Pilot Job Accounting and Auditing in Open Science Grid

Igor Sfiligoi
Fermilab, Batavia, IL, USA
sfiligoi@fnal.gov

Greg Quinn
The University of Wisconsin, Madison, WI, USA
gquinn@cs.wisc.edu

Chris Green
Fermilab, Batavia, IL, USA
greenc@fnal.gov

Greg Thain
The University of Wisconsin, Madison, WI, USA
gthain@cs.wisc.edu

Abstract

The Grid accounting and auditing mechanisms were designed under the assumption that users would submit their jobs directly to the Grid gatekeepers. However, many groups are starting to use pilot-based systems, where users submit jobs to a centralized queue and are successively transferred to the Grid resources by the pilot infrastructure. While this approach greatly improves the user experience, it does disrupt the established accounting and auditing procedures. Open Science Grid deploys gLExec on the worker nodes to keep the pilot-related accounting and auditing information and centralizes the accounting collection with GRATIA.

1. Introduction

Job accounting and auditing information is important both for the economics of the Grid and for detecting anomalous behavior. However, while the original Grid authentication, authorization, accounting and auditing mechanisms were designed under the assumption that users would submit their jobs directly to the Grid gatekeepers, direct submission accounted for only a tiny fraction of job submissions. With the proliferation of Grid sites, most users prefer to submit their jobs to an intermediate queue and have a workload management system (WMS) distribute their jobs among the Grid sites.

Over the past few years, many groups have started to use pilot-based WMSes for their ability to keep Grid-wide fair share between their users. These systems do not submit the jobs directly to the Grid gatekeepers, but send only so-called pilot jobs. Once a pilot job starts on a Grid resource, it will fetch a real user job and execute it. The traditional Grid authentication, authorization, accounting and auditing mechanisms are not used by the pilot, subverting the established accounting and auditing procedures. Examples of pilot-based WMSes are DIRAC[1], glideinWMS[2] and PanDa[3].

Open Science Grid (OSG)[4] addresses these problems with gLExec and GRATIA.

2. Accounting and auditing problems of the traditional Grid model

In the traditional Grid model, the site Grid gatekeeper is responsible for authenticating and authorizing a user based on the provided X.509 proxy certificate. If a user is accepted, his/her job is submitted to a local batch system that handles the job from that moment on. The accounting and auditing systems monitor the gatekeeper and the batch system activity, extracting the needed accounting and auditing information. See Figure 1 for an overview.

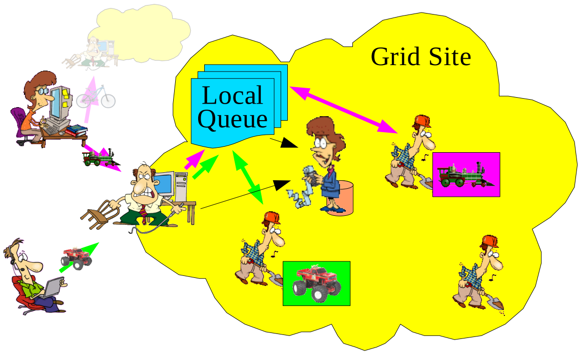

Figure 1. Traditional Grid workflow

Pilot-based WMSes have quite a different workflow. Here, users submit jobs to a WMS specific batch system and their presence triggers a pilot factory to send pilot jobs to various Grid sites. The pilot jobs then follow the path user jobs would and eventually start on a set of worker nodes. Each pilot job then pulls a user job from the WMS batch system and executes it. See Figure 2 for an overview.

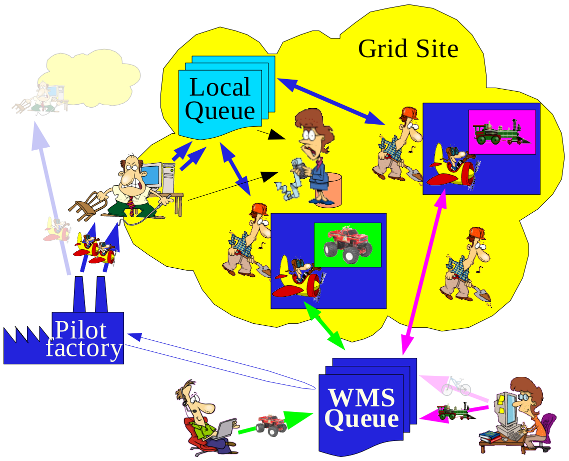

Figure 2. Pilot workflow in a traditional Grid setup

However, the accounting and auditing systems are still only monitoring the site provided gatekeeper and batch system. They cannot distinguish between resources used by the final user jobs from resources used by the pilot infrastructure. Thus all the processes will result as being ran by the pilot user and the resources will be accounted to the pilot itself. This is often unacceptable from both the pilot owner's and the site administrator's points of view.

Having accurate per-user accounting information can be important when justifying the requisition orders. Many sites receive financing both from VO sponsors and from local community sponsors. A site may need to demonstrate that local users actually used the resources sponsored by the local community.

Having detailed auditing information is also essential. If a process is found to be performing unauthorized or illegal activity, the responsible person must be accurately identified. Sites usually need to be able to determine this information without involving the pilot owner. At the same time, the pilot owners do not want to be held responsible for the actions of users they are serving.

3. Installing a gatekeeper on every worker node

In order to get proper accounting and auditing information, pilot jobs need to inform the local security system when running a user job. However, pilot jobs cannot be blindly trusted by the Grid sites, so the sites need a trusted, local tool with the following requirements:

- Resources will be accounted to a user if and only if a pilot job is able to demonstrate that that user entrusted it with his/her job. Possession of a valid users's X.509 proxy certificate is the minimum requirement.
- The tool must be able to automatically compute the user job's accounting information and distinguish between user and pilot processes for auditing purposes. In other words, the tool must not rely on the pilot job to provide this information.

OSG has started deploying gLExec[5], a X.509-aware derivative of the Apache suexec[6], on its worker nodes. gLExec is a privileged executable that, given a X.509 proxy certificate, authenticates and authorizes the user and runs the associated user job under the appropriate local identity, allowing for reliable auditing and accounting. In other words, it is like having a gatekeeper on every worker node.

Once gLExec is deployed on the worker nodes, pilot jobs can use it to launch the users' jobs. With the accounting and auditing systems now also monitoring the *"gatekeepers"* on the worker nodes, the accounting and auditing information is once again correct. See Figure 3 for an overview.

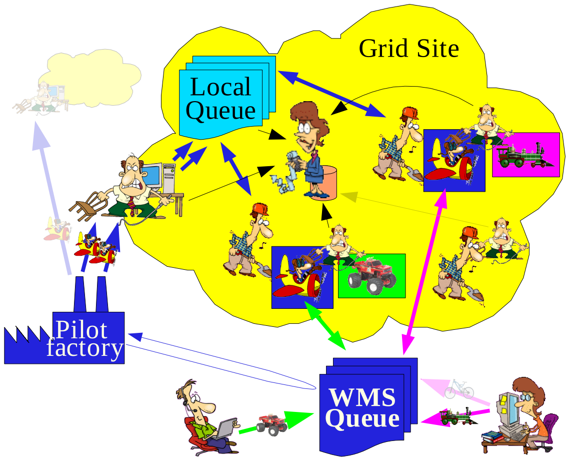

Figure 3. Pilot workflow with WN gatekeepers

4. Accounting and auditing information provided by gLExec monitoring in OSG

On OSG resources, gLExec is configured to interface to the GUMS[7] authorization and mapping system, and is using a OSG-specific monitoring process that is launched at user job startup. The monitoring process logs both the information involved in the authorization and the information about the user job's processes. All of these data can be used for auditing and accounting purposes.

4.1. User authorization information

gLExec logs all the invocation attempts. If a valid X.509 proxy certificate is presented, the following information, the same that is sent to GUMS, is also logged:

- The X.509 Subject, also known as the Distinguished Name (DN).
- The Fully Qualified Attribute Name (FQAN) of an eventual VOMS extended attribute.
- The VO name and Issuer DN of the FQAN in the VOMS extended attribute.

If GUMS authorizes the user, the following information is also logged:

- The UNIX User Identifier (UID) the user job processes will run as.

4.2. Collecting job information

Once a user job starts, the monitoring process starts tracking the job process tree and measuring its CPU usage.

In general, tracking process families can be difficult to do reliably. The OSG gLExec monitoring relies on the Condor[8] procd daemon, that uses a novel approach of dedicated tracking secondary Group Identifiers (GIDs) to achieve the goal. A detailed description of the process is presented in the next section.

The monitoring process logs accounting and auditing information of long running processes, defined as lasting more than 5 minutes. Shorter lived processes are not considered important enough to warrant the additional log space. Moreover, auditing and accounting information of the job as a whole are logged, too.

The auditing and accounting data logged for each process/job are:

- Start time and end time, effectively obtaining the wallclock time used by the process/job.
- CPU used by the process/job, split between CPU used in user state and CPU used in system state.
- The parent process id of the monitored process.

The information is collected by polling the process tree at regular intervals, currently fixed at 1 minute. Since only long running processes are logged, no interesting auditing information is lost. However, the process accounting information can be underestimated by up to one minute.

It is also worth noting that while the total wallclock time is always accurately reported, the job total CPU usage can be significantly underreported if the user job spawns a large amount of very short, CPU intensive processes, and does not wait for them to finish. Although this is a real problem, it was not deemed worth the additional monitoring load necessary to properly handle it.

5. Reliable process family tracking

Tracking process families can be difficult to do reliably. The basic approach is to use process parent-child relationships as shown in Figure 4.

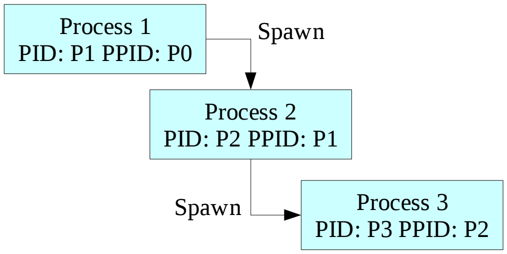

Figure 4. Parent-child relationship

However, while the above method works well for a large class of well behaved jobs, a determined user can easily circumvent it. The parent-child relationship can easily be broken by terminating the parent while leaving the child alive, as shown in Figure 5. Such processes are typically called daemonized processes.

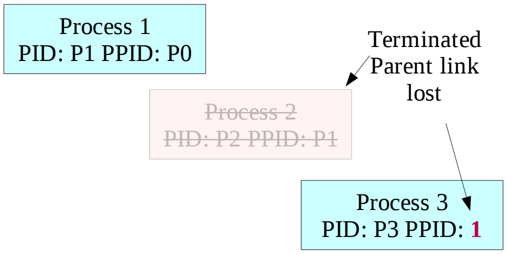

Figure 5. Broken parent-child relationship

This technique can be very effective in evading process relationship tracking. If the parent lives for a very short time, even periodic polling and recording of the complete process tree will miss a large fraction of daemonized processes.

It should be noted that while there is no real need for any non-malicious job to produce daemonized processes, using them can sometimes ease the maintenance of grid job scripts. A classical example is represented by a network transfer companion to a CPU intensive application. Using a nested script to start the companion binary will almost certainly break the parent-child relationship. While this could be avoided by directly invoking the companion, the flexibility of using scripts certainly represents a valid use case that should be supported.

5.1. The ideal process tracker

In order to reliably track the process family tree, an additional unique token can be used. If the parent-child relationship is broken as described above, any process containing the token is known to be part of the user's job. See Figure 6 for an overview.

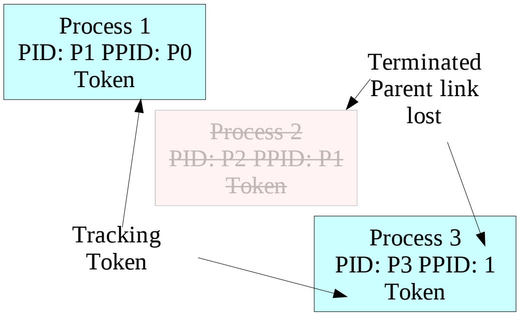

Figure 6. Process tracking using a unique token

In the context of token-based tracking, an ideal tracking system would satisfy all of the requirements listed below:

1) The token must be inherited across forks and execs.
2) The token doesn't effect the semantics of the children.
3) Only a process with root privilege can insert a token.
4) The token cannot be removed by the child processes.
5) Root privilege is not needed to read the tokens from any process on the systems.
6) Multiple tokens are allowed in the system, so that multiple independent groups can be tracked.
7) Multiple tokens are allowed in any process, allowing for nested groups to be tracked.

We are not aware of any system that meets all of these requirements. In the next sections we describe a few widely used methods, followed by the method used by the gLExec monitor.

5.2. Using process groups for process tracking

A popular approach for process tracking is based on process groups. The job's initial process starts a new session, becoming the session leader, and all the children inherit the same process group. The UNIX shell and the PBS batch system[9] are known to heavily rely on this method.

This method violates the rules #3, #4 and #7. Rule #4 is the most troublesome, i.e. the process group is not protected information and can be changed by the child processes at any time. So it cannot be used to obtain reliable monitoring and accounting information.

5.3. Using environment variables for process tracking

The environment associated with a process can also be used for tracking. Any variable stored in the initial process environment is inherited by the child processes. By using well structured unique variable names it is easy to track the process tree. The Condor batch system has been relying on it for years.

This method violates the rules #3 and #4. Again, the violation of rule #4 prevents it from being used for reliable monitoring and accounting. Indeed, any child process can add, change or remove any environment variable.

5.4. Using user identifiers for process tracking

A reliable token that all UNIX processes posses is the User Identifier (UID). Only a tools with root privilege can change the UID of a process. The Condor batch system is known to support this option for dedicated batch system resources.

This method violates the rules #2 and #7. The violation of rule #2 makes it usable in very limited setups only. However, when using dedicated UIDs for process tracking is an acceptable option, this method is very reliable.

5.5. Using secondary group identifiers for process tracking

Secondary Group Identifiers (GIDs) are also a reliable feature of all the UNIX processes. Only a tool with root privilege can add or remove a secondary GID to/from the list. The OSG gLExec monitor and recent Condor versions use this method.

This method violates the rule #2. However, it is relatively easy for a system administrator to set aside a dedicated set of GIDs, making this reliable method deployable in most batch setups.

Figure 7 shows an overview of secondary GIDs for process tracking.

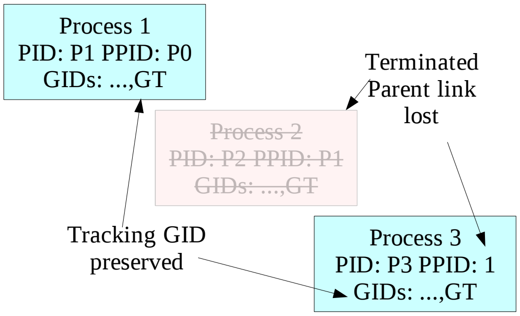

Figure 7. Process tracking using secondary Group Identifiers

6. Integration with the OSG accounting system

GRATIA[10] is the official OSG accounting system. GRATIA has a modular architecture based on probes that allows for a flexible collection of information. GRATIA probes exist for all the major batch systems deployed in OSG as well as for gLExec.

Grid sites can be configured to report to a local or to the OSG-central collector. When a local collector is used, (a subset of) collected data is also forwarded to the OSG-central collector. Detailed message passing description is however beyond the scope of this paper. See Figure 8 for an overview.

To obtain correct accounting information for pilot jobs, one needs to subtract the resources accounted to the final users by the gLExec probe, from the resources accounted to the pilot job by the batch system probe.

In OSG this can be done only in an aggregated mode (as opposed to job by job accounting) today, as there is no direct correlation between the information provided by the batch system probe, running on the Grid gatekeeper, and the gLExec probe, running on one of the worker nodes. We expect that future versions of GRATIA will address this problem.

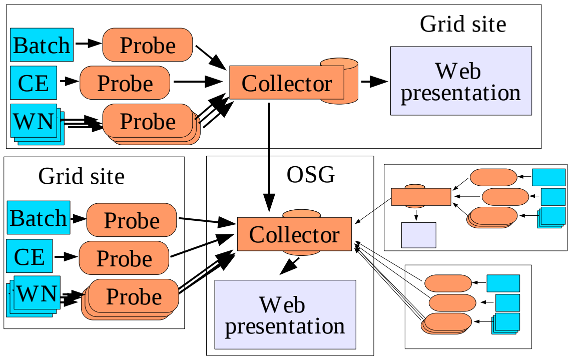

Figure 8. GRATIA accounting system

7. Conclusions

Job accounting and auditing information is important both for the economics of the Grid and for detecting anomalous behavior. However, pilot-based WMSes circumvent the established accounting and auditing mechanisms. To properly detect the jobs handled by pilot-based WMSes, OSG is relying on gLExec, deployed on each and every worker node, to properly track the user processes and account for the user CPU consumption. The accounting records are aggregated in a centralized store by the GRATIA system.

A distinguishing characteristic of the described system is the use of group identifiers for job tracking, allowing for reliable auditing and accounting even in the event of daemonized processes.

8. Acknowledgements

Fermilab is operated by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the United States Department of Energy.

This work was also supported in part by the National Science Foundation.

9. References

[1] A. Tsaregorodtsev, V. Garonne, and I. Stokes-Rees, “DIRAC: A Scalable Lightweight Architecture for High Throughput Computing”, *Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04)*, 2004, pp. 19-25, http://doi.ieeecomputersociety.org/10.1109/GRID.2004.22.

[2] I. Sfiligoi, “ Making science in the Grid world: using glideins to maximize scientific output”, *Nuclear Science Symposium Conference Record, 2007. NSS '07. IEEE* **2**, Honolulu, HI, USA, 2007, pp. 1107-1109, ISBN 978-1-4244-0923-5

[3] “The PanDA Production and Distributed Analysis System”, https://twiki.cern.ch/twiki/bin/view/Atlas/PanDA, Accessed April 2008.

[4] Ruth Pordes, et. al., “The open science grid”, *Journal of Physics: Conference Series* ***78*** , Institute of Physics Publishing, 2007 (15pp), http://www.iop.org/EJ/abstract/1742-6596/78/1/012057/

[5] D. Groep, O. Koeroo, G. Venekamp, “David Groep, Oscar Koeroo, Gerben Venekamp”, *To be published in Journal of Physics: Conference Series (JPCS) CHEP2007*, Preprint: http://www.nikhef.nl/grid/lcaslcmaps/glexec/glexec-chep2007-limited.pdf

[6] “suEXEC Support”, http://httpd.apache.org/docs/2.2/suexec.html, Accessed April 2008.

[7] M. Lorch, et. al., “Authorization and Account Management in the Open Science Grid”, *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing, IEEE Xplore*, Seattle, 2005. http://ieeexplore.ieee.org/servlet/opac?punumber=10354

[8] “The Condor Project”, http://www.cs.wisc.edu/condor/, Accessed April 2008.

[9] “OpenPBS”, http://www.openpbs.com, Accessed April 2008.

[10] ”Gratia”, https://twiki.grid.iu.edu/twiki/bin/view/Accounting/WebHome, Accessed April 2008.