Deployment of the CMS software on the WLCG Grid

W Behrenhoff\textsuperscript{1}, C Wissing\textsuperscript{1}, B Kim\textsuperscript{2}, S Blyweert\textsuperscript{3}, J D’Hondt\textsuperscript{3}, J Maes\textsuperscript{3}, M Maes\textsuperscript{3}, P Van Mulders\textsuperscript{3}, I Villella\textsuperscript{3}, L Vanelderen\textsuperscript{4}

\textsuperscript{1} DESY, Notkestraße 85, 22607 Hamburg, Germany
\textsuperscript{2} University of Florida, Gainesville, FL 32611, USA
\textsuperscript{3} Vrije Universiteit Brussel, Pleinlaan 2, 1050 Elsene, Belgium
\textsuperscript{4} Universiteit Gent, Sint-Pietersnieuwstraat 25, 9000 Gent, Belgium

E-mail: wolf.behrenhoff@desy.de

Abstract.

The CMS Experiment is taking high energy collision data at CERN. The computing infrastructure used to analyse the data is distributed round the world in a tiered structure. In order to use the 7 Tier-1 sites, the 50 Tier-2 sites and a still growing number of about 30 Tier-3 sites, the CMS software has to be available at those sites. Except for a very few sites the deployment and the removal of CMS software is managed centrally. Since the deployment team has no local accounts at the remote sites all installation jobs have to be sent via Grid jobs. Via a VOMS role the job has a high priority in the batch system and gains write privileges to the software area. Due to the lack of interactive access the installation jobs must be very robust against possible failures, in order not to leave a broken software installation. The CMS software is packaged in RPMs that are installed in the software area independent of the host OS. The apt-get tool is used to resolve package dependencies. This paper reports about the recent deployment experiences and the achieved performance.

1. Introduction

The CMS detector is a high energy collider experiment operated at the Large Hadron Collider (LHC) at CERN. Analysing the huge amount of data expected to be produced, a large computing infrastructure is required. This infrastructure \cite{1} is built on top of the Worldwide LHC Computing Grid (WLCG), which is organized in a tiered structure. The CMS experiment is supported at the Tier-0 at CERN, 7 Tier-1 centres, 50 Tier-2 sites and still growing number of about 30 Tier-3 sites. To be able to process CMS workflows, all sites require the CMS software being pre-installed. With the exception of only very few sites, all CMS software deployments and deletions are managed centrally. In the following we describe how the deployment process works, explain the automation of various steps and report about recent operation experiences.

2. CMSSW: Packaging and Installation

2.1. Packaging

The main CMS software stack CMSSW is packaged in the RPM (Redhat Package Manager) format. Dependencies of the RPMs are handled with the apt-get tool. A single CMSSW release is a rather monolithic build, which includes almost all base functionalities for Monte Carlo (MC) simulation, event reconstruction and event displaying. A number of external packages can be shared among several CMSSW releases. External packages are for example ROOT...
and GEANT4, MC generators, parton density functions but also a C++ compiler and various libraries.

CMSSW is available for different architectures, where an architecture is defined by a certain compiler version and the system for which it is compiled, e.g. ia32 under Scientific Linux (SL) 4 or amd64 under SL5. Present CMSSW releases are officially supported on SL only. Nevertheless there have been successful installations on various other (mainly Linux) platforms.

When looking at the disk space used after installing one version of CMSSW from scratch, one finds that about 5.5 GB are occupied in 115k files. About 2.5 GB of this space is used by external packages. Further releases require less space because of package sharing with already installed releases. Depending on whether a patch release or major release is added to the installation, a few 100 MB up to several GB of additional disk space are needed.

2.2. Installation

Installation of the CMS software is done in two steps. The first step is called bootstrap and is done only once per architecture to prepare the installation. It first checks that required libraries on the operating system are installed and sets up the package management software and installs common tools which are needed by all CMSSW versions. Because CMSSW is installed under a user account without system administrator rights, packages cannot be added to the RPM database of the host OS. Thus, a separate RPM database within the CMSSW installation directory is created. To fulfil all dependencies of the CMS software on software which is not shipped with CMSSW, the CMS RPM database must know a list of installed system libraries. Therefore an RPM package with the only purpose to fulfil these dependencies is created and installed into the CMS RPM database.

Once the bootstrap has been done, the actual installation can be started. After ensuring that the environment is setup correctly for the use of the CMS RPM database and the CMS versions of the package manager, the apt-get tool can be used to update the package repository and to install a CMSSW release including all required dependencies. This step is repeated, if new releases are installed without doing a new bootstrap.

3. Installation on Grid Sites

The installation of CMSSW on Grid sites is managed centrally by a deployment team. This way, problems that might occur during installation can often be diagnosed and fixed very fast, site administrators are only involved if no solution is possible within the abilities of the deployment team, e.g. fixing misleading site configurations.

At Grid sites the CMS software gets installed on a shared file system that is mounted on all Worker Nodes. Prominent choices for the technology of that shared file system are NFS, AFS or Lustre. The installation on that shared file system works in principal like a local installation. The main difference is that the deployment team has no local accounts at the remote sites and all installation jobs have to be sent via the Grid. These jobs are using the VOMS role lcgadmin to be prioritised and to gain write privileges to the shared file system. Due to the nature of Grid jobs, the installation is running unattended and thus requires a well-tested installation procedure in order not to leave a broken software installation.

In the following we describe the different procedures used for sites with gLite [2] or ARC [3] middleware (Europe, Asia) and OSG [4] sites (America).

3.1. gLite and ARC

For the sites running gLite or ARC middleware a program called release-installer (RI) handles the grid job creation. On the left side of figure 1, it is shown how the RI fetches the list of

\[ \text{data taken from a fresh installation of CMSSW}_{3.6.0} \]
computing elements (CE) it needs to handle. The main input is a manually managed list of
CEs. Experience has shown that generating this information automatically is not feasible. In
principal all CEs are available from the top-level BDII [5], but certain sites do not want to
participate in the central deployment. CMS maintains a special database called siteDB [6] to
track available computing resources. Basing the deployment on siteDB would require a high
discipline to keep the information up to date, which is often not the case.

For all CEs in the list, the RI first checks their status. Only those CEs that are currently
available are considered; the CEs that are in maintenance or unavailable for other reasons are
skipped. Further all those CEs are skipped where a previous job did not run successfully unless
the deployment operator has looked at the logfiles and cleared the error state.

The right side of the figure shows how the RI determines the releases to install. The CMS
TagCollector [7] contains information about all CMSSW releases, particularly whether it is a
current production release or if it is already deprecated. Site specific requests to install additional
releases are read out by the RI from a manually maintained XML file.

SAM tests provide a list of installed releases on a CE, so do the software tags. The list of
installed releases is then compared to the list of production releases and manual requests. An
installation Grid job is created if there is a difference. Once the job is done, its output is parsed
and stored in a SQLite3 database.

3.2. Object oriented deployment framework
To create the Grid jobs and to install the software on gLite and ARC sites, an object-oriented
deployment framework has been newly developed, which consists of about 10 classes and several
thousand lines of code.

All installations steps of the CMS software as described in section 2 are implemented in one
class. Each class instance contains information to maintain a CMSSW installation of a specific
architecture in a given local path. The class takes care about correct environmental variables,
does the bootstrap and updates the RPM database if necessary, and installs or removes CMS
software. Because the class does not know anything about Grid computing, it can be tested
thoroughly on a local directory before being used in Grid jobs.

Information about computing elements is encapsulated in a different class which collects
and caches data from several different sources, for example from the CMS siteDB, the CMS
dashboard, from SAM tests and from the BDII. Thus all information can be accessed via a

![Figure 1. Automated installation on gLite and ARC sites](image)
single interface, and also differences between ARC, CREAM [8] and LCG CEs are hidden.

To deploy CMSSW on Grid sites, a Grid job needs to be created for each installation on each site. Thus a class has been developed to create, submit and retrieve Grid jobs including support for MonALISA [9] based reporting to the CMS dashboard.

The framework is also used by a CGI script to view the deployment status and the logfiles through a web browser. This is especially helpful in case of deployment errors.

3.3. OSG
OSG sites have a different setup and can execute the installation jobs directly on the CE (i.e. not on a worker node). Also a different framework is used, in which the deployment across Tier-2 and Tier-3 sites is triggered from a CRON job. It reads out the list of production releases from the TagCollector and compares it to the list of installed releases for each site which is stored in a MySQL database. As soon as there is a new release, the installation starts automatically.

Manual requests for additional CMSSW versions at a particular site are done via a digitally signed email request.

4. Release deprecation
As the CMS software is constantly developed and improved, old releases become obsolete and need to be removed from the software area. This not only forces physicists to use newer and better releases but also ensures that the amount of disk space needed (see 2.1) does not exceed a certain limit.

The deprecation process starts with a proposal sent to the release announcement mailing list. Unless there are objections, the deprecation announcement is sent typically a week later and the TagCollector is updated. After the deprecation, only software tags are removed from all sites so that no new jobs using the deprecated releases can be submitted. Analysis jobs which have already been submitted should not fail because of missing software, thus removal jobs are not sent immediately but only after a grace period of at least five days. While new releases are being installed on regular basis, deprecation rounds happen only a few times per year and many releases get deprecated at the same time.

5. Experiences
5.1. File Locking
The CMSSW installation of a Grid site usually lives in a shared file system like NFS that is mounted across the compute nodes. It has turned out that file locking, which is required for the RPM database and certain files used by apt-get, often causes problems when used with shared file systems. A work around has been implemented which moves the RPM database directory into a temporary directory on the local machine where no locking problems occur. A symbolic link is then created to link this temporary directory to its original place. With this setup, RPM

Figure 2. Automated installation on OSG sites.
can lock its files locally and do the installation. Afterwards the link is removed and the RPM directory is copied back to its original position in the shared file system.

Working on a copy of the database has the additional benefit that we do not end with a corrupt database even if the Grid job is cancelled at any time because a backup of the original version is restored automatically with the next installation job. Thus the whole procedure is done even if the shared file system at a site supports locking properly.

5.2. Differences in site configuration

A lot of sites have a simple configuration with only one CE and a single CMS software area. The CE publishes the supported architectures and the installed CMSSW releases via software tags which are set right after the installation has finished. However, sites can have more than one CE. These CEs can either share the software area or have separate areas. If they are separate, the software has to be installed on all CEs, otherwise the installation has to be done on only one of the CEs. In the later case, it does not matter which of the CEs is selected to install the software, so that in case of a downtime of one CE the installation can be done on a different CE. This is detected automatically by the release installer.

Not only the software area can be shared among the CEs but also the list of software tags. All combinations are possible, also the configuration of a shared file system and separate tags. In this case software tags need to be set on all CEs although installation was only done on one CE. This has also been automated. However, manual tag correction is still required if errors occur while the tags are changed.

5.3. Typical deployment cycle

Typically, the time from the release announcement to complete installation ranges between 30 minutes and a few hours for most sites. The time variation is due to the size of the release, the performance of the file system, the internet connection of the site, and the site availability. Figure 3 shows a typical deployment progress plot for a patch release. Many jobs have finished in about half an hour, but there is also one site which didn’t receive the installation job immediately and some sites with longer queuing and installation time.

Due to the large number of sites, it sometimes happens that a site is in a downtime, that an error occurs during the installation, or simply that the job is queued for a very long time. On gLite/ARC, deployment jobs are queued for a median of only 5 minutes, although a few outliers with up to 60 hours queuing time lead to a much higher average. Once the job is running, the installation is done in 15 minutes on average.

These numbers show that the completion of a deployment round can take a few days even if the releases have been deployed to 90% of the sites within an hour.

6. Summary and Outlook

Effort has been put into automation of the CMS software deployment. The results are deployment tools which can simplify the whole deployment process to one single command line call in the optimal case or can even be run as a CRON job. If a site is in downtime and did not receive a newly announced release, the tools will find out automatically and install all missing releases once the site is up again. The deployment tools work flawlessly and are still being improved with the aim to further reduce the operational effort. CMS has started to evaluate a deployment method using cernvmfs[10]. In this scheme clients fetch via a Squid cache required experiment software from one central installation and perform additional caching on the local disk.
Figure 3. Typical deployment plot of a patch release. Sites shown twice have two separate software areas.

References
[1] The CMS Computing Model 2004 ed C Grandi, D Stickland and L Taylor CERN-LHCC-2004-035/G-083
[2] gLite. http://glite.web.cern.ch/glite/
[3] Advanced Resource Connector. http://www.nordugrid.org/middleware/
[4] Open Science Grid. http://www.opensciencegrid.org/
[5] Berkeley Database Information Index. https://twiki.cern.ch/twiki/bin/view/EGEE/BDII
[6] S Metson et al 2010 SiteDB: marshalling people and resources available to CMS J. Phys.: Conf. Ser. 219 072044
[7] CMS Tag Collector. https://cmstags.cern.ch/CmsTC/html/
[8] Computing Resource Execution And Management Computing Element. http://grid.pd.infn.it/cream/
[9] MONitoring Agents using a Large Integrated Services Architecture. http://monalisa.caltech.edu/
[10] CERNVM-FS. http://cernvm.cern.ch/