



KnowARC D5.1-1

10/8/2007

GUIDE TO THE NORDUGRID BUILD SYSTEM

Technical description and Installation Guide

Anders Wäänänen*

1 Introduction

This document gives a technical overview of the NorduGrid build system. In section 2 we describe the core build tool mock and section 3 describes how to configure the Grid runtime environment for using the build system. Section 4 has examples on how to submit builds with some helper scripts and section 5 show how to setup a system to monitor the builds.

2 Mock

Mock [1] was originally made for the Fedora Extra - an add-on repository for the Fedora Core Linux distribution. It is a simple wrapper around the Unix command `chroot`. Mock uses yum to create a base build system - a full minimal Linux tree under `/var/lib/mock`, mounts the pseudo filesystems `/proc` and `/dev` and then calls `chroot` to run the `rpmbuild` command with the `src.rpm` given on the command-line. The build thus takes place in a clean build environment void of site specific modifications.

The NorduGrid build system interfaces the ARC job submission to mock using runitme environments as described in section 3.

Mock can be made to create build environments for other distributions than the Fedora Core which was the original target. This has been worked out for the NorduGrid build system which currently supports mock building for the following distributions:

- Fedora 1, 2, 3, 4, 5, 6, 7
- Red Hat Linux 7.3, 9
- Red Hat Enterprise Linux 3, 4, 5
- OpenSuSE Linux, 10.2

for both architecture `i386` and `x86_64` where appropriate for a total of 22[†] platforms. Mock configuration files are created on the fly for each Grid job based on specifications in the job.

Mock has the ability to cache build environments such that the base systems do not need to be created for each build. This is used to some extend, but only within a single Grid build. Each Grid build supports building of multiple inter-dependent RPMs and can thus take advantage of previously built RPMs. A full Linux distribution can thus in principle be rebuild in a single Grid job.

Mock uses yum to download and setup RPM packages for the build environment. This can put enormous load on the yum repositories used by a build cluster. The NorduGrid build system utilizes various optimisations to work around this including local yum mirrors for all platforms and squid web caches on the worker nodes.

Mock is built around yum and rpm and can only produce rpm packages. It can even be tweaked to create packages on systems where rpm is not the native packaging tool such as Debian and Ubuntu. However, on these systems it would be much more appropriate to use a tool similar to mock but with deb as the underlying packaging format. Fortunately such a tool exists (`pbuilder`). Together with `debootstrap` the NorduGrid build system could be extended to support deb packaging.

3 Runtime Environment

The package build runtime environments (RTEs) are organised as:

```
/APPS/PKGBUILD/<vendor>-<release>-<architecture>
```

where `vendor` is the operating system vendor (eg. redhat), the `release` is the operating system release (eg. el4) and `architecture` is the operating system base architecture (`i386` or `x86_64`).

[†]Fedora 1 and 2, and Red Hat Linux 7.3 and 9 does not have 64 bit versions

The RTE scripts are hard linked[‡] to a single script called `.BASE`. For every build platform that is to be supported one should create a hard link to the `.BASE` script. This script is configured by a script called `.site` in the same directory. Here one can configure the location of the helper scripts and the location of the distribution software repository:

```
BUILD_CONFIG_DIR=/home/apps/mock/config
REPO_URL=ftp://ftp.example.com/myrepo
```

It is a very good idea to create local mirrors of the distributions one wants to build for in order to speed up the build. Running a web cache like squid on each compute node is also a good idea.

The RTE script replaces any user specified executable with the build script `build_command.sh` located in the `BUILD_CONFIG_DIR` directory specified in the `.site` file. The build script is a wrapper around mock which does:

1. Run any specified `.pre` scripts
2. Create mock configuration file
3. Run mock for each `src.rpm` uploaded in alphabetical order
4. Run any specified `.post` scripts

4 Clients

The build jobs are ordinary Grid jobs which requests the appropriate runtime environment.

A helper application: `srcrpm2xrsl` has been provided to facilitate the creation of the xRSL scripts. In it's simplest form it can be invoked as:

```
srcrpm2xrsl -d redhat-el4-i386 mypackage.src.rpm
```

The script will parse the `src.rpm` for various information including the produced binary RPMs. The above example will create an xRSL which describes a Grid job that will produce binary builds of `mypackage.src.rpm` for Red Hat Enterprise 4.

The produced xRSL will rename the input files on the compute element so that the `src.rpms` will be built in the specified order.

The NorduGrid nightlies uses the script above in a cron job to create builds every 24 hours.

5 Build Monitoring

The build monitoring system consists of a standard web server with PHP support and a single PHP script which must have direct read access to the built packages. The `index.php` script should be saved somewhere appropriate (eg. `/var/www/html/builds`). Site configuration is handled by a file called `site.php` in the same directory. This file should look something like:

```
<?php
$topdir = "/var/www/ftp/software";
$topurl = "http://download.nordugrid.org/software";
?>
```

[‡]To create a hard link in the package runtime environment directory simply do: `ln .BASE redhat-el4-i386`

where `topdir` is the base location for the directory tree holding the built packages and `topurl` is the corresponding web location. The subtree itself is organized such that binary packages are located in:

```
<topdir>/<package>/<type>/<version>/<vendor>/<release>/<architecture>
```

and source packages are in:

```
<topdir>/<package>/<type>/<version>/src
```

`<package>` is the package name, `<type>` is on of: (releases,nightly,testing), `<version>` is the package version, `<vendor>` is something like (redhat, fedora, opensuse, ...), `<release>` is the operating system release (eg. 7.3, 9, el4)

The web server must be configured with support for PHP in the chosen location.

5.1 Notes on web and ftp server setup

When configuring SSL for the Apache web server use:

```
SSLCertificateFile    /etc/grid-security/hostcert.pem
SSLCertificateKeyFile /etc/grid-security/hostkey.pem
SSLCACertificatePath  /etc/grid-security/certificates
SSLVerifyClient       optional
```

It can also be useful to have an ftp server in parallel with the http server for serving packages. A common popular ftp server is vsftpd. To hide the log files from direct download the following configuration lines can be used:

```
hide_file={.listing,output}
deny_file={.listing,*output*}
```

References

[1] Mock. [Online]. Available: <http://fedoraproject.org/wiki/Projects/Mock>