of resources including commercial clouds. It is using Gratia to get historical information about instantiated VMs (wall duration, cpu usage, charges) per vo, instance type, availability zone.

- FIFE Project

FIFE is using gratia accounting to facilitate experiments preparation for SPPM meetings, get historical information about users efficiency, success rate, data transfer and dCache pool usage. In order to provide this information

- Daily summary of job wall duration, cpu usage? per site, VO, user, exit code for Batch and BatchPilot resources
- Daily transfer summary (size and wall duration) per storage site, vo, user, direction, exit code.
- Collect daily dcache pools utilization per vo

## Nebraska:

All internal HCC usage accounting is done using gratia. Reports to stakeholders and users often include gratia produced graphs. In order to do so the following information is needed:

- daily summary of job wall duration per site, VO
- use the \*-running probes to [show utilization graphs](#)

## CMS Tier-1

According to Dave Mason CMS Tier-1 doesn't need accounting information but we need to report to WLCG and the only source is Gratia.

## Evaluation of Open Source Solutions

There are several accounting services that are used by other communities.

1. XSEDE (XDMoD)
  - [evaluation](#) (Marco Mambelli)
  - Main findings:
    - The open source version of the project Open XDMoD can acquire information only about jobs running on PBS (OpenPBS, Torque/Maui)! or SLURM clusters.
    - Uses mysql
    - The role-based customizable views.
    - The display uses standard technologies that reduce the load on the server and move it to the client (HTML5, JavaScript).
    - The system monitors only job metrics (when, where, how and how many jobs ran and which projects they relate to) and is not really flexible. The addition of new metrics and the addition of new queries would both require coding.

2. EGI Accounting (APEL)

APEL is an accounting tool that collects accounting data from sites participating in the EGI and WLCG infrastructures as well as from sites belonging to other Grid organisations that are collaborating with EGI, including OSG, NorduGrid and INFN. The accounting information is gathered from different sensors into a central accounting database where it is processed to generate statistical summaries that are available through the EGI/WLCG Accounting Portal. Statistics are available for view in different detail by Users, VO Managers, Site Administrators and anonymous users according to

well defined access rights. It is written in Python and uses MySQL. It has the following components:

- **apel-parsers:** extracts data from the following batch systems: LSF, PBS SGE/OGE, SLURM, HTCondor and place it in the client database.
- **apel-client:** processes the data and sends it to the APEL server using SSM.
- **apel-server:** processes data from all sites and sends it on to the accounting portal using SSM.
- **apel-lib:** contains all the library python code for apel-parsers, apel-client and apel-server.
- **apel-ssm:** is a messaging system

Stores only JobUsageRecords (only jobs that are started via batch manager - could be pilots or actual jobs). Probably, started to work on Transfer data (not enough manpower). Accounting Portal is graphical front-end that allows to see usage per VO/site/month.

## Proof of Concept

A minimal collector program was written to accept forwarded bundles of records from an existing collector and index them into Elasticsearch. The collector splits the bundle into individual XML records, and converts the XML into a “flattened” JSON structure. Several instances of this collector were run, with JobUsageRecord forwarding enabled from the following collectors:

1. OSG Main collector (gratia-main-osg.fnal.gov)
2. OSG ITB (gratia-itb-osg.fnal.gov)
3. HEP Cloud/AWS (fermicloud054.fnal.gov)

No changes were made to any probes or existing collectors.

In addition, logstash was used to migrate the complete MasterSummaryData table (7.4 million records) from the main OSG database to Elasticsearch, with foreign key fields denormalized (e.g. VO, Site).

## Elasticsearch Development Cluster

- Three-node cluster of VMs running on Fermicloud, each with 4 cores, 8 GB RAM, and 250 GB disk.
- All access through proxy server running on fifemondata.fnal.gov, requires authorized certificate to perform write or admin operations.
- Serving 15K requests/hr

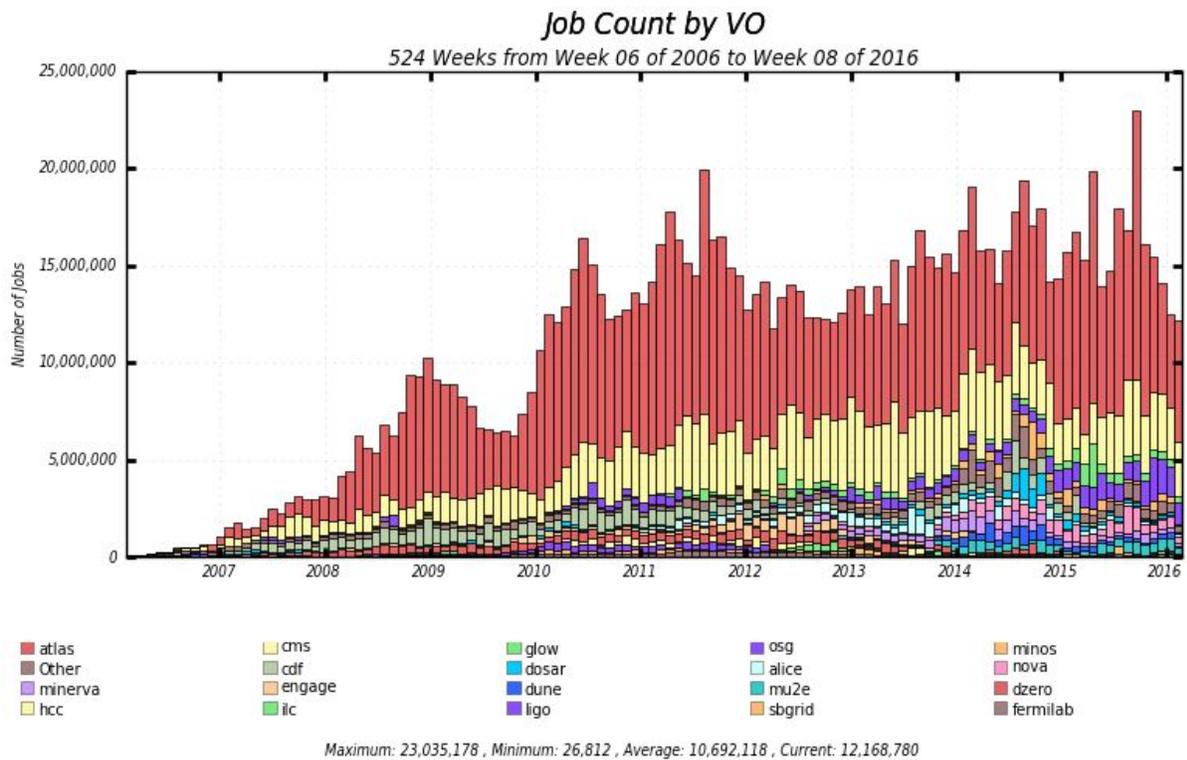
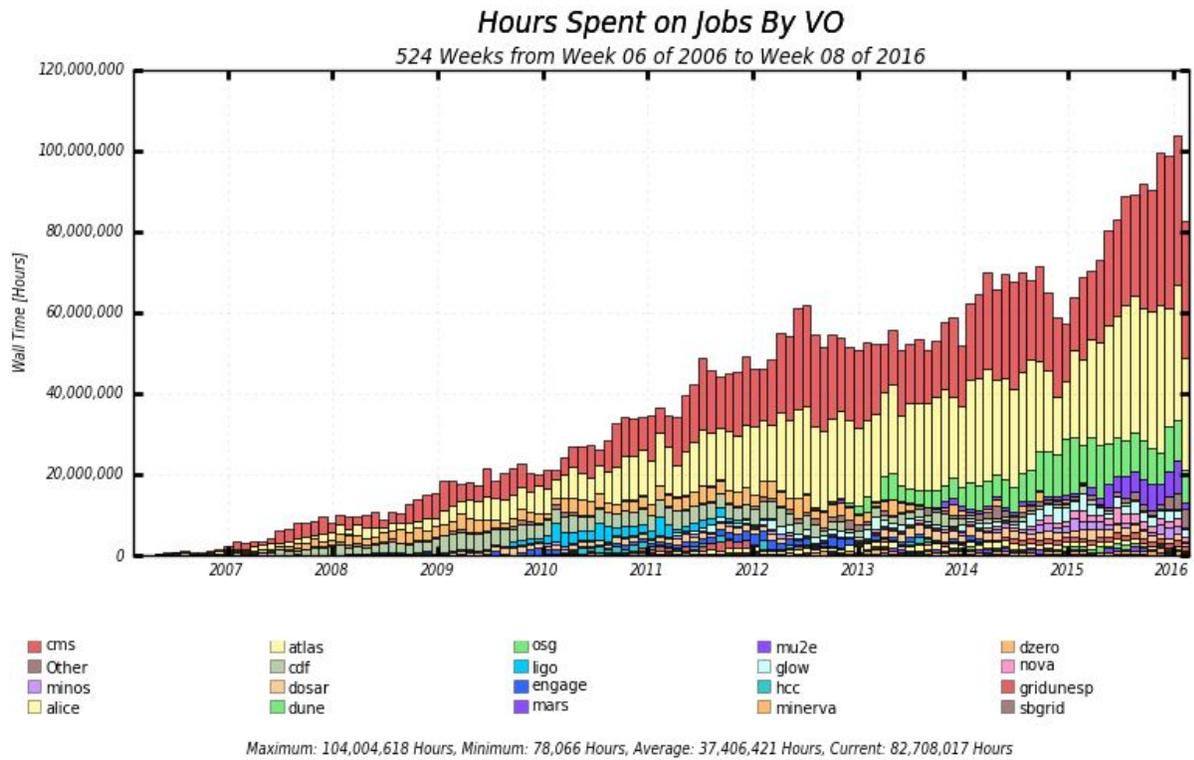
- Usage stats as of 2/22:
  - 168 indices, 83M documents, 38 GB
    - Gratia OSG MasterSummaryData (7.4M records & 3 GB)
    - Gratia OSG JobUsageRecord (1.5M records & 1 GB per day)
    - Gratia ITB JobUsageRecord (200K records & 117 MB)
    - Gratia AWS JobUsageRecord (2.5M records & 1.2 GB)
    - Fifebatch monitoring:
      - Job attributes (2.5M records & 1.5 GB per month)
      - Slot attributes (5M records & 2 GB per day)
    - Misc logs and test data

## MySQL vs Elasticsearch Storage

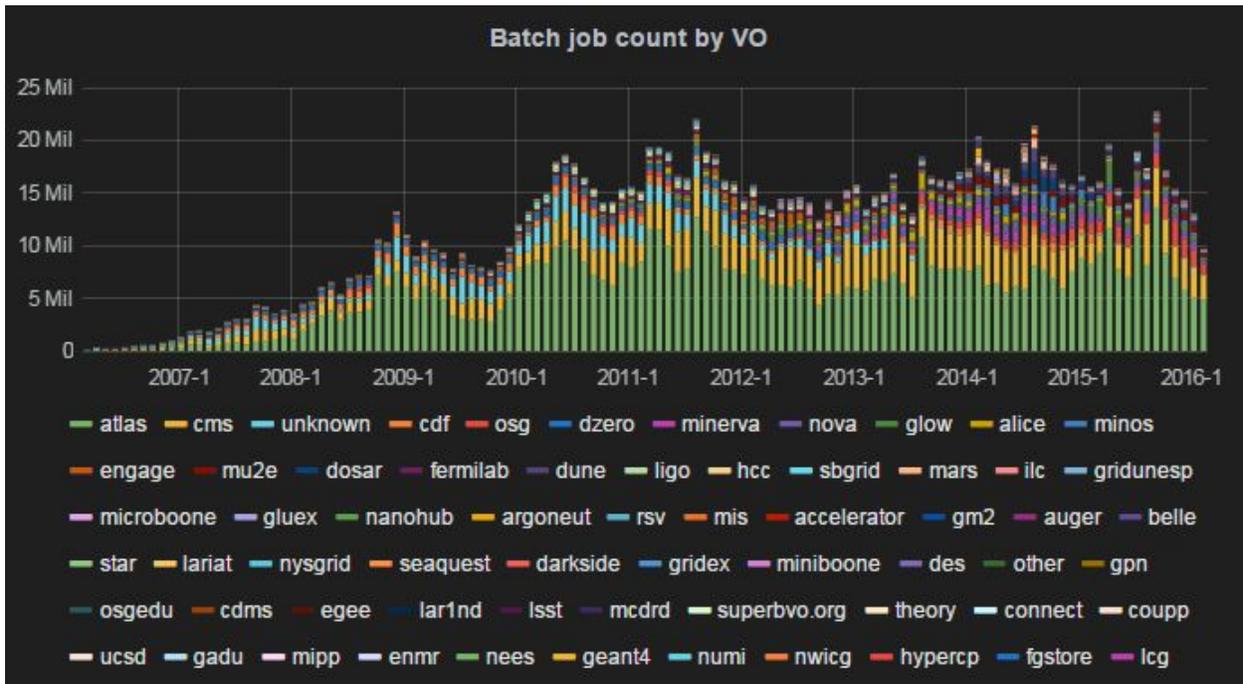
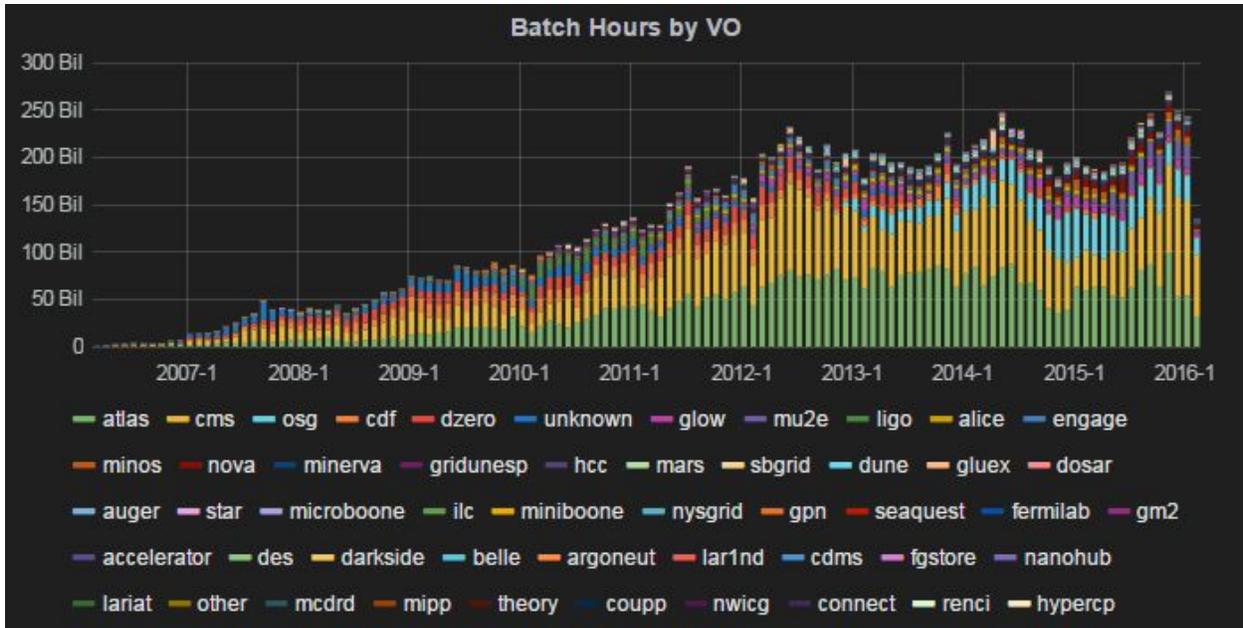
	MySQL Record Size	Elasticsearch Record Size
MasterSummaryData*	0.2 KB (1.2 GB / 6e6)	0.4 KB (3.0 GB / 7e6)
JobUsageRecord	0.6 KB (221.6 GB / 381e6)	0.7 KB (1.1 GB / 1.7e6)

\* MasterSummaryData is denormalized in Elasticsearch.

# GratiaWeb vs Grafana + ElasticSearch



When using GratiaWeb and the Gratia MySQL database for queries of multiple years, the plots can take a long time to be generated, and sometimes even timeout. In order to compare the performance. The data from MasterSummaryData (about 7 million rows) was migrated to elasticsearch using logstash. Similar plots were created using Grafana.



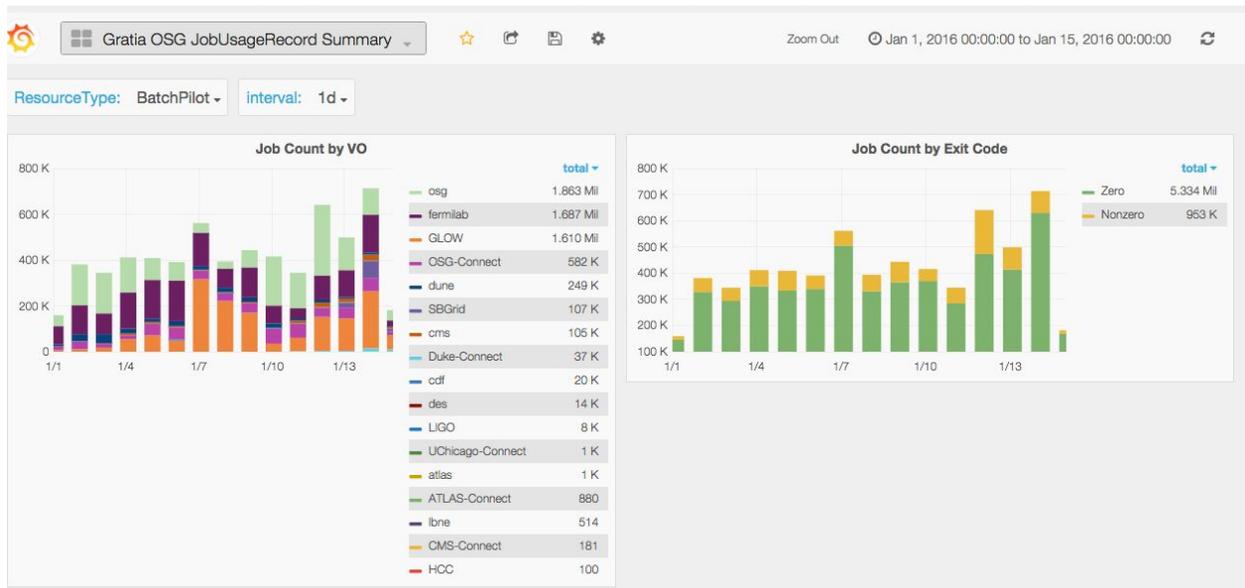
The response time using elasticsearch in comparison with MySQL decreased from minutes to seconds for large amounts of data (10 years).

# Grafana Dashboards from Elasticsearch

The Grafana dashboard framework (<http://grafana.org>) is currently used to power Fifemon (<https://fifemon.fnal.gov>). It can graph data and generate tables from several sources: Graphite, InfluxDB, Elasticsearch, KairosDB, OpenTSDB, Prometheus, and Amazon CloudWatch. Example dashboards were created in Grafana displaying the Gratia data from Elasticsearch:

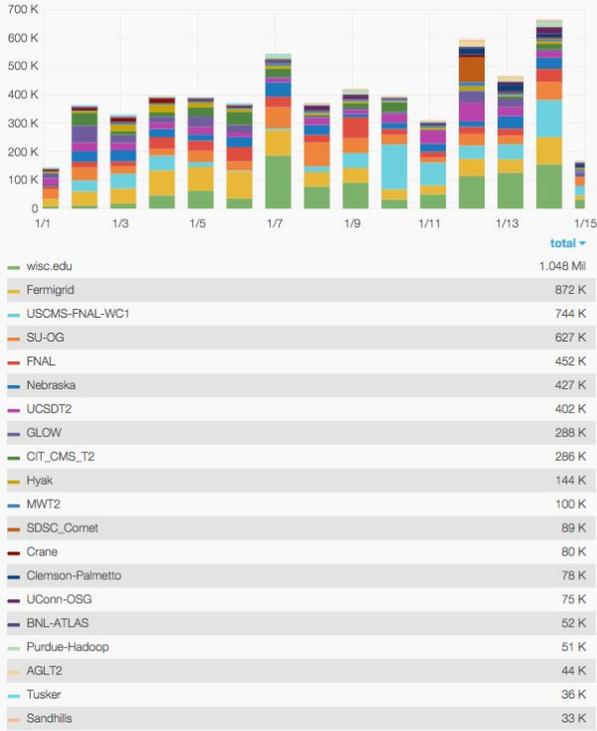
## OSG JobUsageRecord

<https://fifemon.fnal.gov/gratia/dashboard/db/gratia-osg-jobusagerecord-summary>

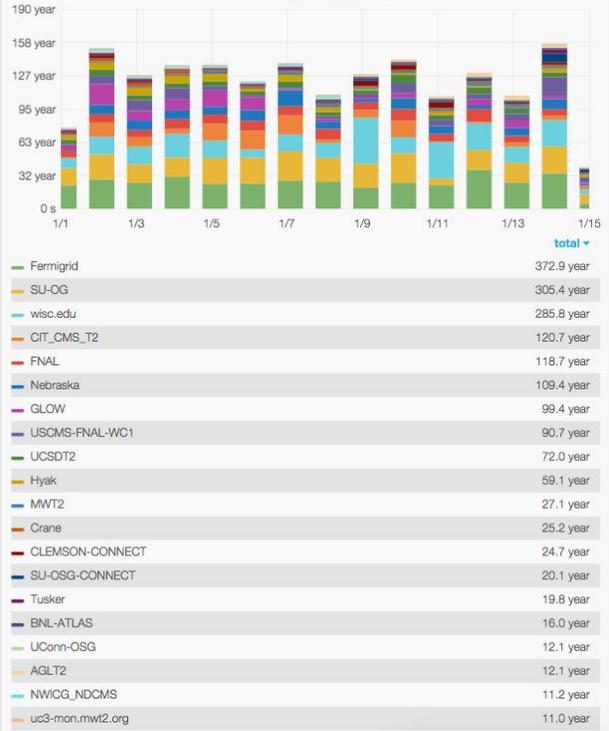


GLIDEIN SITES (HOST DESCRIPTION)

Job Count by Site (Top 20)

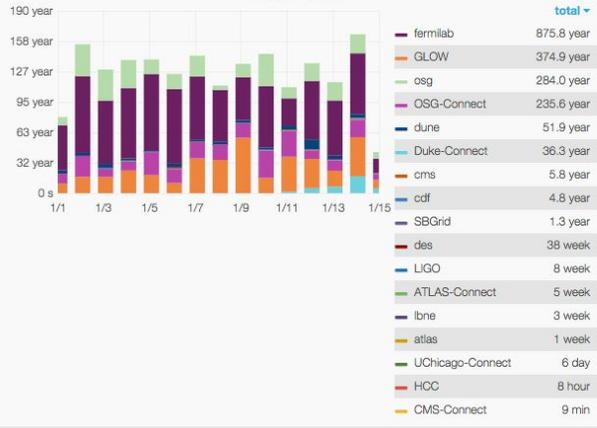


Walltime by Site (Top 20)

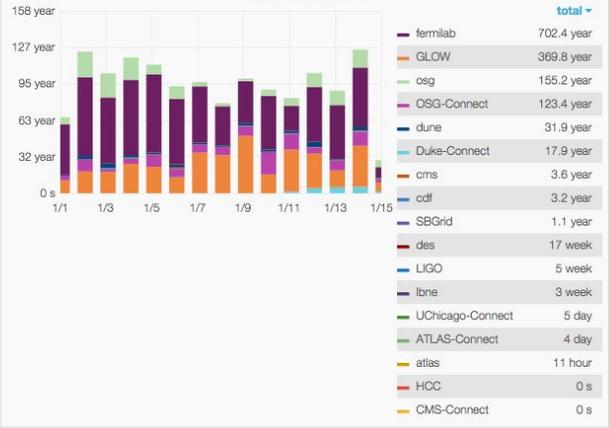


WALL AND CPU TIME

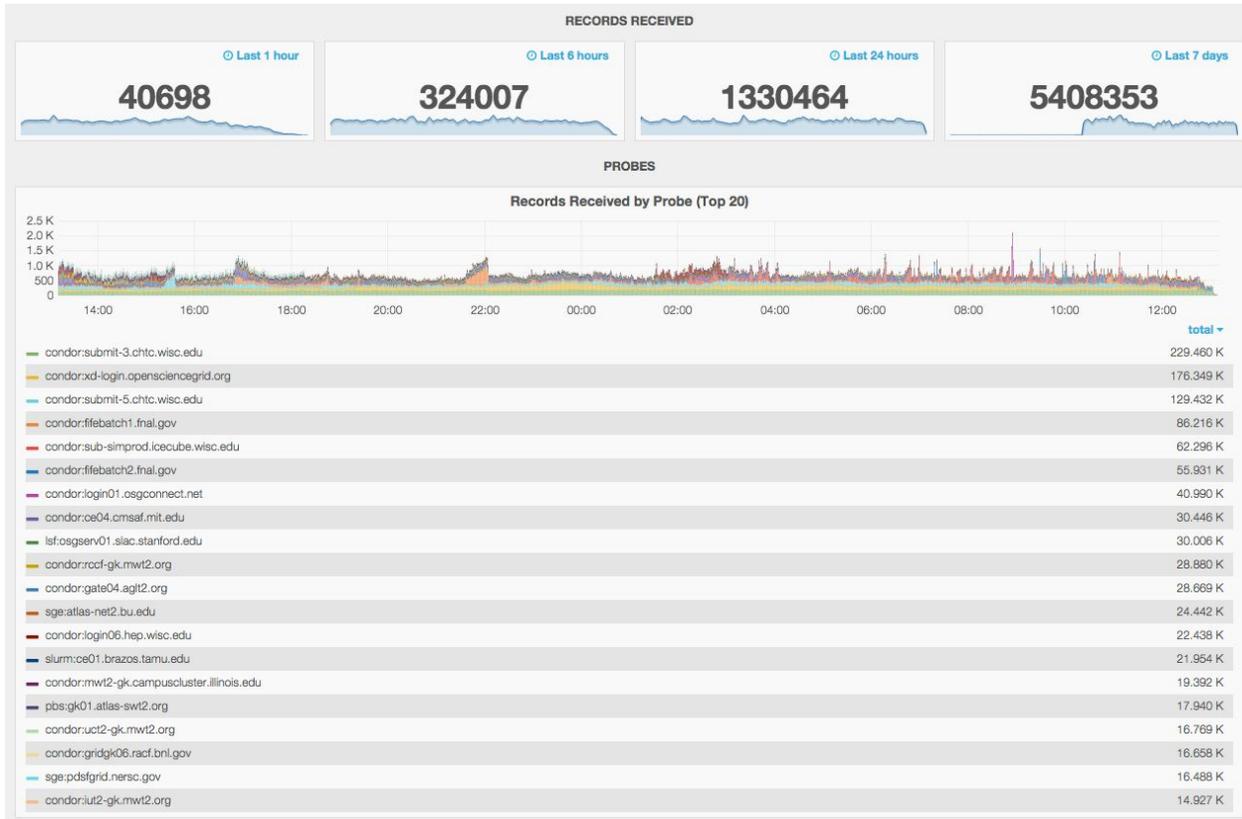
Walltime by VO



CPU Time by VO



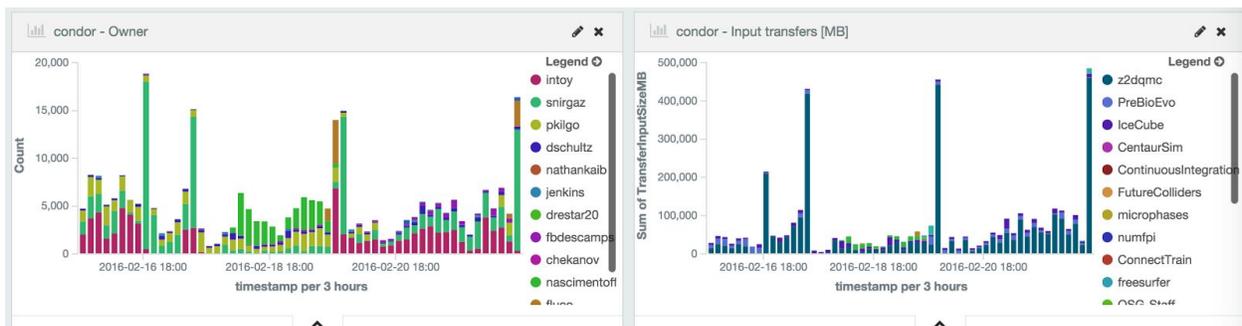
<https://fifemon.fnal.gov/gratia/dashboard/db/collector-status-osg>

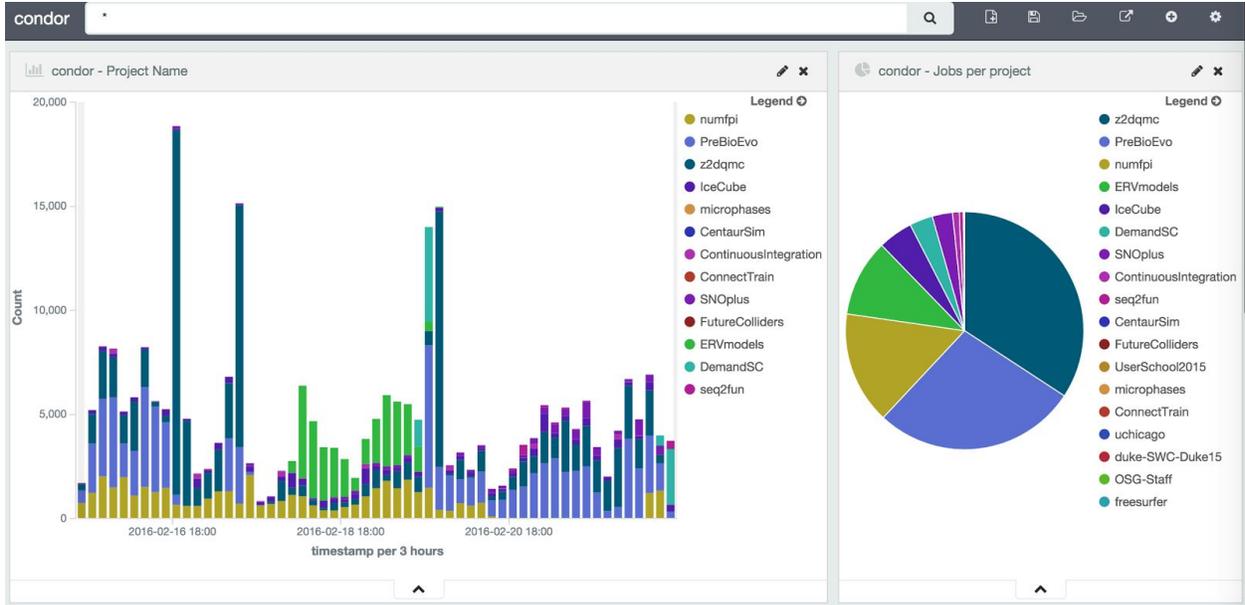


We are not proposing anything outrageously new:

### Elasticsearch of HTCondor history (OSG Connect)

- OSG Connect <http://bit.ly/1oZ1qvy> (atlas, analytics2015)
- [Class ads seen in last 30 days.](#)
- <http://fpaste.org/327527/85795145/> (job from OSG Connect)





#### OSG Connect use case:

- notify subsystem used to notify a small wrapper script whenever the HTCondor history log is rotated (every 10MB in our case).
- Behaves much like a cron:

```
# incrontab -l
```

```
/var/lib/condor/spool/history IN_MOVED_TO
```

```
/usr/local/bin/condor_elasticsearch.sh $# main
```

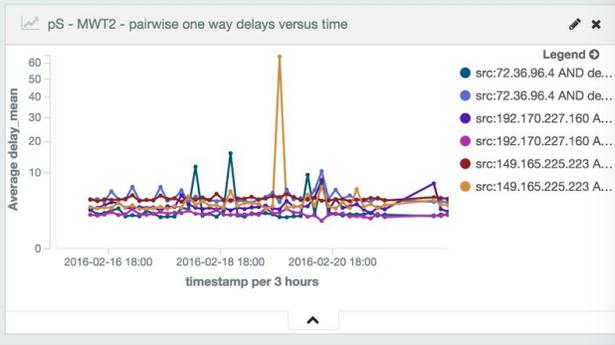
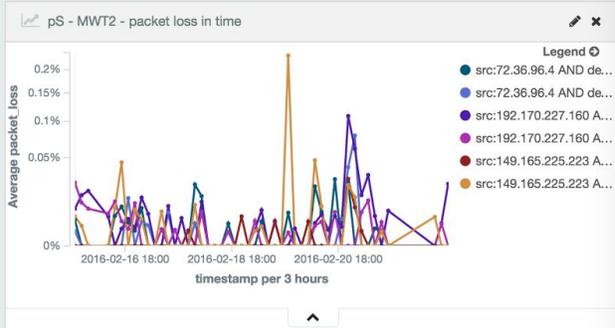
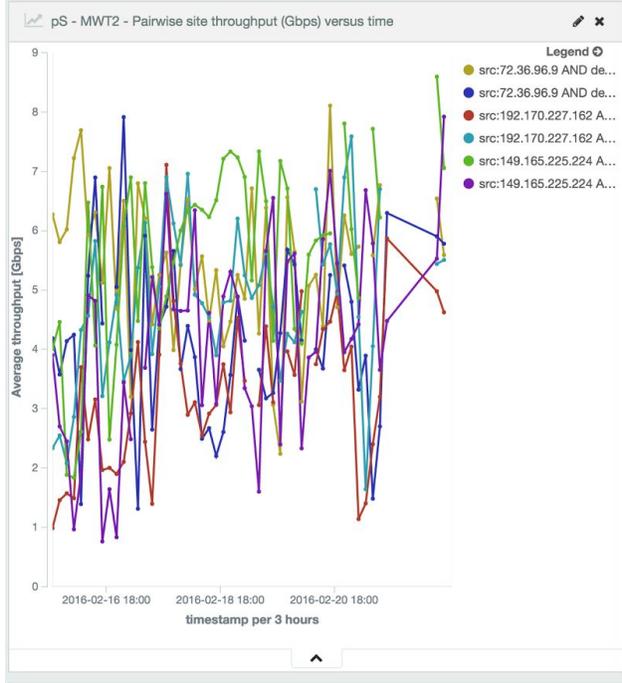
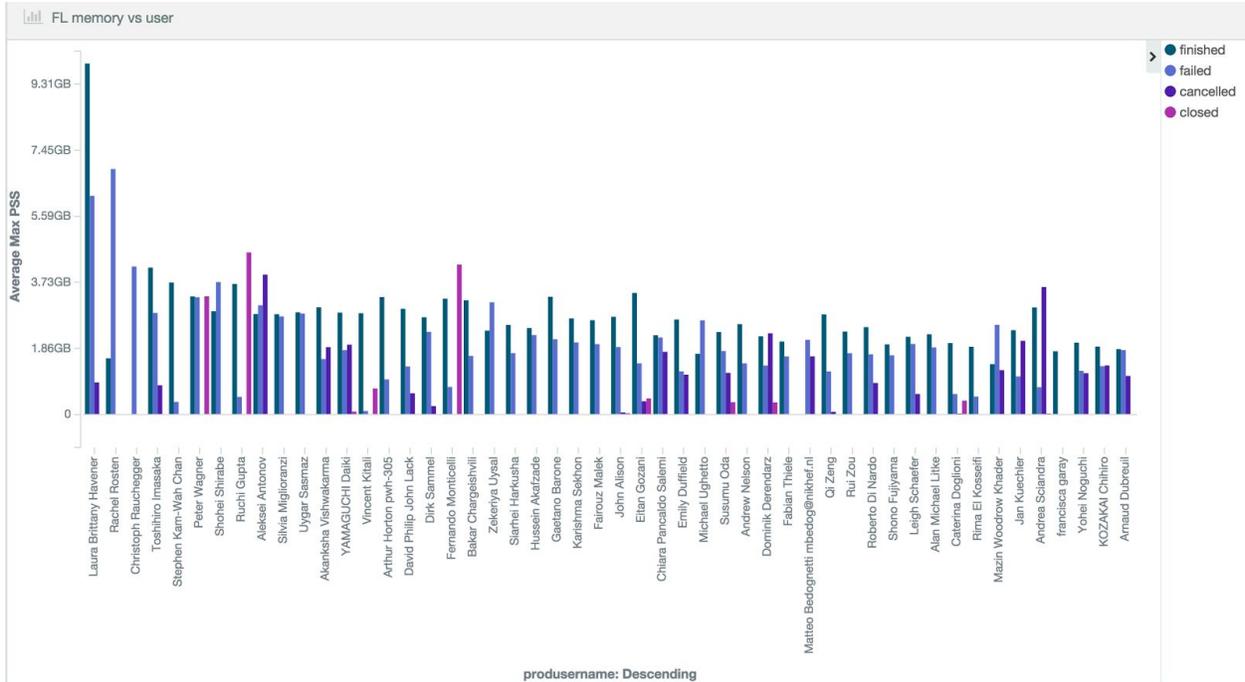
- Wrapper script takes history file name and queue name (multiple schedds on OSG Connect)
- Wrapper runs a python script to clean up and index data into elasticsearch
- Raw data is gzipped and archived on stash in case of lost indices.
  - 2-3GB/mo of compressed data at current job rate

## Elasticsearch in HEP

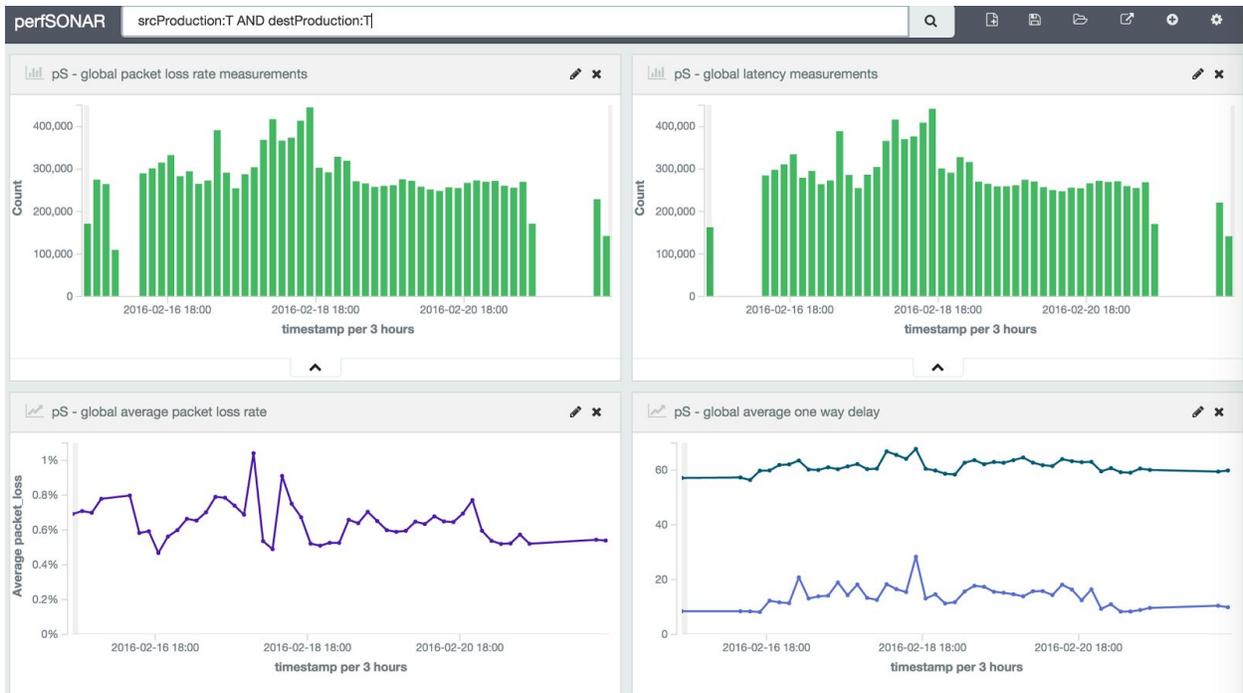
- ATLAS Analytics platform (running in CloudLab w/ indices backed up to MWT2)
  - <http://bit.ly/1L5XDGH> (recent talk), platform: <http://bit.ly/1XKu67f> (open)
  - 3B docs in 1084 indices spread over 9182 shards. Data size (single copy of it) ~3TB. We have at least 2 copies. Growing ~2GB a day. We are indexing much more but some large indices from Rucio are deleted after 10 days. Updates at 3kHz (max 10kHz seen).
  - WLCG perfSONAR dashboard (global) and regional federation dashboard (MWT2)
    - WLCG perfSONAR → OSG → CloudLab
    - <http://atlas-kibana.mwt2.org:5601/app/kibana#/dashboard/pS-MWT2>



<b>0.94</b> Average wall_time [h]	<b>32,868,787.533</b> Sum of wall_time [h]	<b>186.83</b> Max wall_time [h]	<b>1.641</b> Average queue_time [h]	<b>-1,293</b> Average currentpriority	<b>636.14MB</b> Average Max PSS	<b>175.11GB</b> Max Max PSS
<b>0.444</b> Average cpu_eff	<b>4.67GB</b> Average inputfilebytes	<b>15,294.93</b> Average nevents	<b>1.493</b> Average noutputdatafiles	<b>4.096</b> Average ninputdatafiles	<b>79.52MB</b> Average outputfilebytes	<b>34,973,428</b> Count
	<b>1,091</b> Unique count of producerid				<b>120</b> Unique count of computingsite	



Global (all WLCG sites) <http://atlas-kibana.mwt2.org:5601/app/kibana#/dashboard/perfSONAR>



- Other uses

- Rucio (ATLAS distributed data management system)
    - Error monitoring
    - Activity tracking
    - Tracking a file/dataset
  - Usage of beyond-pledge resources
  - Per cloud performance metrics
  - Data formats and dataset popularity
  - xAOD usage monitoring / analysis
  - FAX monitoring jobs accessing data over WAN
  - FAX redirectors monitoring
  - Monitoring storage resources
  - PanDA task duration analysis
  - Analysis of Analysis (distributed analysis user metrics)
- Prototyping: [UChicago](#): Early The ES instance at UC uses 4 Dell R410 nodes with 3TB of storage on each and with most indices set to replicate data to all the data nodes. The master and client nodes are VMs. With this setup we're pretty overprovisioned in regards to cpu (load is usually under 4 with 12cores/24 w hyperthreading). The data nodes have 48GB of memory which is sufficient (ES should use no more than 32GB RAM per instance. The rest is used by the OS for caching.. ES has 12TB raw at Chicago (with 2x replication).

- [CERN](#)
  - [Elasticsearch vs Oracle evaluation](#)
  - Used for Messaging service to monitor the status of their services. At the moment, there are more than two billion documents.
  - Will expand the usage of Elasticsearch to the three following areas:
    - Data transfer movements between the sites
    - Job processing
    - Status of the sites and services
- [dCache](#) (ELK)

<https://github.com/dCache/logstash4dcache>



- Tier-2 site for the ALICE collaboration at LHC (the Torino INFN CC) [is using ELK](#)
- [CMS DAQ](#)

- CMS has implemented a monitoring system for post-long shutdown 1 that complements the redesigned File-based Filter Farm and takes advantage of elasticsearch. It is able to provide a quasi-realtime full-detail insight into event processing information, using same mechanisms to inspect live run information as well as run history.

## If Elasticsearch is a solution we will need:

- Design how to integrate OIM information