LOFAR Cluster Contributed Software Guidelines

There are science or development groups that would like to add pieces of software to the suite of software currently available on the cluster (see here).

Here we provide some guidelines.

OS and library version limitations

The cluster runs on Ubuntu 8.04 LTS. our policy is to be restrictive in sofar as using non-distribution supplied versions of libraries of tools. In some cases, new versions can be added to the system without problems (e.g., the Pythonlibs) , but we have also seen cases (e.g., Boost 1.40) were build systems were not able top properly cope with the available mix of old and new versions, resulting in build failures, link failures or runtime failures.

It is therefore advised to be aware of the limitations of the current OS in terms of supplied library versions, and to to the best effort to have your code built using the available versions of libraries. If this is not possible, we can try to accomodate a newer version for you, but we cannot guarantee that this will not cause any problems for other applications.

We will certainly **not** provide newer version of the C/C++ compiler (gcc, g++) and associated libraries.

Build requirements

Regular builds

If your package development process profits from a daily or weekly build, we can accomodate that by setting up a build script that can be run from a cron job. We currently have this available fro the LofIm and the LUS packages, abut others can be added as well. But please think about the impact of a failed build when you have multiple users dependning on your software. We can also accomodate fixed releases next to regular builds. That allows your users to have a stable version for day-to-day work, and a state-of-the-art version for testing.

Embedding new applications or libraries in other packages

If you have some new functionality that must be added to and embedded in some other package, please arrange that with the package owner. If additional env. vars are needed for running the new code, that must be communicated to us as well.

Please be aware that your application or library should be buildable in the build framework of the package to which you add it on our systems. Carefully check library version requirements of your add-

on against the requirements of the package to which you add it. Resolve conflicts as much as possible so the code can build and run on the available systems with the available library versions.

Package initialization

Often packages require PATHs to be set correctly, or have library paths added to the user's `LD_LIBRARY_PATH` environment var. Sometimes, packages require custom environment vars to be set. This can be accomodated by us, as long as it is clear what these env. vars. should be for your package.

Most packages have an csh-based initialization script in the directory `/opt/scripts`, which through some wrapper code is made to work also for bash-shell users. Adding new env. vars. requires some manual work, though, so if this is needed, let us know.