Application Deployment, Packaging, and Portability(DRAFT)

This document attempts to describe requirements for application portability, packaging, and deployment in a Grid environment for IF experiments at Fermilab. The model of sending all of the software for a grid job with each job is expensive in terms of network bandwidth, so mechanisms which allow software to be placed on grid sites in advance and then used by experiment jobs is preferred.

This document will address each of the issues of deployment (how software gets to the grid worker nodes), packaging (how software is bundled for distribution/deployment) and how software should be built/constructed for portability.   These all implicitly involve the issue of software dependencies – identifying what software needs what other software to run.

Deployment

Applications can be accessed from a grid job in several ways:

1. already installed locally (via rpm) on node
2. accessed via OSG $APP area.
3. pulled down from a “neighborhood” SRM.
4. included in the tarball sent with the job submission

Of course, for a grid job to use approaches 2 and 3, some sort of preliminary software upload job would need to have been run to put the software near the job, and to safely rely on item 1, some sort of check job should have been run to ensure the existence of said RPM's on the system. This secondarily implies the necessity of cleanup of old, unused software previously sent out.

Packaging

Several widely used mechanisms exist:

1. ups/upd packages
2. tarballs from a Well Known Source
3. rpms

Such a system should take advantage of these package formats to avoid needing to repackage software.

Portability

- Usual Approach
  - Build on oldest/narrowest available system (i.e 32 bit SLF4)
  - Make sure compat-* rpms are known to be dependencies
- Linux Standards Base (LSB)
  - Tools to build software
  - all non LSB dependencies identified
  - works many more places

For IF experiments using OSG, the first approach is likely sufficient for the foreseeable future. Some experiments are also pursuing building on newer/wider systems, while this is useful for interactive use, it is only a complication of the batch submission universe, where we should concentrate on using the most portable code.

Components needed

Overall, deployment solution will need several components:

- Software availability service
- Installation/checking job
- Job tarball creator

Each of these will be discussed in more detail.

Software Availability Service

Overall, we need to know what software is available at a given Grid site, and how it is available -- either locally as rpms, installed in the OSG $APP area, etc. Without this knowledge, we must send every dependency of our job along when we package and send the job tarball. An installation/checking job must be able to report back to this service to provide this information, and the tarball creator must be able to query it to decide what to include in a job tarball. Similarly, if software is cached at a given SRM for use at a given site, the availability service should know that that software is obtainable there by a running job.

Installation/Checking Jobs

We need to be able to submit a job with suitable credentials (i.e. VO-admin) to install software in the $APP area. Then a separatejob, running with normal user credentials, to report back about what software is available to users with that credential on local nodes at a given Grid site. This includes checking $APP, checking locally installed rpms, and verifying downloads from any local SRMs that are defined. It should be noted that for this to be possible, some standardizing of the directory tree structure under the OSG $APP area is called for, which would be shared between the software install and software check jobs, and which will be assumed by the jobs being submitted.

Tarball creation

While many requirements of this functionality are described elsewhere, for application deployment, we wish the tarball creation utility to be able to omit software components known to be already installed where the job is going to be submitted. This can result in significant reduction in per-job application data transfers.

Appendices

Notes on $APP

(from http://osg-docdb.opensciencegrid.org/0003/000379/001/ComputeElement.pdf)
"Installation of software on $APP are best handled by providing role based access to a batch slot that is dedicated for this purpose on the fileserver that exports $APP. A simple example implementation of this is to have a special batch queue that keys in on the uid a user is mapped to. This is in production at UCSanDiegoPG since OSG 0.2.1. Sites may provision one such batch queue for each of their primary client VOs plus another one shared among all other VOs allowed on the site. The queues then send their jobs to dedicated batch slots on appropriate pieces of hardware. The intention here is two-fold. First, write access to $APP should be allowed to a privileged subset of the VO users, the VO admin, rather than all VO users. Second, installation can be time and resource intensive. It is desirable to not have this activity happen as fork jobs on the CE."