



## Software Packaging with DAR

Natalia Ratnikova<sup>a</sup>, Anzar Afaq<sup>a</sup>, Greg Graham<sup>a</sup>, Tony Wildish<sup>b</sup> \*, Veronique Lefebure<sup>c</sup>

<sup>a</sup>Fermi National Accelerator Laboratory  
P.O.Box 500, MS 234, Batavia 60510, IL, USA

<sup>b</sup>Princeton University  
Princeton, New Jersey 08544 USA

<sup>c</sup>CERN/HIP, 1211 Geneva 23, Switzerland.

One of the important tasks in distributed computing is to deliver software applications to the computing resources. DAR, Distribution after Release tool, is being used to package software applications for the world-wide event production by the CMS Collaboration. This presentation will focus on the concept of packaging applications based on the runtime environment. We discuss solutions for more effective software distribution based on two years experience with DAR. Finally, we will give an overview of the application distribution process and the interfaces to the CMS production tools.

### 1. Introduction

Compact Muon Solenoid CMS HEP experiment [1] will run at the LHC accelerator at CERN. CMS is using Grid technologies [2] to utilize available computing resources for the world-wide distributed mass production.

To make this possible software applications must be brought to the production sites. We want to have an automated way to create self-consistent distributions of the software applications, based on the software releases installed at CERN.

The Distribution After Release DAR tool was developed at Fermilab for quick-and-easy deployment of the software applications, which can run on the systems that do not have pre-existing application specific environment.

The concept and first tool prototype were proposed in 2001 [3], and since the end of 2001 the tool was used for packaging and installation of the software applications in the CMS distributed Monte Carlo Event Production.

### 2. DAR Concept

The distribution unit is an application, which is considered to be a complete, self-contained software program, including required shared libraries and other files. Applications are executed in a particular runtime environment and accomplish a particular computing task [4].

One important and natural requirement assumed for the distributed computing on the Grid, is that software applications should be relocatable, i.e. software application could be installed and executed in the arbitrary location in the file system visible on the worker node (ref to alternative approaches). This complies with the Grid architectures, where the disk space required for the software installation can be allocated by the resource broker, along with other resources such as CPU time, etc.

We proceed from the assumptions that

- relocatable software does not contain hard-coded absolute paths in the program or in the shared libraries (except those referred to the system area)
- all required executables are found in the locations specified in the PATH environment

---

\*Present address: CERN-EP/CMS, 1211 Geneva 23, Switzerland.

variable, which is extended appropriately for each given application

- distributions containing pre-compiled binaries also rely on the operating system compatibility.

Most of real quality software products are relocatable, and the actual locations of the software components for a given installation are usually defined in the software configuration parameters. One of the standard ways to pass the configuration information to the application during the runtime is through the use of the UNIX shell environment variables, such as `PATH`, `LD_LIBRARY_PATH`, and others.

DAR is using the set of the runtime environment variable specific for a given application in order to decide which files need to be packaged into the distribution DAR file. The DAR file is then delivered to the working site, and can be installed in any new directory. The runtime environment for the application is set using script generated by DAR during the installation.

### 3. DAR Implementation

DAR distinguishes between three types of the runtime environment variables, depending on the value:

- The variable value is associated with some path to the existing file or directory in the local file system.
- Variables specifying a list of paths in the local file system (`PATH` -like variables), where entries are separated by the colon delimiter.
- Variables set to simple values not associated with any existing object in the local file system. These could be for example special flags controlling the execution mode, URL addresses, and other parameters that may be used during the execution time.

All physical files and directories found in the locations specified through the runtime environment variables are copied into the distribution, preserving the underlying directory structure.

In case of `PATH`-like variables DAR walks through the specified list of paths and copies all contents into separate directories.

During installation DAR generates shell setup environment script to be used later to initialize the application environment according to the actual location of the software installation. The directory structure and the order of paths in the `PATH`-like variables are preserved to guarantee that the application will pick the same objects as in the original environment. For those variables, which are not associated with any files or directories, DAR will keep the original value. Finally DAR generates a list of included files with the indication of the checksum and location relative to the top of the installation directory.

Thus DAR provides a generic way to replicate both the application and its environment in the new location.

#### 3.1. Optimizations

Of course the resulting DAR file will likely contain superfluous directories and files. DAR provides options for more selective packaging.

The same files can be referred through different environment variables. To avoid multiple copies in the distribution, DAR recognizes these situations and includes only one instance of the file into the distribution, and substitutes other references by symbolic links.

The erase option allows expert to remove files or directories that are formerly referred by the application environment, but are known to be not necessary for running the application. The example of such files are `*.html\`; `*.ps\`; `*.pdf\`; `CVS\` etc . In general for detection of files, that could be safely excluded, expert's knowledge of the software application is required. It may take several iterations to figure out what can be removed, and whether it is efficient and safe.

#### 3.2. Tests

The primary goal of the DAR distribution is to reproduce the same application environment in different locations. The measure of success of the packaging is a reproducible operation of the application. Two tests are usually applied.

First we compare output from the application

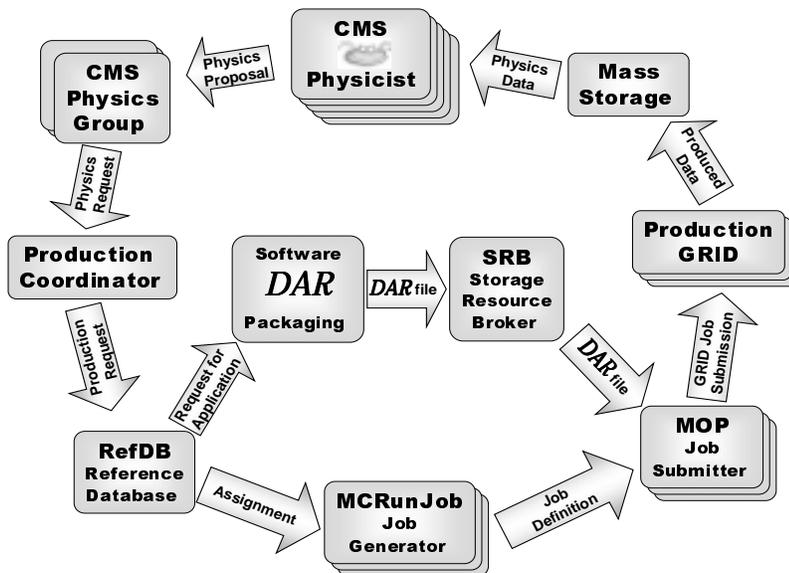


Figure 1. The CMS physics data production cycle.

executed in the native environment and from the application installed from the distribution DAR file and executed on the same node. The difference in produced output, if any, could either indicate that the application is not truly relocatable, or that the use of expert options broke the consistency of the application.

After the DAR distributed application successfully passes the test on the same node, the same application running on a separate node is tested. The difference in this case usually indicates some inconsistency of the system setup on different nodes.

To insure that the installation itself was not corrupted, one can always compare the contents of the installation directory against the list of files and their checksums provided by DAR.

#### 4. Using DAR in CMS production

DAR created distributions are being used as a mandatory way to install software for the official CMS Monte Carlo production. Using the same set of applications and consistent software dis-

tribution mechanisms insured stable performance and trustworthy results.

##### 4.1. CMS physics data production cycle

The general scheme of the CMS physics data production cycle is presented in Figure 1.

Physicist proposes, and the corresponding physics group approves request for the Monte Carlo production. The request is formalized and the database RefDB [5] is filled with all the necessary information including applications to be run and running parameters, amount of data to be produced etc. This information is used to make a request for creating of the corresponding software distribution. DAR creates the necessary distribution DAR-file and put it into the Storage Resource Broker [6] which provides a world wide access to the file.

Job generator MCRunJob [7] converts production assignment into a set of scripts which will be actually run on the farm. Next link is a Job Submission (MOP [2]), which is a layer between the CMS Production system and Grid computing resources. MOP takes care that required DAR

distribution is installed on the Production Grid [2] and then submits requests prepared by the MCRunJob.

After the data are generated and processed on the Grid, the processing summary is stored back into the RefDB, and produced data are stored in the mass storage system. This completes the cycle and physicist can verify and analyze data, generated according to the original request.

#### 4.2. Creating DAR Distribution

Request for the DAR file is initiated by the Production Coordinator based on the original production request. This triggers a creation of the DAR file based on the corresponding software release installation at CERN.

The RefDB-DAR interface has been developed to formalize the requests for applications and provide bookkeeping of the available distributions.

The RefDB-DAR interface allows to download request file from the RefDB. The `refdbdar` utility is then used to parse and validate the RefDB request file, builds requested executables, establishes corresponding environment. Then it uses DAR to package the application. The resulting DAR file is verified and stored in the SRB [6] for distribution.

#### 4.3. Installing DAR Distribution and Job Submission

Production sites get the assignments with the indication of the required DAR file. DAR-ball is then downloaded from the SRB and installed, using DAR, on the worker nodes.

MOP is a system for distributing CMS Monte-Carlo production jobs over the Grid.

MOP has capability of running any type of scripts (jobs) at remote Grid sites, called Worker Sites.

MOP run jobs as DAGs (Directed Acyclic Graphs) which could be combined together to create complex workflows.

In general every DAG contains four stages:

- Stage-in: Bring in the required input files (from several sources) to the worker site.
- Run: Execute the job itself, producing results, logs, data.
- Stage-out: Send out produced results, data and log files.
- Clean-up: Clean the left over files and directories at worker site.

DAR installation at a worker site is achieved by creating a special MOP job that first pulls DAR tool and application DAR distribution in stage-in, runs installation by invoking DAR in run-stage, brings back the results of the installation to the submission site in stage-out, and then performs clean-up operation at the worker site.

#### 5. Conclusions

DAR-based distribution scheme is successfully used in the CMS event production for an extended period of time. It allows to keep the pace with the software developments and deliver software applications to the production sites with ease and in a timely fashion. Being re-packaged into RPM files, applications can be re-used within different distribution approaches (e.g. LCFG).

#### REFERENCES

1. CMS Experiment, see for example: <http://cmsdoc.cern.ch/cms/outreach/html>
2. Gregory E. Graham, et al, The CMS Integration Grid Testbed. CHEP03 proceedings, CHEP03, La Jolla, March, 2003
3. Natalia M. Ratnikova, Gregory E. Graham, CMS Software Distribution and Installation Systems: Concepts, Practical Solutions and Experience at Fermilab as a CMS Tier 1 Center. Proceedings of CHEP01, Beijing, September, 2001
4. N. Ratnikova, A. Sciaba, S. Wynhoff, Distributing Applications in Distributed Environment. NIMA, Volume 502, No. 2-3, (April 2003) 458-460.
5. V.Lefebure, RefDB: A Reference Database for CMS Monte Carlo Production. CHEP03 proceedings, CHEP03, La Jolla, March, 2003.
6. Storage Resource Broker. <http://www.npaci.edu/DICE/SRB>
7. G.Graham, MCRunJob: A Workflow Planner for Grid Production Processing. CHEP03 proceedings, CHEP03, La Jolla, March, 2003