| | |
|---|---|
| **Document Name** | Job Tracking on the Open Science Grid for the DZero VO |
| **Authors** | G.Garzoglio |
| **Document #** | cd-docdb 3129 |

| **Version** | **Date** | **Comment** |
|---|---|---|
| v0.1 | Feb 16, 2009 | Interview notes |
| v0.2 – v0.4 | Mar 2, 2009 | First Complete Draft |
| V1.0 | Mar 16, 2009 | Feedback Integrated |

# Job Tracking on the Open Science Grid for the DZero Virtual Organization

*Gabriele Garzoglio*

*March 16, 2009*

## Table of Contents

# 1. Introduction

In December 2008, Qizhong Li, head of computing for DZero, contacted the Computing Division to raise concerns on the usability of the Grid for DZero. The concerns could be grouped in three main categories:

1. **lack of monitoring**: users complain that tracking their jobs through the OSG infrastructure is difficult and, sometimes, impossible;
2. **lack of a resource procurement process**: in case of peak need, DZero users and the DZero computing leadership do not have an established procurement process, agreed with OSG.
3. **suboptimal selection of Grid resources for Monte Carlo jobs**: the SAM-Grid module that interfaces with the OSG Resource Selection System (ReSS) implements a load balancing algorithm that is too simplistic under many circumstances.

This document focuses on the concern over **lack of monitoring**. Its goals are

1. Document the concerns of DZero users for the main activities of Data Reconstruction, Montecarlo production, and Data Analysis on the Grid.
2. Recommend DZero strategies for mitigating any lacks of OSG-provided infrastructure. Provide recommendations based on practices of other Grid communities. For this document, we have interviewed experts and users from CMS, Atlas, OSG Engagement, and CDF.
3. Provide recommendations to OSG for possible directions to improve the users' experience on job tracking. DZero is responsible for discussing these recommendations further with the OSG Executive Board.

# 2. Executive Summary

This document analyzes three categories of monitoring, particularly relevant for the DZero activities (sec. 4): (1) job status monitoring from Grid middleware, (2) monitoring of the characteristics of the resources that run the jobs, and (3) monitoring of the internal status of running jobs. For DZero production, category 1 is the one that deserves most attention; for DZero analysis, categories 1 and 3 are the most relevant.

The main concerns from DZero regard (in priority order) (sec. 5): (1) the reliability of monitoring information (e.g. the Grid thinks that some jobs are still running, while they are not), (2) the lack of completeness of status information (e.g. Grid middleware gives very little information on why jobs are in a certain state), (3) the lack of an integrated portal for information display, (4) the slow propagation (timeliness) of information from the monitoring systems.

Some of these concerns can be mitigated by using currently existing technologies (sec. 6). We believe that the major benefits can be obtained by the adoption of a pilot-based workload management system, in particular GlideIn WMS, because of its quasi-interactive monitoring features.

The concerns reported by DZero are sometimes related to shortcomings of the OSG monitoring infrastructure (sec. 7). These include bugs and lack of sufficient diagnostic interfaces in the software stack, as well as services that are of interest but not offered by the OSG, such as a monitoring display.

In this white paper we also mention relevant aspects of how other VOs do monitoring, even if these are not directly applicable to the DZero use cases. Of particular interest are (1) the forwarding of CMS jobs to OSG via a gLite WMS in EGEE; (2) the tracking of the internal status of CMS running jobs via a VO-maintained MonaLisa service; (3) how the limited running time (a few hours) of VO Engagement jobs let them overcome potential job status monitoring problems by simply killing and resubmitting their jobs.

We conclude by thanking CMS, Atlas, CDF, and OSG Engagement representatives for their invaluable input.

## 3. The DZero Grid Infrastructure

The DZero grid infrastructure is more complex for production activities than for data analysis[1].

For **data analysis**, users submit jobs to OSG via a personal Condor-G. Currently, the only analysis jobs submitted to the OSG consist of a CPU intensive application, with minimal data I/O requirements. Local storage at sites is not required: input and output are handled through a GridFTP server maintained by the user. Jobs tend to run for 12 to 24 hours.

The current deployment is based on Condor 7.0.2

**Production activities** consist of two applications: (1) raw data processing and (2) montecarlo production. Both activities use the SAM-Grid infrastructure to handle job requests, data I/O, and monitoring (discussed in later sections). In short, users submit job requests to a SAM-Grid queuing node (based on the Condor scheduler) via a remote client (based on Condor client

[1] Data analysis has been discussed with Michael Wang; production activities with Mike Diesburg, for data processing, and Joel Snow, for Monte Carlo production.

commands). Jobs are matched and submitted to Execution sites (based on the Globus Gatekeeper / Job Managers). At the execution site, job requests are split into multiple job instances (typically a few hundred: e.g. 1 job for every input file for data reconstruction or 1 job for every 250 montecarlo events to be produced). These job instances can either be submitted to a local batch system or forwarded to another Grid, such as OSG. The execution site also triggers data delivery (binary, control, and input data) and controls data traffic shaping (typically involving SRM-based site-local storage). After output has been produced and (typically) locally stored, both applications run a merging step, to allow for more efficient long term storage of the results. Each application tends to run for 4 – 24 hours, with a few jobs running for as little as 2 hours and for as long as 6 days.

SAM-Grid forwards job instances to the OSG using a Condor-G scheduler, for queuing, and the OSG Resource Selection Service (ReSS), for match making. The current deployment for SAM-Grid and for the OSG client is based on VDT 1.10.1 (Condor 7.0.5).

The higher degree of complexity of the infrastructure used for production activities entails more complex monitoring scenarios. For production, monitoring of the SAM-Grid infrastructure needs to integrate with the monitoring services of the OSG. Today, this level of integration is only partial.

# 4. The Different Categories of Monitoring Information

This document analyzes problems encountered by DZero users with three main categories of monitoring information: (1) job status from Grid middleware, (2) general characteristics of the clusters and machines running DZero jobs, and (3) internal status information of running jobs.

## 4.1. Job Status Monitoring from Grid Middleware

In modern Grid systems, several middleware components contribute to the management of a job. For example, when submitting a job to the OSG, a chain of components are involved in dispatching the job to its final running environment (Worker Node). These components include client commands, queuing services (e.g. Condor-G Scheduler), computing resource gateways (e.g. Globus Gatekeepers), cluster job schedulers (e.g. PBS, Condor, ...), etc.. For more complex systems, such as the SAM-Grid, the chain is even longer. These components keep a record of the job[2] and are involved when the user enquires about the job status.

[2] Client commands are typically not persistent, thus, strictly speaking, they do not keep a record of the job.

Ideally, each of these components should be able to provide **a short description of the job status** (e.g Idle, Running, Completed, …), as they know it, and **why the job is in that state**. Today, this information is only partially available. For example, when the short description of the job status is "Idle" in the Condor-G Scheduler, is it because no resources can match the job requirements, or because the system has surpassed the total number of jobs allowed by policy for that resource, or because the system has not run a match making cycle yet, or because… Some commands, such as *condor_q –analyze*, only provide partial explanations to this question, especially for the Condor-G system. Having better diagnostic interfaces should allow users to know how far in the middleware chain a job is and for how long the job will have to wait before running.

In this category of monitoring information, we also include aggregate job status information. That is, the total number of jobs in a certain state, as known by a middleware component**. For DZero users, a particularly relevant monitoring metric is the total number of idle and running DZero jobs at each remote cluster batch system**. These aggregate statuses are typically obtained from the systems monitoring the resources (sec 4.2). The main problem with these systems consists in the (possibly perceived) poor reliability of the information. An alternative mechanism consists in querying all schedulers that manage DZero jobs and aggregate the results. This works reasonably well for production applications, which are all submitted using the SAM-Grid system. As new analysis groups start using the Grid, it is difficult to guarantee that the "aggregator" (e.g. *condor_q –global*) knows about all job queues, i.e. that the aggregate information includes all jobs. In the past, this category of information was made available to users through the MonaLisa service. The service was considered by users reliable, timely, and information well presented.

## 4.2. Monitoring of the Characteristics of the Resource that Runs the Job

The systems in this category **monitor characteristics of the remote clusters and/or machines** that run the jobs. Examples of this information include worker nodes metrics such as amount of memory, CPU load, and local disk space. Examples also include cluster metrics such as computing gateway contact information, total number of available job slots, storage gateway contacts, and opportunistic storage size.

The OSG provides this information through the Generic Information Providers (**GIP**), a series of scripts that run at the remote resource. The information follows the **Glude Schema** and is published by the CEMonitor service (**CEMon**) to two central systems, each handling a different format:

1. the Berkeley DataBase Information Index (**BDII**) describes site information using LDIF format (a structured information tree);

2. the OSG Resource Selection Service (**ReSS**) uses set of Condor classads (lists of attribute / value pairs);

The problem with these information systems is the perceived low reliability of the information produced by GIP. In reality, as discussed in sec. 5.1.2, the quality of the information has much improved since the initial user experiences.

Other information systems in OSG are not strictly monitoring system, as they deal with registered / static site information (VORS / OIM), alarming conditions (RSV), and job accounting (Gratia).

Pilot-based Workload Management Systems (WMS) (sec. 6), such as **Panda** or **GlideIn WMS**, provide operating system-level information about the worker node that runs the job. Today, most OSG sites accept pilot jobs, but the Pilot-based WMS infrastructure must be maintained by the Virtual Organization. This paradigm may change in the future, should OSG charge a Facility to maintain a common WMS infrastructure for multiple VO [3].

**For DZero, especially for analysis users, particularly interesting metrics are the total number of available CPU slots and the number of idle and running jobs for DZero and for all other VOs.** As for job status monitoring, users found the MonaLisa system particularly helpful for this category of information. In general, however, human consumption of this category of information is deemed less crucial for operations than the monitoring of job statuses (sec. 4.1).

It should be noted that for production activities, the SAM-Grid is integrated with this information through ReSS, for automatic resource selection.

### 4.3. Monitoring of the Internal Status of Running Jobs

This category of monitoring **allows users to know the internal status of a running application**. On the OSG, it is typically achieved in two ways:

1) **Integrating the application with monitoring libraries**. These libraries send messages to a central monitoring system, when the application reaches internal milestones. For example, this mechanism is used by USCMS. USCMS instruments its applications with MonaLisa libraries. The VO maintains a dedicated MonaLisa server, which receives

[3] This paradigm is used in the case of VO management services, for which FermiGrid maintains VOMS and VOMRS instances for multiple OSG VOs.

information from the applications and display them to the USCMS Monitoring Dashboard.

2) **Looking at application log files**, as they are written on the local system. For example, this mechanism is used by CDF and USCMS through the facilities offered by the GlideIn WMS system.

In general, other mechanisms include querying directly interfaces exposed by the application over the network. These are not popular on the Grid for the presence of firewalls[4] and for the difficulty in coding such interfaces.

This category of monitoring has the problem that, by design, it gives information about an application only if it is running. It should be noted that this category of monitoring is complementing, rather than substituting, job status monitoring (sec. 4.1).

DZero analysis users would be particularly interested in this kind of monitoring. To achieve this goal, we recommend the adoption of the GlideIn WMS system (sec. 6). DZero production, instead, already supports this category of monitoring. The SAM-Grid system, in fact, wraps DZero applications with programs that send status information to dedicated XML Databases. The same XML Databases are also used by the Runjob system, a workflow engine that prepares the environment for and wraps the DZero applications.

## 5. Problems and Desired Properties of the Monitoring Infrastructure

This section discusses different properties of job status monitoring, such as timeliness, reliability, presentation, and completeness of information. For each property, we discuss known issues and, where appropriate, expected behavior, as discussed with DZero Grid users.

It should be noted that users have formed their expectations on how a monitoring system should behave by using the MonaLisa system. Deployment of MonaLisa on the OSG is no longer recommended by default because its core engine is not open source and because of a reported high load in administrative maintenance. This means that OSG does not require that sites install the product. The system, however, is still available in the VDT distribution, it is still deployed by some sites, and a central MonaLisa repository is maintained by the OSG Grid Operations Center.

For monitoring the internal status of running jobs, the OSG MonaLisa repository could be used, if applications are properly "instrumented" (sec. 4.3); however, using a GlideIn WMS

[4] Typically, OSG clusters do not allow incoming network connections to the worker nodes.

system is another solution, proposed later on (sec. 6.1). For monitoring the status of jobs and resource, the sparse deployment of MonaLisa services at sites is probably not a solution to the problems faced by DZero. In any case, OSG is open to discuss further the usage of MonaLisa by DZero.

In summary, the high-level problems with monitoring for DZero are

- Individual middleware components do not offer interfaces to obtain complete and reliable job status information (short status & reasons for that status).
- Information from resources (GIP) is perceived as unreliable.
- OSG does not provide a system that integrates all of this information in a single coherent location.

## 5.1. Reliability

All DZero Grid users report problems in the reliability of monitoring information. In particular, the components affected seem to be (1) Condor-G / Globus Gatekeeper communication, (2) Grid Information Providers (GIP), and (3) SAM-Grid monitoring.

### 5.1.1. Condor-G / Globus Gatekeeper

All DZero Grid users occasionally observed that a job status reported by Condor-G was inaccurate when compared to the status reported directly by the batch system running the job. For example, some jobs are reported in Condor-G as “running” for days, while the batch system has completed the jobs days before. The same happens for jobs reported in Condor-G as “idle”.

This problem seems to occur in Condor-G v7.0.5 (VDT 1.10.1) when interacting with the Globus Gatekeepers from the Globus Toolkit pre-web services v4.0.5 (VDT v1.8 and v1.10). This effect might be caused by multiple problems. For example, recent investigations have uncovered a problem in the Globus job-manager for the LSF batch system (VDT Ticket #5009; Globus Ticket #6688). However, there is anecdotic evidence of this problem also on PBS systems, such as the DZero CAB cluster at FermiGrid. For completeness, it should be mentioned that USCMS users do not see this issue, but they use a different version of Condor-G (v7.2).

To mitigate this problem we recommend that the OSG Software Tools Group works with DZero and the relevant external software providers (Condor and Globus) to fix possible bugs in Condor-G v7.0.5 and/or Globus.

### 5.1.2. Grid Information Providers

DZero users are interested in aggregated job status metrics from Grid sites for their VO. For example, the total number of DZero jobs that are running, idle, etc. at the given Grid site. Users are also interested in resource characteristics, such as available job slots. The MonaLisa system had a reputation for reliability in providing this information.

Today on the OSG, this information is available from sites via the Grid Information Providers (GIP) in the Glue Schema (VOView and VOInfo entities). This information is published to the ReSS and BDII systems (sec. 4.2).

DZero Grid users have expressed concerns with respect to the reliability of the information from GIP. While in the past some concerns were well founded, in recent years the quality of the GIP product has improved considerably. In addition, GIP is an actively maintained product: bug reports can be filed to the Grid Operation Centers (GOC). It should also be noted that GIP is also highly important for OSG / LCG interoperability in CMS activities.

To mitigate DZero's concerns, we recommend that DZero users try to use information from GIP e.g. via ReSS (*GlueCEState** attributes), after the deployment of the OSG v1.0 update (to be released on March '09). Such information can be obtained from command line tools, such as *condor_status* [4].

### 5.1.3. SAM-Grid

DZero users have reported occasional problems in the reliability of job status information in the SAM-Grid system. The system is implemented on top of Condor-G and Globus and it may suffer from the same problems in communication of Condor and Globus Gatekeeper (sec. 5.1.1).

In the past year, the Computing Division has been sponsoring projects dedicated to improving the quality of production operations through the SAM-Grid. These projects represent the best venue to raise the reliability issues, so that the appropriate priority can be given to each problem.

## 5.2. Presentation

DZero users have reported that the graphical representation of job and resource status information would be useful for their operations. In particular, plotting system metrics vs. time would be useful to spot potentially problematic trends. A graphical representation of job and resource status would also give a feeling of the system at a glance. Such representations could include graphs of the number of jobs in a certain status (idle, running, etc.) for single sites or for

the whole Grid. Currently, OSG does not offer this service. In any case, graphical interfaces should always be provided together with command line tools.

Users also asked to put more of the information already available together in a single display. The Computing Division has recently started the Metrics Correlation and Analysis Service (MCAS) project [1] to address some of these concerns. Also, the MyOSG project [2] provides web interfaces to different operational-oriented information available in OSG. MyOSG also provides the ability of exporting such information and arranging it in personal web portals (e.g. iGoogle).

## 5.3. Completeness

Ideally, for each of the middleware components involved with job handling, job status monitoring should provide a short job status description and a reason for that status (sec. 4.1). In this area, DZero users' complaints center on OSG job handling components.

### 5.3.1. Job Handling Components

Middleware components do not satisfactorily report the reasons why jobs are in a certain status. In particular, when jobs are "idle" waiting for resources, it is not clear what these resources are. Another case is when jobs are "held" (or "failed" in SAM-Grid terminology). Better diagnostic interfaces should be developed for all Grid middleware systems, in particular for Condor and Globus.

A mitigation strategy could be for DZero operations to integrate queries to some status inspection interfaces, e.g. using *condor_q –analyze*. However, users should be warned that this command gives only partial information for the use cases of Condor-G. For example, only log files can tell if no more jobs can be sent to a remote Globus Gatekeeper because the maximum number of jobs at a single cluster has been reached[5].

For production activities, users can use *condor_q –analyze*, querying remotely the forwarding node with

*condor_q <job_id> –pool osg-ress-1.fnal.gov –global –analyze*

where <*job_id*> can be obtained from the web monitoring (http://samgrid.fnal.gov:8080) in the following way. From the main page, click "submission", then click the name of the queuing

[5] For DZero production, this limit is set to 1250 jobs.

server that holds the grid job requests (e.g. samgrid2.fnal.gov); click on the status of the grid job request of interest, then on “Remote Monitoring”. The page shows a list of <*job_id*> (e.g. 133025), their short status, and other metadata.

For analysis, one would query their local Condor-G installation with

*condor_q –analyze <job_id>*

Using Condor-G, typically the command does not help much for jobs in “idle” state. In this case, we recommend the usage of *condor_q –globus*. This command allows to distinguish between jobs idle in the Condor-G queue and idle in the remote batch system.

### 5.4. Timeliness

In large distributed systems, information delays are inherent with almost every architectures. On the other hand, especially in complex systems, the ability to timely spot emerging trends in system metrics is an effective tool in preventing disruption of service. In addition, timely information helps with operations, giving users necessary feedback on what to do next.

For DZero production, users believe that a delay of 10 – 15 minutes in the propagation of their job status is appropriate for their operations. For analysis, ideally this delay should be 5 minutes or less. When operations are not automated, humans need to wait for the information to propagate before moving on to the next step in their operations: this makes timeliness of information more pressing. In general, timely information is needed for monitoring dangerous trends with the system.

Currently, information from the resources (GIP) is published every 5 minutes to BDII and every 10 to ReSS. These publication times are configured at each site (CEMon configuration). Shortening these times has the effect of increasing the load of the Computing Elements and of the receiving servers. These times can be shortened if these machines are capable to handle the increased information flow. In any case, when using Condor-G to submit jobs, the mechanism with the shortest delay to know the status of the job is the command *condor_q –globus* (by default, this is 5 minutes and is controlled by the condor configuration parameter CONDOR_JOB_POLL_INTERVAL).

DZero users have reported delays of 30 minutes in the propagation of their job status through the system. These delays make operations challenging. Having a system that bypasses the current chain of status propagation may help in this case. A pilot-based workload management infrastructure has this advantage (sec. 6).

# 6. Existing Mitigating Solutions – Input for DZero

Throughout this document, we have given recommendations to DZero users on how to improve their monitoring experience on the Grid. In this section, we collect those recommendations and discuss pros and cons of adopting a pilot-based workload management system.

The immediate recommendations for analysis and production users are:

1. To improve resource monitoring and aggregate job status monitoring, we recommend reevaluating the reliability of the information from the Generic Information Providers (GIP), for example through ReSS (sec. 5.1.2). Potential bugs and site configuration problems should obtain a good level of attention when reported through Grid Operation Center (GOC) tickets.
2. To improve job status monitoring, in particular the reasons behind a job status, use *condor_q –analyze* and *condor_q –globus* (sec. 5.3.1).
3. To improve job status monitoring, worker node monitoring, and internal status monitoring of running jobs, we recommend the adoption of a pilot-based WMS technology (see 6.1).
4. To improve job status and resource monitoring, DZero should have a focused discussion with OSG on using MonaLisa more widely at OSG sites. At this time, it is probable that many sites will not provide MonaLisa services anymore (sec.5).

The recommendations applicable in about 6 months are:

5. To improve job status monitoring, Condor has improved the efficiency of the command that provides aggregate job status information (e.g. number of jobs in a certain state). In particular, Condor-G v7.3.x has improved the *condor_status –globus* command. We recommend that this command be evaluated, as soon as Condor-G v7.3 becomes available through VDT.
6. To improve the presentation of all information, DZero users should evaluate the MCAS project in the near future.

## 6.1. Pilot-based Workload Management Systems (WMS)

The core idea of pilot-based Workload Management Systems consists in automatically procuring Grid nodes and making them available to users as if they were part of a single batch system.

In short, the system works as follows. Users submit their jobs to a VO queuing service. In response, a pilot-based WMS component, sometimes called Pilot Factory, submits “pilot” jobs to

the Grid through the standard Grid Resource Gateways (e.g. Globus Gatekeepers). When running, pilot jobs have three main responsibilities:

1. check the sanity of the remote execution environment on behalf of the VO;
2. register the resource with its characteristics to the VO resource pool;
3. receive a user job through internal reliable protocols (i.e. bypassing the standard Grid channels) and run it.

We recommend the adoption of a pilot-based workload management system, such as GlideIn WMS, to DZero Analysis first. For the use case of analysis, in fact, little or no integration of GlideIn WMS with other systems is necessary. Analysis has also a smaller volume of jobs than production and, therefore, requires a simpler system configuration. The use of a Pilot-based WMS solution can then be transferred to the Production use cases as well. For production, GlideIn WMS needs to be integrated in the SAM-Grid infrastructure. The effort required is roughly estimated to 1 FTE month for development and start up operations.

Pilot-based WMS systems are currently used by several VOs, including Atlas (using the Panda WMS), CMS, CDF, and Minos (all using GlideIn WMS). As reported by representatives of these VOs, pilot-based WMS systems have several facilities to improve job and resource monitoring. The rest of this section discusses them.

**Benefits of Pilot-based WMS for Job Status Monitoring**

- With a pilot-based WMS infrastructure, users are effectively isolated from most of the Grid Middleware, including Computing Gateways and batch systems at sites. In the experience of VOs adopting the technology, users do not need to track jobs through several layers of Grid Middleware. User's main concern is whether there are not enough pilot jobs running, i.e. enough allocated resources, to run the user's jobs.
- User's job status is made available through robust mechanisms to the VO resource pool system (e.g. Condor batch system). In GlideIn WMS, to (at least partially) understand the reasons why a user job is in a certain state, users can run batch system status diagnostics commands, such as *condor_q –analyze*. In this environment, the command works better than for Condor-G.
- GlideIn WMS provides a fully integrated graphical diagnostics system for the health status of pilot jobs. This system helps diagnose problems accessing the standard Grid resource gateways.

**Benefits of Pilot-based WMS for Resource Status Monitoring**

- Once a Pilot job runs at a remote resource, it registers the node where it is running with the VO resource pool. This registration includes dynamic information about the characteristics of the Worker Node, such as available CPUs, Memory, System Architecture, etc. Users can have detailed information about the allocated Grid nodes by querying their resource pool servers.

**Benefits of Pilot-based WMS for Monitoring Internal Status of Running Jobs**

- GlideIn WMS allows for quasi-interactive execution of commands at remote nodes. This feature allows running unix commands such as *ls*, *ps*, *top*, and *cat*. Effectively, these commands are short monitoring jobs. These are dispatched to the same machine running the user job through the same internal channels used for dispatching user jobs. This feature can be useful to get the status of a user job, by looking at parts of a local log file. In particular, this would fit well with the current DZero analysis use case. To provide an integrated monitoring display, CMS is working on a Graphical User Interface to display results of typical quasi-interactive commands (*ls*, *top*, etc.). This display is not yet available for production usage.

**Other Benefits**

Besides monitoring, a pilot-based WMS infrastructure provides other operational benefits. Many of these are discussed elsewhere [3]. This are the benefits explicitly mentioned by the VO representatives interviewed:

- Failures of pilot jobs submission affect users operations only in the availability of computing capacity. In other words, users do not need to resubmit / recover any of the user's jobs when pilot job submissions fail.
- Pilot jobs failures are arguably simpler to diagnose, because pilot jobs consists of standard (typically) short code. VOs, such as Atlas, find that this simplicity encourages help from site administrators.
- OSG and its facilities are discussing the possibility of delegating the maintenance of pilot factories for various VOs to one OSG facility. This operational model is efficient because support personnel are more experienced than normal users in tracking middleware failures in (pilot) job handling. This operational model would lower the entry cost for a VO for using Grid facilities. The representative of Atlas reported that this model is successfully working for their operations.

## 7. Possible Infrastructural Improvements – Input for OSG

Throughout this document, we discuss how some monitoring problems could be addressed with the involvement of OSG. This section collects this input and presents some more ideas for improvements. This section should be considered as a set of recommendations that DZero can bring up to the attention of the OSG Executive board.

- Some of the major monitoring problems reported by DZero are related to the low reliability of the communication between Condor v7.0.5 and Globus v4.0.5 (sec. 5.1.1). This may be related to multiple issues (see VDT Ticket #5009 / Globus Ticket #6688 for one of them). We recommend that the OSG Software Tools Group facilitates the interaction between the external software providers and the VO, to address this issue.
- DZero users are interested in graphical representations of job and resource status (sec. 5.2). These representations may include plots of number of jobs in a certain status (running, idle, etc.) vs. time at each site or for the whole VO. These plots would be helpful in spotting trends of the system before error conditions arise. Similar plots are available for the Gratia accounting system, but only after jobs are finished. Metrics of interest are already available from various OSG information systems (BDII, ReSS, etc.), but no service accesses this information to create any display. We recommend that the OSG Software Tools Group investigates possible technical solutions for this request.
- DZero users have reported that the monitoring information already available is often scattered throughout several web pages (sec. 5.2). DZero would be interested in composing relevant metrics in a single display. Recently, the Fermilab Computing Division has started the MCAS project to address similar needs [1]. We recommend that OSG investigates MCAS or a similar solution for its users.
- Many services of the OSG software stack do not provide sufficient diagnostic interfaces for a user to understand (sec. 5.3.1):
    - why a job is in a certain status;
    - why a job has disappeared from the system;
    - for how long a job still needs to wait to run.

  Some software tools already provide limited diagnostic capabilities. For example, Condor provides the *condor_q –globus* and *condor_q –analyze* commands. In particular, *condor_q –analyze* works well for the condor batch system, but it could be much improved for Condor-G. We recommend that the OSG Software Tools Group works with the relevant External Software Providers to improve diagnostic interfaces.
- We observe that pilot-based infrastructures offer multiple benefits to users (sec. 6.1). Some of these benefits overcome shortcomings of the OSG monitoring infrastructure:
    - lack of information reliability through the standard resource gateways

- lack of support for reporting the status of a running application (previously supported through MonaLisa)
- lack of diagnostics interfaces for the standard job handling Grid middleware (Condor-G / Globus)

Instead of focusing on addressing some of these issues directly, OSG could give priority to outsourcing the operations of a pilot-based infrastructure to a member facility, in order to facilitate the usage of such technologies by OSG VOs.

- Some of the interviewed representatives find informative the EGEE GGUS messages on the availability of individual services at sites. In general, GGUS messages are considered more informative than the ones provided by the OSG GOC. The GOC informs registered users of central services downtimes and of site downtimes. OSG could investigate what messages users find most interesting in GGUS and improve GOC communications.

## 8. Experience from Other VOs

This section collects some the experience in job monitoring, which representatives from Atlas, CMS, CDF, and OSG Engagement shared with us. This experience is not directly applicable to the DZero use cases, but might still be of general interest. The experience that was applicable to DZero has been already integrated with the sections above.

**Job Status Monitoring from Grid Middleware**

- USCMS job submission to the OSG is done differently from other VOs. USCMS submits jobs to the OSG through an LCG gLite WMS (Resource Broker). This is possible because all OSG sites are advertised to LCG thanks to the interoperability of the information systems. This indirect job routing has the advantage that user of the gLite WMS can select resources based on the presence of a dataset. Such data-driven resource selection is favored by most users. Another advantage is the integration of all LCG and OSG resources under a single interface. Also, when running on EGEE, the infrastructure provides additional job status information through the Logging and Bookkeeping service. The disadvantage of such a mechanism is an increased time lag in communication and a higher degree of complexity, when debugging job handling problems. In the near future, this mechanism will change in favor of using GlideIn WMS with both the analysis (CRAB) and production (ProdAgent) infrastructures.
- The OSG Engagement VO depends less critically than other VOs on a job monitoring infrastructure. In fact, all of their jobs consist of short-running jobs (a few hours); therefore, when a job takes too long to run (> 10 hours), instead of trying to diagnose whether the status is reported correctly, the system automatically kills it and resubmits it. From that moment on, that resource will be penalized in future job / resource matches. This strategy works because

the VO only uses opportunistic resources from a pool of many OSG sites. During normal operations, OSG Engagement users can access the status of their jobs through the *condor_grid_overview* command. This is an in-house development that aggregates job status from *condor_q* and *condor_status* commands.

**Monitoring of the Characteristics of the Resource that Runs the Job**

- CMS, Atlas, and VO Engagement users tend not to look at this information. All of these VOs heavily rely on automated resource selection systems.
- CDF experienced problems in the past using this category of information to select resources. In particular, their system would occasionally select a CPU so slow that their job (typically a MonteCarlo) would surpass the eviction time limit. Integration with the new GlideIn WMS is expected to address this problem.

**Monitoring of the Internal Status of Running Jobs**

- CMS instruments all of its analysis jobs with the MonaLisa libraries. These send messages when the job reaches a milestone to a MonaLisa server, maintained by the VO. This information is then displayed in the CMS dashboard.
- CDF has found that the pseudo-interactive monitoring provided by their pilot-based infrastructure is adequate to address monitoring of running jobs.
- Atlas and OSG Engagement users are not particularly interested in this category of monitoring.

## 9. Acknowledgments

This document could not have been written without the patience and availability of the users that I have interviewed. Their input on their monitoring use cases has been of fundamental relevance. In particular, I want to thank Mike Diesburg, Joel Snow, and Michael Wang from DZero; Burt Holzman from CMS; Donatella Lucchesi from CDF; Maxim Potekhin from Atlas; and Mats Rynge from OSG Engagement.

## 10. References

1. The MCAS Project: http://www.fnal.gov/docs/products/mcas/
2. MyOSG Project: http://myosg.grid.iu.edu/
3. Wuerthwein, F., “Arguments in favor of a ‘pull model’”, OSG document 93. http://osg-docdb.opensciencegrid.org/cgi-bin/ShowDocument?docid=93
4. ReSS user interface examples https://twiki.grid.iu.edu/bin/view/ResourceSelection/ReSSUserInterfaceTools