CMS Software Packaging and Distribution Tools

A. Nowack*, RWTH Aachen University, Aachen, Germany
M. Thomas, CALTECH, Pasadena, California, USA
S. Ashby, CERN, Geneva, Switzerland
S. Argirò, CERN & INFN-CNAF, Bologna, Italy
M. Corvo, CERN & INFN, Padova, Italy
N. Darmenov, CERN & INRNE, Sofia, Bulgaria
R. Darwish, D. Evans, N. Ratnikova, FNAL, Batavia, Illinois, USA
T. Wildish, Princeton University, Princeton, New Jersey, USA
B. Kim, University of Florida, Gainesville, Florida, USA
K. Rabbertz, University of Karlsruhe, Karlsruhe, Germany
V. Büge, University of Karlsruhe & FZK, Karlsruhe, Germany
J. Weng, University of Karlsruhe & CERN

Abstract

We describe the various tools used by CMS to create and manage the packaging and distribution of software, including the various CMS software packages and the external components upon which CMS software depends. It is crucial to manage the environment to ensure that the configuration is correct, consistent, and reproducible at the many computing centres running CMS software. We describe the tools used to generate distributable software packages, to track the dependencies between packages and their versions, and to manage their distribution and installation on Tier-0, Tier-1, Tier-2, and other computing centres worldwide.

INTRODUCTION

The Compact Muon Solenoid (CMS) experiment is realized in a large international collaboration of 160 institutes in 37 countries with about 2,000 scientists and engineers. The main goal of the computing effort is the analysis of the data produced by the experiment in order to find new and interesting physical results. This means that enormous amounts of data in the order of several petabytes have to be analysed per year.

In order to achieve the goal, this needs physicists worldwide developing code for various analyses, enormous computing resources, as well as distributed Monte Carlo simulation and data analysis on Grids [1, 2, 3, 4]. This all requires that CMS software can be installed on Grids [5, 6], local clusters, desktop PCs, and even laptops.

CMS Software Projects

The CMS software can be viewed as a complex structure of components. These components can be classified as: a) software developed by the CMS collaboration and b) external tools. We refer to the first class of components as *CMS projects*. External tools are provided by third parties as the HEP community, the open source community or, in one case, a commercial partner.

Each CMS project exists in multiple versions, that are released as the developement advances, and uses certain versions of other projects and/or external tools. As a consequence, the scenario in which several versions of CMS projects and externals co-exist in the same installation is the normal one. Furthermore, this scenario is complicated by the existence of multiple platforms. In the past, Red Hat 7.3 [7] and Scientific Linux CERN 3 (32 bits) [8] coexisted for some time. In the future, there might be also support for 64 bits or MacOS or other platforms.

Software Management

The CMS software projects are managed by SCRAM (Software Configuration, Release And Management) [9]. It is also used to manage projects in LCG [10]. SCRAM manages the multiple versions of a project and the configuration of the dependencies. Furthermore it is used for compiling and linking the project using make.

However, SCRAM does not address the problem of the distribution of the project with all its dependencies to other sites. In addition, the distribution of the binary files (libraries and executables) related to the project for the different platforms is not foreseen by SCRAM.

In short, SCRAM is well suited for developement and release of the CMS projects, but a different approach must be followed to address the problem of the distribution of the projects and of the external software upon which the projects depend. Therefore the software distribution needs another approach which is described in the following sections.

DISTRIBUTION STRATEGIES

Within CMS, two distribution strategies for the CMS software has been developed:

* andreas.nowack@cern.ch

- The DAR (Distribution After Release) tool allows to automatically replicate a given software application runtime environment. All binaries, libraries, and data files needed to run the software application are stored in one self-contained archive, which can be easily deployed on the target machine, reproducing the same runtime environment.

- RpmGen & XCMSI aim at the installation of the full environment. They duplicate completely a given version of a project and its dependencies and set all configuration files. Each component is packaged separately into RPM packages[1] with proper dependencies.

Both strategies have certain pros and cons which make them suitable for different use cases.

DAR provides a slim and compact distribution and is very robust. But on the other hand only the runtime environment is available in a DAR installation, code development is not possible. If various versions are installed which use the same files, these files are distributed and installed several times. It is possible to re-use installations using incremental distributions. But then incremental distributions become not self-contained anymore.

The advantage of XCMSI is that the full environment which allows maximum flexibility, even code development is possible. Already installed components are re-used. But on the other hand a huge amount of data is shipped during the installation which is in most cases not needed. The re-use of components has the disadvantage that a damage of one component can disturb several projects.

USE CASES

DAR on one hand and RpmGen & XCMSI on the other hand are complementary and focus on different use cases. Three different use cases can be distinguished:

- Monte Carlo production,

- code development, and

- user's analysis.

Monte Carlo Production

The Monte Carlo production needs a very reliable environment. Only a limited number of pre-defined executables are used. Furthermore, only a small number of versions of projects are used in parallel. Thus a duplication of files is affordable. Therefore DAR is used for the Monte Carlo production.

Code Development

For the code development, the complete development environment is needed. Usually several versions of different projects are used in parallel. Thus the reduction of duplication of files is desired. Therefore XCMSI is used for code development on user's resources.

User's Analysis

The user's analysis is not really predictable. Depending on the user, a rebuild of the software may be needed on the target machine. Therefore XCMSI has to be used for the installation if rebuilding is needed. In the opposite case, DAR as well as XCMSI can be used.

PACKAGING AND DISTRIBUTION TOOLS

The following paragraphs describe the tools DAR and RpmGen & XCMSI in more detail.

DAR

DAR [12] is a tool to create compact distributions of the CMS software and to install them. It is a set of python modules.

The creation of archives (so-called *DAR balls*) uses the information about the runtime environment for a given application. This includes the management of file system objects which are all needed binaries, libraries, data files, and so on as well as the handling of the runtime environment settings.

The installation of DAR balls is possible in an arbitrary directory for installation and creates shell scripts to set the runtime environment.

DAR supports packaging of incremental distributions based on an existing DAR ball. Incremental distributions contain complete information about the runtime environment, and re-use files from the existing installation. This allows to save disk space and files transfer, while providing the same functionality.

DAR provides interfaces to other tools developed by CMS:

- SCRAM-DAR is the interface to get runtime environment for SCRAM-managed projects.

- RefDB-DAR interfaces to the CMS Production Reference Database to get requests from RefDB and automatically create a new DAR ball.

- A python API is provided to facilitate the creation and installation of DAR files from within other machinery such as the CMS Production system.

RpmGen & XCMSI

The aim of RpmGen & XCMSI is to allow data analysis and software development on laptops, desktop PCs, local clusters, and Grid clusters. The installation of the complex software and the whole development environment should be as easy as possible. This includes that an arbitrary directory can be used for installation. Every user should be

[1] Named after the Red Hat package manager [11].

able to install the software, thus root privileges must not be needed. In order to be able to adapt the environment to the user's needs, the installation should be modular, so that only the actual needed components have to be installed.

RpmGen

RpmGen is the tool for packaging of the CMS software into RPM packages. It creates an image of a reference installation of CMS software. For this purpose, each component is packaged into a single RPM package. The name of the package contains information about

- the name of the component,
- its type (SCRAM-based project or external tool),
- its affiliation with CMS or LCG,
- the platform,
- the version number of the component, and
- the revision number of the package.

This information gives an unique directory name—relative to the installation directory—which is used for the installation of this component on the target machine. The dependencies of the components are mapped to dependencies between the RPM packages.

RpmGen is a set of perl and shell scripts. The master script (ProjectDist.pl) is called in order to create all packages needed for a given project. First, SCRAM is queried for the list of projects and external tools, their version numbers, and their locations. For every project found, the master script is called recursively. For every external tool found and the original project, at least one tar ball containing an image of the reference installation is created. In case of a project, separate tar balls for source code, documentation, and the platform-dependent part are created. The symbolical links are handled with special care. If a link points to a location inside the copied directory, it is translated into a relative link. Otherwise the file or directory that is pointed to is copied as well. The RPM packages for each component are created using the tar balls produced for all components. The RPM packages contain install and uninstall scripts for the configuration of projects. The dependencies to other needed packages are also included, dependencies to system's RPM packages are omitted in order to be independent of the Linux distribution used on the target machine. If an RPM package already exists for a certain component, the tar ball and the package are not re-created. During the process of packaging, several consistency checks are performed. This includes a check for identical directories for already existing packages. Since the version numbers and the directory names follow a certain convention, this convention is checked as well to spot mis-configurations. Finally, the dependencies of all packages are checked for consistency.

The distribution of the RPM packages is done by a web server [13]. Whenever a new set of RPM packages is ready, it is published on the web server and archived on CASTOR by a script. The web server provides dynamical web pages showing all available sets of packages (so-called *download tags*) and all the packages themselves. Furthermore a script for downloading packages and the installation tool XCMSI, the most recent documentation, and statistics about the download of all projects are provided there.

XCMSI

XCMSI [13] is the installation tool for the RPM packages produced by RpmGen. It is a set of perl scripts and comprises command line tools for downloading and installing of the packages, for basic verification of the installation, and a graphical user interface. The command line tools are able to install the CMS software on local machines and to create Grid jobs for installing on Grid clusters. The installation on Grid clusters also supports DAR balls. The graphical user interface offers an easy way of generating configuration files for XCMSI. Usually, only the installation directory has to be set in the configuration file, but there are also various expert options which only have to be changed in rare cases. Furthermore, the graphical user interface allows the selection of projects to be installed and the execution of the command line tools for installation and verification. It is also possible to safely uninstall packages which are not needed any more.

SUMMARY AND OUTLOOK

DAR is successfully used for the CMS Monte Carlo production at all production centres.

RpmGen & XCMSI are widely used for distributing the CMS software on all kinds of clusters and PCs. Currently, more than 300 different versions of various projects are available including old projects for Red Hat 7.3 and all projects for Scientific Linux CERN 3. In total, they consist of more than 1,300 RPM packages with 22 GB. In average 36 projects are downloaded from the web server per day from more than 230 different sites in about 35 countries.

The next steps in the development of the packaging and distribution tools are the following:

- The RPM package creation done by RpmGen is currently being integrated into the automatic build procedure of the CMS software. This will finally provide an automatic creation and publication of RPM packages whenever a new release is available.
- In XCMSI, the validation methods and the error handling are to be improved further.

REFERENCES

[1] P. Elmer et al., "Development of the Monte Carlo Production Service for CMS", CHEP'06, Mumbai, February 2006.

[2] P. Garcia-Abia et al., “CMS Monte Carlo Production in the Open Science and LHC Computing Grids”, CHEP’06, Mumbai, February 2006.

[3] St. Lacaprara al., “CRAB: a tool to enable CMS Distributed Analysis”, CHEP’06, Mumbai, February 2006.

[4] O. Gutsche et al., “Distributed CMS Analysis on the Open Science Grid”, CHEP’06, Mumbai, February 2006.

[5] K. Rabbertz et al., “CMS Software Distribution on the LCG and OSG Grids”, CHEP’06, Mumbai, February 2006.

[6] D. Adams et al., “LHC Grid Computing Project: Requirements on Software Installation”, December 2004, `http://project-lcg-gag.web.cern.ch/project-lcg-gag/LCG_GAG_Docs/SoftInst.pdf`.

[7] Red Hat web page, `http://www.redhat.com/`.

[8] Scientific Linux CERN web page, `http://linuxsoft.cern.ch/`.

[9] SCRAM web page, `http://cmsdoc.cern.ch/cms/Releases/SCRAM/doc/scramhomepage.html`.

[10] LCG Project Applications Area web page, `http://lcgapp.cern.ch/project/`.

[11] E. C. Bailey, “Maximum RPM: Taking the Red Hat Package Manager to the Limit”, Red Hat, Inc., 2000, `http://www.redhat.com/docs/books/max-rpm/`.

[12] N. Ratnikova et al., “Distributing software applications based on runtime environment”, CHEP’06, Mumbai, February 2006.

[13] XCMSI web page, `http://cern.ch/cms-xcmsi`.