EXPERIMENT SOFTWARE INSTALLATION EXPERIENCE IN LGC-2

R. Santinelli, F. Donno, S. Campana, A. Delgado Peris, P. Méndez Lorenzo, A. Sciabà
CERN, Geneva, Switzerland

Abstract

The management of application and experiment software represents a very common issue in emerging Grid-aware computing infrastructures. The current solution adopted by the LHC Computing Grid (LCG) infrastructure for HEP experiments allows an Experiment Software Manager (ESM) to install the software on a VO-specific file system shared among all worker nodes (WN) of a farm. With this work we present a more flexible service based on P2P technology that has been designed to tackle the limitation of the current system. Here we illustrate the design, deployment and preliminary results obtained.

1. INTRODUCTION

In the LHC Computing Grid, while the middleware is installed by system administrators at a site level via customized tools [1,2] that serve also for the centralized management of the entire computing facility, Gigabytes of Virtual Organization (VO) specific software or frequently changing user applications need to be pre-installed, configured and validated before a user job is executed at a site. Following the requirements imposed by the experiments, in LCG Experiment Software Managers (ESM) are designated people with privileges for completely managing software for a specific VO on a per site basis. An ESM can also publish univocally identified software tags in the LCG Information System (IS) to announce the availability of a specific software version. Users of a VO can then select, via the published tag, sites to run their jobs. The solution adopted by LCG has mainly served its purpose but it has several drawbacks.

In this article we report on the work done to collect a list of requirements for the realization of an effective software installation service for Grid and propose the Tank&Spark Grid service as a possible solution to the problem.

In Section 2 we describe the problem of Grid application and experiment software installation. In Sections 3 and 4 we list the requirements from experiments' and site administrators' points of views, respectively. In Section 5 we describe the current solution adopted by LCG and used mainly by ATLAS and CMS, and report on the feedback received. In Section 6 we illustrate our proposed solution currently under test and give some implementation details. Preliminary results using the new software installation service, a summary on related work, future work, and the conclusions are presented at the end of this article.

2. THE PROBLEM

The problem of application and experiment software installation in LCG and more in general in Grid-aware computing facilities is not trivial. From an **end-user** point of view the main requirement is the following: it has to be guaranteed that experiment specific software, needed for running a job, is available and validated at any site the job can run. To achieve this, the ESM requires adequate tools that allow for triggering software installation on Grid, managing VO disk space and software versions, planning for software upgrade and removal, publishing site and software status related information, etc. From a **site administrator**'s point of view what described above becomes a source of concerns in terms of site security, local policies to be respected, maintenance scheduling and related problems, etc.

Many issues need technical answers and solutions, such as:

- Establishing a mechanism that allows for scheduling a software installation process when appropriate.
- Ensuring adequate disk and space management.
- Failures and conflicts resilience.
- Resolving software dependencies issues.
- Handling concurrent installations.
- Satisfying pre-requisites before the experiment software installation is triggered.

The above is a non-exhaustive list of issues that has to be addressed when designing an application software manager service for Grid.

3. THE EXPERIMENT'S VIEW

During the start-up of data challenges executed on the LCG infrastructure we collected requirements in terms of software installation from the four LHC experiments and from the site administrators running the LCG facility. Here, we list the main features that a software installation service should provide, from a user perspective.

Table 1: Disk space needed per experiment

Experiment	**Disk Space**
Alice	1-2 GB
ATLAS	6 GB
CMS	2 GB
LHCb	1-2 GB

- Each LHC experiment requires frequent update of its software releases, about three times per month. Software should be installed freely and whenever necessary removing old unused versions if applicable.
- In Table 1 we report the space requested by each LHC experiment in order to allow for 2 releases of the software to coexist.
- All software for the experiment should be installed relative to a path. The path is accessible through an environmental variable.
- No root access should be required to install experiment software and actually experiments required this to be always the case.
- Only a subset of the experiment's users can write into the experiment area. This is achieved through the ESM special accounts. An ESM should be able to add/remove software at any time without communication with the site managers.
- The software has to be accessible on the WN through POSIX calls.
- The ESM should be able to install software on a per site base as well as launching a request to the entire Grid supporting the specific VO.
- The ESM should be able to verify the installation in separate steps. Different kind of validation procedures can be run by the ESM at different moments.
- The ESM must have the possibility to publish in the Grid Information System for a site special software Tags to advertise all installed and validated versions of the software in order to direct jobs to that site.
- It is the responsibility of the ESM to prepare a given software distribution for a given release and manage dependencies.
- Experiment software can be packaged, as the experiment requires: tarballs, RPMs, DAR files, Pacman [3], etc. Installation and validation scripts should be provided by the experiments. Therefore, dependencies should be expressed in a way that those scripts can process them.
- The user environment should be setup by a script placed in a given location that the user job sources as a very first step.

4. THE SITE ADMINISTRATOR'S VIEW

The main concerns from the site manager's point of view are summarized in the following list.

- For security and maintenance reasons, no daemons running on WNs are allowed. Neither user applications nor software installation Grid services should have control over WNs.
- Every individual access to a site must be traceable. For this reason, no shared accounts are allowed.
- The information published by a site must not be corruptible.
- Service actions, such as restart, flushing, etc. must not be triggerable externally unless policies can be applied. In fact, this could lead to denial-of-service (DoS) attacks when the service is continuously restarted. Such a DoS attack not only affects the VO with the compromised ESM account, but also will bring down the entire site.
- Access to any tool/service must be strongly authenticated, and the restrictions and policies should be applied on the server-end and not on the client-end.
- Possibly inbound/outbound connectivity requirements from WNs should be avoided.
- It should be possible to control and apply site policies to the software installation mechanism.
- One should not assume a shared file systems among WNs to serve experiment software. Such a requirement in fact poses serious performance, reliability and scalability problems for large installations.

5. THE CURRENT SOLUTION

The current solution [4] proposed by LCG relies on the figure of the ESM. The X.509 certificate subject of such a person is mapped locally on a Grid farm to a special Grid account with special write privileges in certain experiment areas. Each experiment selects one or more ESMs. The ESM is the person in charge to update the software of the experiment at each site.

The experiment software is first packaged into a software specific bundle and moved via the Grid Data Management system to one of the Storage Elements (SE) belonging to the site where the software will be installed. In this way the software can be installed on that farm using a local cache and without requiring inbound/outbound connectivity from the WNs.

Then the ESM directs an installation job to the site. Depending on the value of the variable VO_<EXP>_SW_DIR the job installs the software at the indicated location. The content of this variable is essential to follow the behaviour of the tool. If the value is a ".", the software is installed and validated in the job working area and then removed when the job is finished. If the validation step has passed with success, the ESM can publish in the Information System attribute *GlueHostApplicationSoftwareRunTimeEnvironment* a VO software specific tag that certifies the site for that specific version of the VO software. Subsequent jobs ending up on the same WN for execution have to perform first the software installation step in their working area and then execute the real job.

If the value of the VO_<EXP>_SW_DIR environmental variable is not ".", the software is installed in a permanent area (this can be shared among the WNs or local to a specific WN). The ESM job has to first check if the version of the software to be installed is already present. Only if that version is not there the job proceeds with the installation since the ESM is guaranteed to have write privileges in that area. In this case the ESM can run validation scripts in a second step, and only if the validation process is successful, the ESM can publish the relative software tag in the Computing Element (CE) IS using a tool provided by LCG.

The solution adopted by LCG has mainly served its purpose. In particular, it provides a framework within which experiments are free to use their own proprietary distribution tools (Pacman [3] for ATLAS, tarballs for Alice, DAR for CMS, a CVS repository accessible via http/wget for LHCb). However, the current solution has several drawbacks, as reported by the LHC experiments in [5]:

- The lack of "roles" severely constrains the abilities of software managers. An ESM should be able to dynamically switch his/her role and become a normal user able to submit normal user requests to the Grid.
- Many jobs failures are often due to loss of visibility of the NFS file system either during software installation or during run. Avoiding the use of NFS on large installation can cure this problem. However with the current system it is impossible to trigger on demand installation on a whole farm of WNs.
- The ESM job has to compete with normal user jobs without any special priority.
- There is no automatic mechanism to trigger a software installation on the whole Grid, i.e. on all sites supporting a specific VO.

6. OUR SOLUTION: TANK&SPARK

With release 2_2_1 of LCG-2 we introduce a service called **Tank&Spark** that satisfies the requirements previously listed. The toolkit is fully integrated with the current solution. However it can work as well in a LCG unconstrained framework. The toolkit provides as well for many other interesting features, and it is fully compliant with the policies imposed by site administrators.

Triggering automatic distribution to the Grid can be achieved via a Grid job or directly contacting the installation service at a site from a User Interface (UI). In this latter case the ESM will not compete with normal user jobs but it can immediately schedule a software installation request.

The architecture of such a service foresees a multi-threaded server *(Tan*k) running on a dedicated machine (a Computing Element for instance), a client application *(Spar*k) running on each WN of a farm and a r-sync server running on a disk-server (a Storage Element) acting as central repository of the experiment software.
Tank is a daemon listening on a dedicated port for incoming connections. It can currently accept GSI-authenticated and insecure connections but other security protocols can be easily integrated. The service can therefore be interfaced to VOMS and use the user credentials interpreting user roles. Tank uses a MySQL database to store internal status information.

The server component represents the central intelligence of the system managing the various releases of the experiment software that need to be installed/removed.

The server obeys to the local policies set by the site administrator on whether the installation/removal process can take place or not. On a WN (or UI) when Spark is invoked (via an ESM job), it installs the software locally and then it contacts Tank. After authentication Spark registers the new software tag in Tank's DB.

On the other WNs, the client program is called by a cron job running every 5 minutes. It retrieves the list of tags relative to software releases installed since the last update on that machine. In case of new updates, the client synchronizes the local software area with the central repository.

Such a schema allows for the management of concurrent installations for the same VO. If an installation or upgrade process is going on, the systems stops another installation process from the same VO because of a temporary lock imposed on that VO that lasts until the process ends. Tank also controls the installation/removal process for a specific WN by setting appropriates field on the DB. In this way the installation on WNs sharing a file system takes place only once allowing for the management of farms with or without a shared file system or in a mixed configuration.

7. THE IMPLEMENTATION

Tank&Spark has been entirely written in C++. The server exposes its methods through the SOAP protocol using gSOAP v2.3.

The Tank server uses the CERN implementation of the GSI plug-ins for gSOAP. Via a local grid-mapfile or other mechanisms, such as the EDG LCAS server, only authorized users (or with the right role) are allowed to perform installation tasks.

However, a module to interface to the generic high-level security interface presented in this CHEP conference is already foreseen. In this way the server can dynamically support multiple authentication mechanisms.

The MySQL database on the server side keeps track of information regarding the status of the nodes under the control of the local Tank server, the last update time, the type of the installation on that node (shared or local), etc. Together with this information, a table storing the tags relative to the software to be installed and their status is kept. The "Monitor" table is used to keep control over users performing installation on a given machine. The same table is also used to enforce local policies.

The MySQL back-end database can be easily replaced with other more reliable and robust databases such as Oracle.

The service uses rsync to synchronize software directories. The rsync mechanism is a plug-in and can be replaced with other tools.

The toolkit is maintained using the GNU Autotools and distributed via RPMs.

Installation, configuration and maintenance are quite easy tasks. A Tank server can serve multiple VOs, while at the moment, on each WN a crontab entry is needed per VO supported running under one of the ESM local accounts.

Both server and client are resilient to failures. If the server goes down while an installation request is on

going, the user is notified and the installation can be triggered later on. For this the retry mechanism of the job submission on the LCG-2 infrastructure can be used.

Nodes contacting the server will just retry at a later time.

If a WN goes down and looses the software disk, the server will take care of triggering the installation of all missing VO specific software versions.

8. EXPERIMENTAL RESULTS

We performed some preliminary functionality and performances tests using the LCG Grid farm in Pisa. The configuration of the farm is Pisa is reported in Table 2.

Table 2: Pisa farm configuration

Node Role	Configuration
CE	PIII 1GHz, 512MB
SE	PIV 1.5GHz, 256MB
9 WN	Dual PIII 1GHz, 512MB/ Dual AMD 1.6GHz, 1GB/ Dual Xeon 2.4GHz, 512MB

We installed Tank on the CE and used as rsync server the one available on the SE.

On each WN a cron job, running every 5 minutes under an ESM local account, serves two VOs. We simulated a mixed configured farm with some WNs sharing a file system for experiment software.

We simulated a true installation job, either by submitting it through a Grid job or by running the tool as a standalone application from a WN. The job installs the software tagged as *CMSIM-145.1.2-0.* In Figure 1 we show a few screen shots of the process.

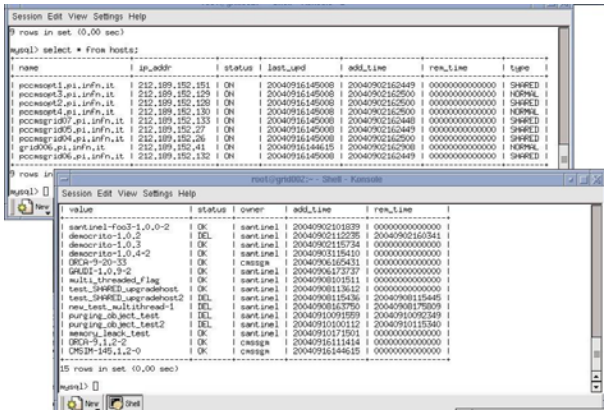

Figure 1: The Tank DB tables

We monitored the memory usage (about 3 MB) and the CPU load of the server program lcg-utank receiving requests for each VO every 5 minutes. In our setup (2 VO and 9 WNs) the server receives 18 connections and the CPU load is negligible (0.1%).

9. RELATED WORK

The Pacman software [3] has been developed by the ATLAS collaboration for software distribution, installation and configuration. Even though the package is very effective and allows for dependencies management, it cannot be considered an alternative to Tank&Spark. In fact it cannot trigger installation on a set of WNs. However Pacman could complement our solution replacing for instance the rsync server.

A very recent work on application software installation is the one being developed in EGEE: the gLite PackMan [6]. As presented this tool can be an alternative to Tank&Spark. However PackMan does not just offer a framework for ESMs since it forces the packager to give a very detailed description of the software via specific metadata files. In addition, PackMan does not tackle the problem of automatic installation on a farm of WNs.

10. CONCLUSIONS

In this work we have presented our experience with application and experiment software installation tools, the list of requirements for an adequate tool, and our proposed solution Tank&Spark. Preliminary results show that the service proposed seems to be quite stable and highly performing. Further enhancements include a more reliable way to notify ESMs of the status of the installation process, full support for pool ESM accounts, a more efficient way to handle requests coming from a UI, etc.

ACKNOWLEDGEMENTS

This work has been funded by Istituto Nazionale di Fisica Nucleare, Rome - Italy and Ministerio de Educación y Ciencia, Madrid - Spain.

REFERENCES

[1] Quattor: http://cern.ch/quattor/

[2] LCFGng: http://datagrid.in2p3.fr/distribution/datagrid/wp4/edg-lcfg/documentation/

[3] Pacman: http://physics.bu.edu/~youssef/pacman/index.html

[4] The LCG-2 Software Installation tool: https://edms.cern.ch/file/412781/SoftwareInstallation.ps

[5] Preliminary observations on LCG-2 based on the 2004 LHC data Challenges: LCG-GAG-DC04.

[6] gLite PackMan: http://agenda.cern.ch/askArchive.php?base=agenda&categ=a043837&id=a043837s1t0/transparencies