Integrating Slurm Batch System with European and Polish Grid Infrastructures

Dominik Bartkiewicz, Krzysztof Benedyczak, Rafał Kluszczyński,
Marcin Stolarek, and Tomasz Rękawek

Interdisciplinary Centre for Mathematical and Computational Modelling,
University of Warsaw, ul. Pawińskiego 5a, 02-106 Warszawa, Poland
{bart,golbi,klusi,mstol}@icm.edu.pl,newton@mat.umk.pl
http://www.icm.edu.pl

Abstract. ICM, one of the major Polish HPC resource providers, migrated to the Slurm batch system as the first site in the PL-Grid and WLCG grid infrastructures. This article describes the Slurm integration issues and the solutions developed. The integration was focused on several areas where grid middleware interacts with the batch scheduling system, additionally taking into consideration the PL-Grid specifics. In particular, the resource usage accounting required a dedicated Slurm support and the scientific grants enforcement policy needed a new implementation. What is more, in this work we present the reasons of the Slurm adoption and other improvements that were necessary to handle the increasing load of the grid site.

Keywords: Slurm, accounting, gLite, UNICORE.

1 Introduction

The PL-Grid project [4], [8] was started in the year 2009 as a response to many requirements of Polish scientists for more user-friendly access to high-performance computing (HPC) resources. It consists of 5 main Polish supercomputing and networking centers, geographically distributed. The goals of the project were to build the Polish national grid infrastructure, provide users with computing and storage resources accessible in a uniform way. On top of this infrastructure, there are being prepared solutions and services for specific scientific environments, so called "domain grids" (addressed by the current PLGrid Plus project [7]). Unfortunately, distributed resources provisioned in PL-Grid are highly heterogeneous, what makes the integration sometimes not so easy as it was assumed at the beginning.

In the year 2012, ICM HPC system was migrated to the Slurm batch system [10], [14] as the first site in the PL-Grid and WLCG infrastructures. This change was dictated by the fact that Torque [13] became significantly inefficient under heavy load, what we describe in a next section.

The integration tasks included also the implementation of the PL-Grid scientific computational grants system, called POZO. POZO is an infrastructure wide resource access policy, which defines resource limits for scientists, granted by the infrastructure managers. The whole system is based on computing grants assigned to a user or scientific team. Every user has a small private grant issued every half a year to allow him/her to access the infrastructure for one thousand CPU hours. In case when the user needs much more time for simulations, he/she needs to create a scientific team and apply for a proper grant. For this purpose, the Bazaar [9], [12] system is used where HPC administrators negotiate with the user how many resources the user will need. After a successful agreement, no batch system used in PL-Grid should allow to access more resources than agreed.

POZO policy access describes how the system should behave when user resources are exhausted. All batch systems used in PL-Grid have to work according to the policy. In particular, Slurm used at ICM. The way of POZO implementation is described in the article.

The PL-Grid Infrastructure has complex requirements with regard to the accounting of the resource consumption by the grid jobs. It is required for billing both the individual users and also groups of users. Therefore, the accounting data needs to be collected and centrally stored for the internal management of the PL-Grid Infrastructure itself. This data must include records of both grid jobs (regardless of the middleware used) and jobs submitted locally, directly to the computing machines. The information about the middleware being used (if any) is also required for the purposes of internal reporting.

Besides the national accounting system, PL-Grid as a member of the European Grid Infrastructure (EGI) is additionally obligated to publish the accounting data to the European-wide database. The data exported to EGI typically should be pre-processed, to provide coarse-grained usage summaries only, and includes exclusively the grid job related records. Therefore, the accounting data required by EGI is different both in terms of the content, format and source jobs from the data being used in the National Grid Infrastructure (NGI).

As the vast majority of accounting data is collected from a resource management system, the newly introduced Slurm system needed to be integrated with the whole, complex accounting infrastructure. What is more, it turned out that the UNICORE accounting system [2] had, similarly as Torque, performance problems when tackling the constantly increasing stream of jobs. In this article we discuss the details of the observed problems and the developed solutions.

2 State of the Art

The Torque batch subsystem, used until recently at ICM, is properly supported by the grid infrastructure and the typical helper tools. This is true for the gLite job submission interface, the gLite APEL accounting system [5], UNICORE job submission and, eventually, Torque is properly supported by the UNICORE accounting system [2].

Unfortunately, in the recent years we observed that version 3.x of the Torque batch system (the latest one at that moment) was unstable under heavy load. It even came to solutions where a server daemon was checked if it responded and restarted in case if not. This was the reason why we have tested the Slurm batch system, which proved much more stable and efficient, especially with a load of thousands of jobs.

One of the grid systems supported by the PL-Grid Infrastructure is gLite with the Cream computing element (CE). CREAM accepts job submission requests described with Job Description Language (JDL) [6]. The CREAM support for the Torque batch system is well tested and works fine including the accounting subsystem. Unfortunately, JDL does not support any attribute, which can be used to express the PL-Grid grant. For authentication purposes, CREAM uses an LCAS/LCMAPS security stack. LCMAPS is a pluggable framework, implemented as a library that can be used by applications to map incoming grid identities to local POSIX identities, taking into account the local site policy. Therefore, LCMAPS plugin has been developed at ICM to allow efficient mapping grid-based users to their own POSIX LDAP accounts. Fig. 1 presents a decision rule used for the mapping.

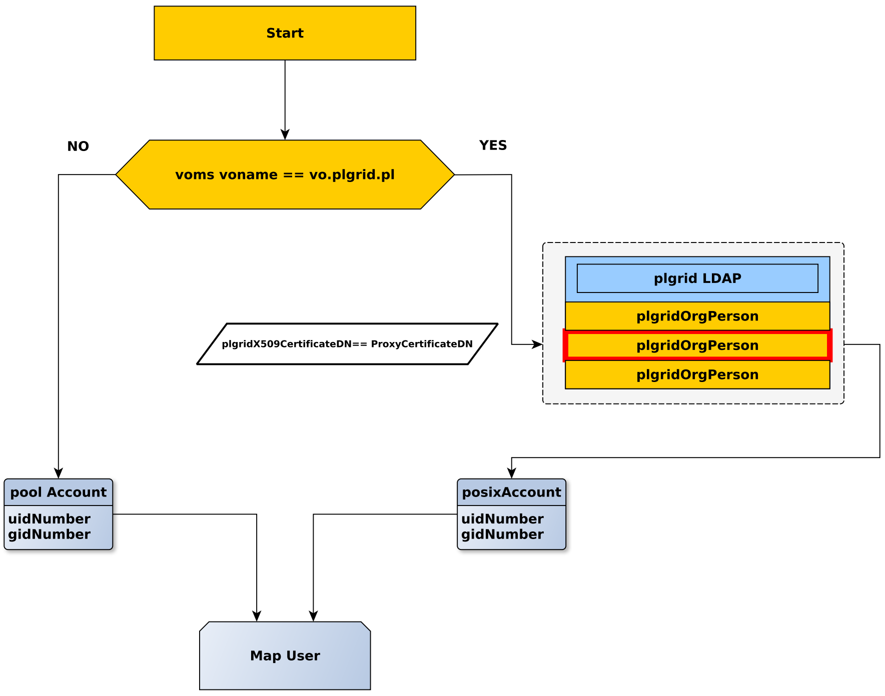

Fig. 1. The diagram of PL-Grid LCMAPS plugin. It shows decision rule based on the user's VO organization.

Such a method reacts in real time to changes in default grants or adding new users to the system. At the same time, it eliminates frequent generation of gridmapfile, which includes all users data. Support for Slurm in the gLite accounting system (called APEL) is officially available since its latest release, however it is immature and significantly buggy. Tests have shown that over 80% of job records were not handled correctly by the APEL Slurm module.

UNICORE, one of the three base grid solutions deployed in the PL-Grid Infrastructure [3], natively supports custom resource requirements. These requirements, among others, can be used to express the information about a requested PL-Grid grant in each UNICORE job. As all resource requirements are subject to brokering, a grant-aware site selection can be easily implemented in UNICORE. This, however, was not yet done as typically the grants are supported on all sites and in the rare cases when this is not true, the automatic job rescheduling solves the problem. Therefore, the integration with the PL-Grid POZO was not problematic from the UNICORE perspective.

UNICORE's accounting system developed at ICM was released back in 2010 and subsequently it has been deployed in the PL-Grid Infrastructure. Since this date, the system has processed several million of jobs using the Torque batch system, which was supported by the system since its beginning (next to Sun/Oracle Grid Engine). The situation with the UNICORE and Slurm was worse, as the Slurm batch system was not supported in the accounting system. To understand the required functionality, the details of the accounting deployment in PL-Grid need to be presented.

The PL-Grid Infrastructure features a complex accounting system or – more precisely – a set of cooperating accounting systems. The principal solution used for national and at the same time internal project accounting is called MWZ. Originally, MWZ was designed to collect accounting data from the gLite middleware. As PL-Grid exposes also QCG and UNICORE interfaces, their accounting solutions were deployed too, with additional plugins exporting the data to the central MWZ database. The export of accounting records to the EGI infrastructure should be done separately by each middleware installed. So far, it was only set up for the gLite system using its typical solution – APEL, the same software, which is used by EGI to receive and store the records in the central database.

The retrieval of the accounting data is as complicated as its distribution. The accounting data must be collected from two sources and then merged: from the resource management system (as Torque or PBS Pro) and from each middleware to enrich the low level data by grid related information.

To summarize this complex accounting landscape, the accounting deployment looks as follows from the ICM site perspective:

– APEL software is installed on the site and reports the accounting data of gLite jobs directly to the EGI central database.
– The UNICORE accounting system is installed and (independently of APEL) collects accounting data about all jobs submitted to the site from the batch subsystem (regardless of the middleware) and from the UNICORE middleware (in case of UNICORE jobs).
– Finally, the accounting data is exported to the central MWZ server and (what is planned) should be exported to the EGI central APEL service in case of UNICORE jobs.

In case of the remaining sites, typically the MWZ agent is installed to collect the data from the resource management system, and the UNICORE accounting module is responsible only for retrieving the grid part. The UNICORE grid part is sent first to the UNICORE accounting system at ICM and from that point it is forwarded (together with other records) to all configured consumers as MWZ or APEL. This scenario is presented in Fig. 2.

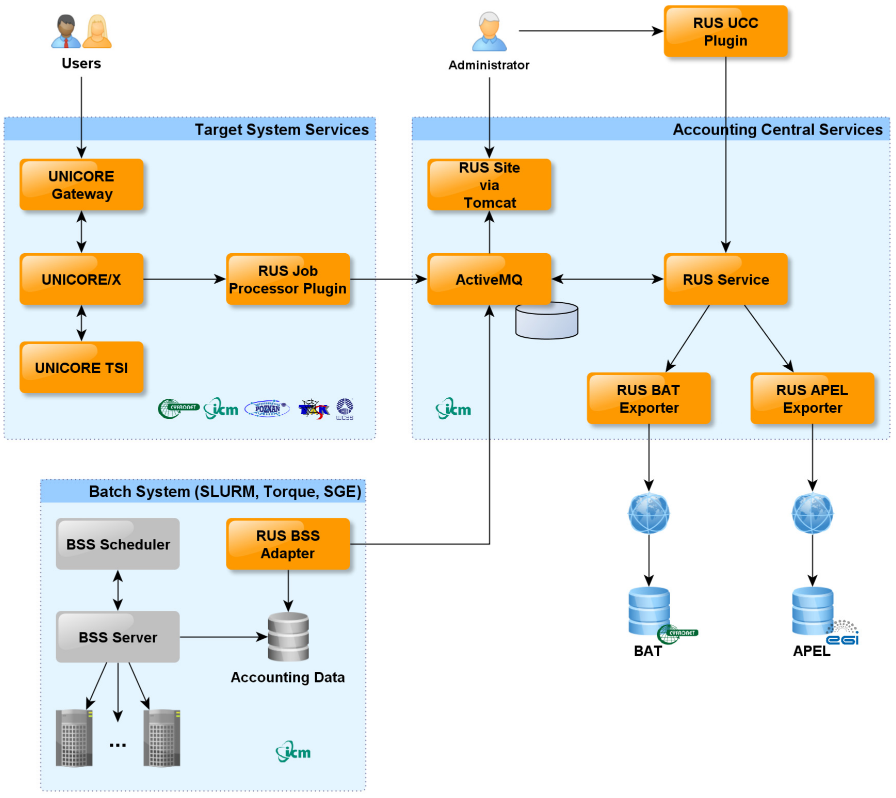

Fig. 2. UNICORE accounting deployment at the PL-Grid Infrastructure

The UNICORE accounting was tested with a typical HPC load and it handled thousands of jobs without any noticeable performance problems. However, as soon as a large amount of small high-throughput computing (HTC) jobs started to be processed, the database size reaches a much larger size. Around 1,000,000 records, the system started to consume large amount of CPU time during regular operation. It has become obvious that sooner or later the time needed to process a single job will be longer than an average delay between subsequent jobs collected by the system. Such a situation would nearly instantly lead to a crash of the whole system.

In the next section, a description of the Slurm integration module as well as the general performance and database model changes are presented.

3 Description of the Solution

The article describes the solutions for the Slurm integration with the gLite and UNICORE middleware as well as the implementation of the PL-Grid grants system in the gLite middleware.

The Slurm scripts from the gLite APEL were updated and the article discusses the issues and their solutions. What is more, the support for limiting the access to resources based on the PL-Grid grants specification (called POZO) with an ability to handle users' default grants has been implemented. The article describes the solution allowing for the enforcement of the PL-Grid POZO system with a new LCMAPS module. In this module, we use the data picked directly from the PL-Grid LDAP database, so the changes take place immediately, once the data is updated in the central PL-Grid database. However, it is important to emphasize that the main source of information about users' grants in POZO system is Bazaar [9], which provides data in XML format, available only for HPC administrators through a secure protocol. The Bazaar part of data contains more detailed information agreed between the site and the user, containing among others:

- maximal total walltime,
- maximal walltime for a single job,
- maximal parallelism of a single job,
- minimal memory value.

During the phase of implementation of POZO for the Slurm batch system three different solutions were tested, all of them having their advantages and disadvantages. All of them are using Slurm built-in associations. An association is a 4-tuple consisting of a cluster name, a bank account, a user and optionally a partition. If set correctly, only the users with valid associations will be enabled to run jobs. The bank account contains information about user's limits, which also are enforced during job execution. This is a convenient way to implement any computing policy, so we decided to use it during POZO implementation.

The first attempt was a simple script working periodically as a cron job. This solution was gathering data from PL-Grid LDAP and Bazaar and adding all users and account information into Slurm database using normal user space Slurm utilities. In this concept, all data was being processed by slurmdbd daemon. A schematic workload of this scenario is presented in Fig. 3.

Unfortunately, the PL-Grid Infrastructure supports more than seven hundreds users, all of them having their private grants. Running such a script in this situation was causing Slurm to be not responsive to any users' grant changes for more than two hours. Obviously, this cannot be accepted and another attempt to implement Slurm support was needed.

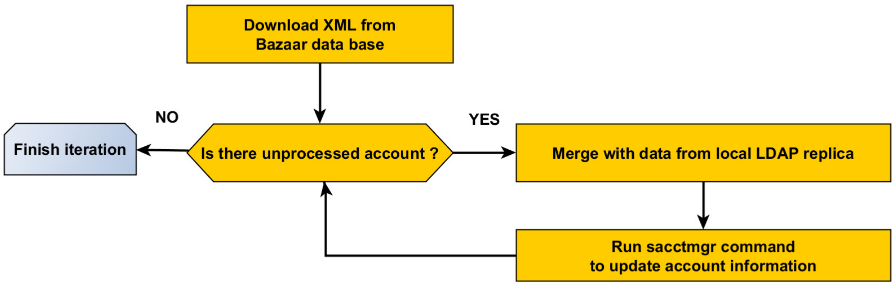
Download XML from Bazaar data base
NO
YES
Finish iteration
Is there unprocessed account ?
Merge with data from local LDAP replica
Run sacctmgr command to update account information

Fig. 3. The diagram of the first prototype of POZO implementation for Slurm

The first idea was to construct a buffer and run Slurm utilities only to update the users' grants data that have changed between a current and previous iteration. The implemented buffer database was almost a copy of Slurm accounting tables. This solution used SQLite [11], because its efficiency was not that important. Unluckily, another situation specific for the PL-Grid Infrastructure prevented this implementation from going into the production phase. The private grants, which are given by default to all PL-Grid users, are changing every half of year. This change is applied by the Bazaar system in several parts. Even though it still could cause efficiency problems every six months. This time it could cause Slurm to be immune to any grants changes for even three days. This again was not acceptable. A schematic design for this workflow is presented in Fig. 4.

While the previous implementations were tested, the Slurm version at ICM was upgraded from 2.4.3 to 2.6.1, allowing us to redesign the solution. The new version of Slurm supported reading the account information from backups with flushing the current state of the accounting database (in previous versions this operation was not able to delete users/accounts). In fact, this operation is not a simple flush. The dump file of the database is being compared with the current state of the database and, based on the results, a binary version of the new database state is prepared. Then, the whole new database is being imported through slurmdbd. This operation takes less than a minute, so with setting up proper timeouts it is possible to perform this action every day, which is a standard accepted by the PL-Grid Infrastructure (the existing solutions for Moab/Maui and PBS Pro work with the same interval constraint). A schematic workflow for this solution is presented in Fig. 5.

Batch system integration has also been done in the designated component of the UNICORE accounting, called BSS-Adapter. It orchestrates various batch system dependent modules, which have to implement a common internal SPI. The job records produced by a selected module are sent to the rest of the UNICORE accounting system using the ActiveMQ [1] message bus.

The Slurm support turned out to be the most difficult submodule of the BSS-Adapter. First of all, a difficult decision needed to be made on from what part

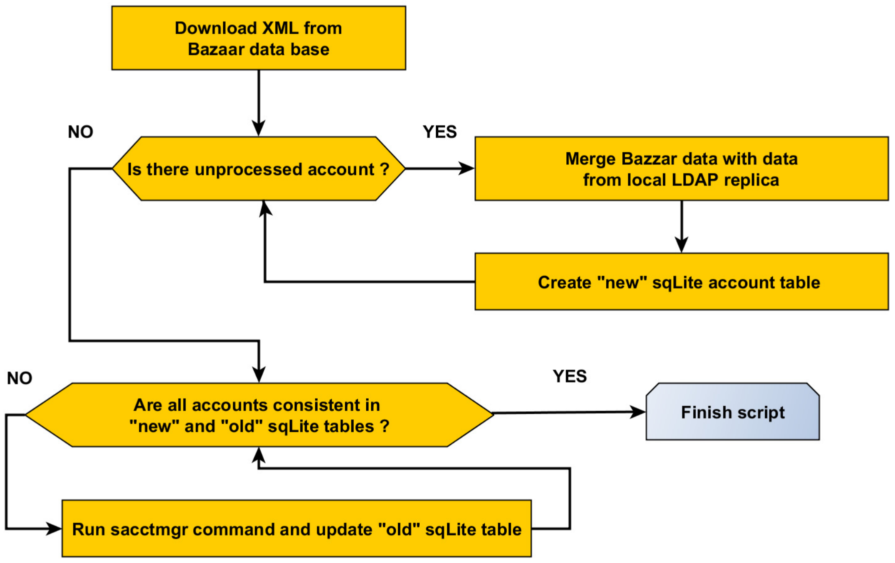
Download XML from Bazaar data base
NO
YES
Is there unprocessed account ?
Merge Bazzar data with data from local LDAP replica
Create "new" sqLite account table
NO
YES
Are all accounts consistent in "new" and "old" sqLite tables ?
Finish script
Run sacctmgr command and update "old" sqLite table

Fig. 4. The diagram of the second solution of integration of the POZO policy with the Slurm batch system

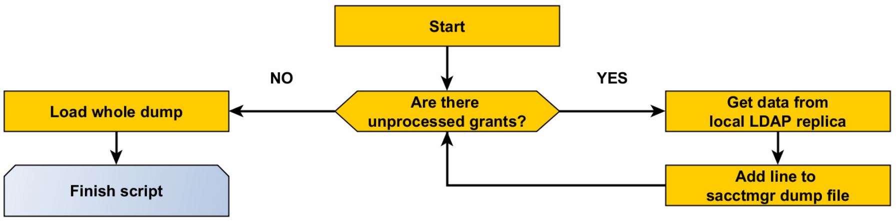
Start
NO
YES
Load whole dump
Are there unprocessed grants?
Get data from local LDAP replica
Finish script
Add line to sacctmgr dump file

Fig. 5. The diagram of the final implementation of POZO used at the ICM. To be fully responsive, the solution needs at least 2.6.1 version of Slurm.

of Slurm the accounting data should be retrieved. As Slurm supports different storage backends for accounting data, it is possible to extract information directly from any of them (MySQL database and text files are the most common choices). Such an approach is easier from the development point of view, however in order to make the RUS BSS-Adapter independent from a SLURM backend, we developed a solution accessing the SLURM accounting data via its standard reporting *sacct* command.

The *sacct* command guarantees the same output format, regardless of the selected accounting storage backend. It also allows for changing its output format, with an ability to choose a format, which is suited to the machine processing.

Theoretically, this looks as a perfect approach, however practice showed a lot of obstacles:

- The output of the *sacct* command is not escaped and in several cases the field content can itself contain a field separator character (which is "|"). This problem complicates the *sacct* output parsing a lot.
- Officially, the output of *sacct* can be controlled with time range parameters, unfortunately some of them don't work correctly. Therefore, the relaying application has to support the situation when the same job record is presented multiple times.
- It was observed that Slurm updates the accounting information of the jobs, which were previously marked as finished. For instance, a job's CPU time is sometimes slightly adjusted.
- The job information is provided in different ways depending on internal Slurm handling – sometimes some of the job's information is split into the so called *job steps*, what is a Slurm artifact.

To handle all the above obstacles, a quite complex algorithm was developed. It is presented in Fig. 6.

The algorithm is invoked in an infinite loop. The data is obtained from the *sacct* command and the output is processed job by job. The parsing is using a special trick to detect inclusion of a field separator character used as an ordinary character: *sacct* is invoked to output all the fields, which can carry such characters twice. By finding the repeated text sequences, the actual content of those fields can be established, without relying on the field separator. Additionally, all job steps are processed and if the data is available in them, it is added to the job record. This is typical for the memory usage.

The next step is a record post-processing. The system is storing in memory a cache of key information of the already sent jobs. The unfinished jobs without a status change are ignored and discarded. Note that this system is not fully guaranteeing that a particular job record is absolutely never sent twice (for instance, this can happen if a BSS-Adapter is restarted and the sent records cache is cleared). However, this is not a problem for the accounting system as the central service ignores the unchanged job data. Therefore, this step is only reducing the network and processing load.

The second post-processing step is a workaround for the problem of Slurm changing the accounting information of a job, which is already in a terminal state and possibly could be already sent to the central accounting system. Therefore, the Slurm module ceases to send job records in a terminal state, which were finished only recently. The security time threshold is set to 2 minutes and tests confirm that after this time the job information becomes stable. Such jobs are going to be sent during the subsequent iteration of the Slurm module.

Besides the Slurm integration, the accounting system was redesigned to handle greater load and provide an unattended operation with a constantly growing

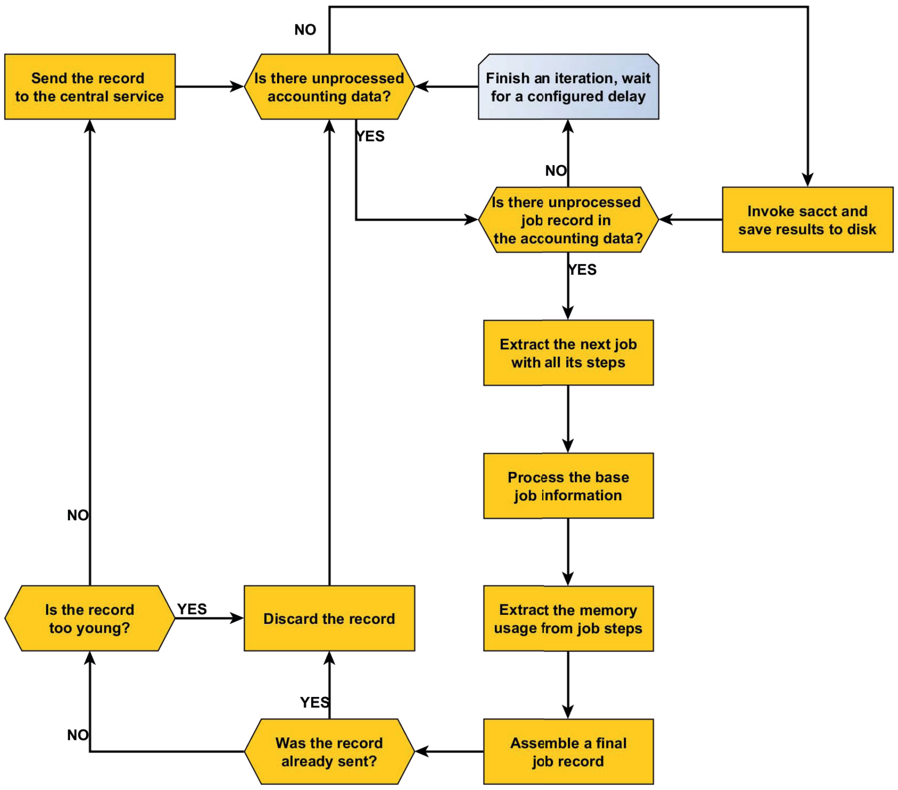
NO
Send the record to the central service
Is there unprocessed accounting data?
Finish an iteration, wait for a configured delay
YES
NO
Is there unprocessed job record in the accounting data?
Invoke sacct and save results to disk
YES
Extract the next job with all its steps
Process the base job information
NO
Is the record too young?
YES
Discard the record
Extract the memory usage from job steps
YES
NO
Was the record already sent?
Assemble a final job record

Fig. 6. Processing of the Slurm accounting data, using the *sacct* program as a source. The highlighted element is a good starting point to start following this infinite loop.

amount of processed records. The storage layer was redesigned and the new version is using a quite different persistence model. First of all, the daily usage summaries (kept for each user, site, queue and job status) become a fundamental element that is used for reporting and user-driven accounting data presentation. The measurements showed that such an approach provides a data amount reduction by approximately 20 times. The individual job records are still stored in the database, but only for a relatively short period of time (configurable, by default half a year) and after this period are automatically moved to a separate archive. This allows for keeping the advanced functionality of the system, which relies on the fine grained information stored in individual job records, yet keeping the amount of processed records at a reasonable level.

What is more, the same mechanism was implemented for the summary records, but with a much longer time period, after which the summaries are moved to an archive. Those two solutions together greatly improved the response time in case of accounting queries, reduced the everyday processing time and, finally, provide a promise that the system can work for many years to come.

4 Conclusions and Future Work

As a result, we have managed to deploy a new version of services with a full Slurm support. All middleware available in PL-Grid is able to execute jobs using Slurm and, what is more important, the accounting of the batch system and grid data is properly gathered and exported to external services in PL-Grid and EGI. The support for the PL-Grid grants system is available not only in UNICORE, but also in gLite. Additionally, users static mapping implemented by LCMAPS plugin was also successfully used for GridFTP connections and GSI-SSH configuration in other PL-Grid sites.

The integration was problematic at the beginning and still requires some work to make it more reliable. Nevertheless, we have managed to make it fully operational, which is a notable success and an example for others to pursue.

Further work concerning integration of Slurm with the new version of POZO may focus on writing a separate slurmdbd accounting backend, which will gather the usage of MySQL and LDAP databases. This work may be of special interest for all Slurm users, because information about users and groups is customarily kept in an LDAP database, while information about their jobs accounting should be saved in a relational database. Such a solution will have to deal with a lot of issues coming from possible incoherencies between separate databases, but will give the administrators a very convenient tool.

References

1. ActiveMQ message bus web site, `http://activemq.apache.org`
2. Bała, P., Benedyczak, K., Kluszczyński, R., Marczak, G.: Advancements in UNICORE Accounting. In: UNICORE Summit 2013 Proceedings, IAS Series 21, iii, Forschungszentrum Jülich GmbH Zentralbibliothek. Verlag Jülich (2013)
3. Benedyczak, K., Stolarek, M., Rowicki, R., Kluszczyński, R., Borcz, M., Marczak, G., Filocha, M., Bała, P.: Seamless Access to the PL-Grid e-Infrastructure Using UNICORE Middleware. In: Bubak, M., Szepieniec, T., Wiatr, K. (eds.) PL-Grid 2011. LNCS, vol. 7136, pp. 56–72. Springer, Heidelberg (2012)
4. Kitowski, J., Turała, M., Wiatr, K., Dutka, Ł.: PL-Grid: Foundations and Perspectives of National Computing Infrastructure. In: Bubak, M., Szepieniec, T., Wiatr, K. (eds.) PL-Grid 2011. LNCS, vol. 7136, pp. 1–14. Springer, Heidelberg (2012)
5. Byrom, R., Cordenonsi, R., Cornwall, L., Craig, M., Djaoui, A., Duncan, A., Fisher, S., Gordon, J., Hicks, S., Kant, D., Leake, J., Middleton, R., Thorpe, M., Walk, J., Wilson, A.: APEL: An implementation of Grid accounting using R-GMA. In: UK e-Science All Hands Conference (2005)
6. Pacini, F., Kunzt, P.: Job description language attributes specification. EGEE project (2006)
7. PLGrid Plus project web site, `http://www.plgrid.pl/en/projects/plus`
8. PL-Grid project web site, `http://projekt.plgrid.pl/en`
9. PL-Grid Resources Bazaar portal, `https://bazaar.plgrid.pl`
10. Slurm Workload Manager web site, `http://slurm.schedmd.com`
11. SQLite database web site, `http://www.sqlite.org`

12. Szepieniec, T., Tomanek, M., Radecki, M., Szopa, M., Bubak, M.: Implementation of Service Level Management in PL-Grid Infrastructure. In: Bubak, M., Szepieniec, T., Wiatr, K. (eds.) PL-Grid 2011. LNCS, vol. 7136, pp. 171–181. Springer, Heidelberg (2012)
13. Torque Resource Manager web site, `http://www.adaptivecomputing.com/products/open-source/torque/`
14. Yoo, A.B., Jette, M.A., Grondona, M.: SLURM: Simple Linux Utility for Resource Management. In: Feitelson, D.G., Rudolph, L., Schwiegelshohn, U. (eds.) JSSPP 2003. LNCS, vol. 2862, pp. 44–60. Springer, Heidelberg (2003)