LTA Pipeline Environment Libraries

The following libraries with given versions are installed in a local directory on Juropa (for now /lustre/jhome17/htb00/htb003/LOFAR-R14-P275 will be moved to the top level home directory of htb003 most likely).

Library	Version
bison	2.5
blitz	0.10
boost	1.44.0
cmake	2.8.5
casacore	trunk
casarest	8741
cfitsio	3240
fftw	3.2.2
flex	2.5.35
hdf5	1.8.4
libpng	1.5.6
libpqxx	3.1
log4cplus	1.0.4
matplotlib	1.2.1
m4	1.4.16
monetdb python client	11.15.7
numpy	1.7.1
OpenBLAS	0.2.5
PostgreSQL	9.1.2
pyrap	trunk
python	2.7.5
scons	1.3.0
setuptools	0.6c11
wcslib	4.4.4
unittest-xml-reporting	1.5.0
pyfits	3.1.2
pywcs	1.10.2
scipy	0.12.0
argparse	1.2.1
libiberty	
LOFAR	1.14

LTA Installation on Juropa

The operating system is:

```
SUSE Linux Enterprise Server 11 (x86_64)
VERSION = 11
```

```
PATCHLEVEL = 1
```

With kernel version:

```
Linux 2.6.32.59-0.3-default x86_64 GNU/Linux
```

The current working installation is in:

```
/lustre/jhome17/htb00/htb003/LOFAR-R14-P275
```

Some things have to be Changed in order to compile everything on Juropa.

General Compile settings; Libiberty

There are unresolved issues with older versions. Tests with gcc4.3.4 and gcc4.4.6 gave the error:

```
File "[install_dir]/lofar/release/lib/python2.6/site-
packages/lofar/parmdb/__init__.py", line 112, in getDefValues
return self._getDefValues (parmnamepattern)
TypeError: No to_python (by-value) converter found for C++ type:
casa::Record
```

The compiler suite has to be changed to gcc4.6.3. You do this with the command:

```
module load GCC/4.6.3
```

You also have to load the gsl module

```
module load gsl
```

Set environment variables:

```
export CC=/usr/local/gcc/gcc-4.6.3/bin/gcc
export CXX=/usr/local/gcc/gcc-4.6.3/bin/g++
```

Using gcc4.6.3 gives the error

```
  File "[install_dir]/local/lib/python2.6/site-packages/pyrap.tables-
trunk_r332-py2.6-linux-x86_64.egg/pyrap/tables/table.py", line 1023, in
addcols
    self._addcols (tdesc, dminfo, addtoparent)
TypeError: No registered converter was able to produce a C++ rvalue of type
int from this Python object of type numpy.int32
```

This can be corrected by changing the order of the “pyrap.tables” import in the node script “imager_awimager.py”

Change from:

```
from __future__ import with_statement
import sys
import shutil
import os
import subprocess
import copy
from lofarpipe.support.pipelinelogging import CatchLog4CPlus
from lofarpipe.support.pipelinelogging import log_time
from lofarpipe.support.utilities import patch_parset
from lofarpipe.support.utilities import catch_segfaults
from lofarpipe.support.lofarnode import  LOFARnodeTCP
from lofarpipe.support.utilities import create_directory
from lofarpipe.support.data_map import DataMap
from lofarpipe.support.subprocessgroup import SubProcessGroup

import pyrap.tables as pt
```

to

```
from __future__ import with_statement
import sys
import shutil
import os
import subprocess
import copy
import pyrap.tables as pt
from lofarpipe.support.pipelinelogging import CatchLog4CPlus
from lofarpipe.support.pipelinelogging import log_time
from lofarpipe.support.utilities import patch_parset
from lofarpipe.support.utilities import catch_segfaults
from lofarpipe.support.lofarnode import  LOFARnodeTCP
from lofarpipe.support.utilities import create_directory
from lofarpipe.support.data_map import DataMap
from lofarpipe.support.subprocessgroup import SubProcessGroup
```

The change of the compiler suite brings additional problems. The system paths change to custom locations which prevents the finding of the correct version of the library “libiberty.a”. The one found is missing the compiler flag -fPIC so it can be linked dynamically. You have to compile it yourself (or copy the correct one from the old path ←- needs checking).

PYRAP

Pyrap will not compile because it will not find the glibc libraries as they are not in the standard location anymore. To correct this you have to edit the pyrap SConsript in pyrap/libpyrap/trunk from

```
env = Environment(ENV = { 'PATH' : os.environ[ 'PATH' ],
                          'HOME' : os.environ[ 'HOME' ]
                          },
```

to

```
env = Environment(ENV = os.environ,
```

Note: you have to use the trunk version and not the latest release because otherwise pyrap tables wont work. Also add the the compile option –enable-rpath. Use “batchbuild-trunk.py” for installation instead of “batchbuild.py”.

FFTW

For the fftw library you have to add the option “–enable-threads” for the compiler.

SCIPY

For scipy add the environment variable UMFPACK=“None”.

Blitz

Version 0.9 did not compile. Version 0.10 works fine.

CMake

Tried using the system installed CMake which is also version 2.8.5. There are Problems with some “find” scripts (LAPACK,BLAS). I remember casacore and Lofar would not install as they should, forgot the details though as the self installed version does what it should. Maybe one should look into it in the future because what CMake to use should not matter.

Casacore

Need to use the trunk version instead of release 1.5 for functionality. Using the buildscript options, Casacore is missing support for hdf5 which is needed for the final steps of the imaging pipeline. Added hdf5 support as well as OpenMP and Threads by hand in the ccmake interface. ToDo: add options to buildscript

LOFAR

The system installed libpng was hiding the selfcompiled version which lead to problems during program execution. The proper path to libpng library and include dir have to be set in cmake. Same goes for the selfcompiled libiberty. ToDo: add options in cmake.build file (at least for libpng since its part of standard installation).

The variants file GNU.cmake has to be edited to set the path to the new compiler version

```
# GNU compiler suite
set(GNU_COMPILERS GNU_C GNU_CXX GNU_Fortran GNU_ASM)
set(GNU_C         /usr/local/gcc/gcc-4.6.3/bin/gcc)      # GNU C compiler
set(GNU_CXX       /usr/local/gcc/gcc-4.6.3/bin/g++)      # GNU C++ compiler
set(GNU_Fortran   /usr/local/gcc/gcc-4.6.3/bin/gfortran) # GNU Fortran
compiler
set(GNU_ASM       /usr/local/gcc/gcc-4.6.3/bin/gcc)      # GNU assembler
```

SSH support

Logging into compute nodes via ssh is not permitted on the system. Subprocesses have to be started on the one rented compute for now via shell or mpiexec command. Distribution to multiple nodes is in the works.

lofarpipe/support/remotecommand.py has to be edited to circumvent ssh for locahost job spawning (svn diff useful?)

File copy

Since the Juropa cluster uses a shared filesystem every data should be (read: HAS to be) present at job start to not waste computing time. The login nodes are supposed to be used for job preparation and analysis afterwards. So we do not need to copy data to the working directory (quota is limited!). The change for that is in lofarpipe/recipes/nodes/imager_prepare.py (again svn diff).

GSM Database

A local version of the GSM Database has to be used. At the moment it is running on Juropa02 but has to be restarted after downtimes or after the demon used up its 30min wallclock time. The database is in the home folder of user zdv596 and can be started with

```
cd /lustre/jhome9/lofar/zdv596
monetdbd start gsm
monetdb start gsm
```

monetdb is installed in /lustre/jhome9/lofar/zdv596/LOFAR-Release-1_14/local/bin

How to install a local GSM Database take a look at this
http://www.lofar.org/wiki/doku.php?id=lta:software_stack_installation#gsm_database_installation

Notes on Processing at the Juropa Cluster

This is going to be the Wiki page for the Lofar Software installation at the Juelich Supercomupting Centre.I will update this page as the installation progresses.

Currently the software is beeing tested on the Juropa system. Everything that is part of the Calibration and the Target Pipeline is working. The awimager causes problems but might be working in an experimental build (details at the end).
The software is available in the home directory of user zdv596. The root path of the install is

/lustre/jhome9/lofar/zdv596/LOFAR-Release-1_14

You can find the lofar software in “lofar/release”. The environment you need is loaded with the script “variables_lofar.sh”

In addition you might need a copy of the measurement data /lustre/jhome9/lofar/zdv596/dataCEP in your home directory and point to it in a file .casarc (just contains:“measures.directory: [yourhome]/dataCEP”)

If you require access to the GlobalSkyModel database, there is a copy of the database from the CEP Cluster (hopefully) running on the Juropa login node jj28l02. Access the databse “gsm” on port 51000 with user “gsm” and pass “msss”

How to keep the measurement and gsm data up to date and distributed has to be discussed

You can now run and test the executables on the login node from “lofar/release/bin” or run python scripts (your own or pipeline scripts in “local/lib/python2.7/site-packages/lofarpipe/recipes”).

To run your jobs on the compute nodes you first have to setup and submit a job via the batch system. A detailed description can be found on the Juropa homepage
'http://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JUROPA/UserInfo/QuickIntroduction.html'

Here is a simple example of the procedure. Basically you use two scripts. One to configure the job and one to setup the environment for your program and run it.
Job configuration is pretty basic right now because we can only utilize one node per job. Do not get confused by the use of comments '#'. The '#' in front of MSUB commands is necessary for the command to be recognized from the Moab batch system.
You submit the job with the command “msub [yourscript]”. Check your status with “showq -u 'username' ”. To see the whole machine with a gui try the “llview” program.

Contents of 'lofarmsub.sh':

```
#!/bin/bash -x
#MSUB -N Lofar-test
# just the name
#MSUB -l nodes=1:ppn=8
#MSUB -l walltime=00:30:00
#MSUB -e error.txt
# if keyword omitted : default is submitting directory
#MSUB -o output.txt
# if keyword omitted : default is submitting directory
#MSUB -M your@mail.de
#Mailadress
#MSUB -m eab
#send mail on end, abort, begin
```

```
./lofarCalibratorPipelinePy2.7.sh
```

The walltime is the time your job will be running on the machine. If it is to low and the job is not finished it will be killed. Is it to high your job might have to wait longer in queue but only the real computing time will be booked.
The maximum walltime is 24h.
The number of nodes has to stay at 1 for the time being. You can experiment with ppn (process per node) which is used for openmp enabled programs.
It is best to name the log files error and output with some job specific parameters and maybe the date.
You can choose to have mails send to you about the status of your job.

Contents of 'lofarCalibratorPipelinePy2.7.sh':

```
#/bin/sh!
#start of jobscript
export OMP_NUM_THREADS=8
#
#
export
PYTHONPATH=/lustre/jhome9/lofar/zdv596/LOFAR-
Release-1_14/local/lib/python2.7/site-packages:$PYTHONPATH
export
PYTHONPATH=/lustre/jhome9/lofar/zdv596/LOFAR-
Release-1_14/lofar/release/lib/python2.7/site-packages:$PYTHONPATH
#
export PATH=/lustre/jhome9/lofar/zdv596/LOFAR-Release-1_14/local/bin:$PATH
export
PATH=/lustre/jhome9/lofar/zdv596/LOFAR-Release-1_14/lofar/release/bin:$PATH
export
PATH=/lustre/jhome9/lofar/zdv596/LOFAR-Release-1_14/lofar/release/sbin:$PATH
#
export
LD_LIBRARY_PATH=/lustre/jhome9/lofar/zdv596/LOFAR-
Release-1_14/lofar/release/lib:$LD_LIBRARY_PATH
export
LD_LIBRARY_PATH=/lustre/jhome9/lofar/zdv596/LOFAR-
Release-1_14/lofar/release/lib64:$LD_LIBRARY_PATH
export
LD_LIBRARY_PATH=/lustre/jhome9/lofar/zdv596/LOFAR-
Release-1_14/local/lib:$LD_LIBRARY_PATH
export
LD_LIBRARY_PATH=/lustre/jhome9/lofar/zdv596/LOFAR-
Release-1_14/local/lib64:$LD_LIBRARY_PATH
#
export LOFARROOT=/lustre/jhome9/lofar/zdv596/LOFAR-Release-1_14
#
/lustre/jhome9/lofar/zdv596/LOFAR-
Release-1_14/lofar/release/bin/msss_calibrator_pipeline.py
/lustre/jwork/lofar/zdv596/Pipeline_testdata/calibrator_pipeline/Observation
64405
```

```
-c
/lustre/jhome9/lofar/zdv596/LOFAR-
Release-1_14/lofar/release/share/pipeline/pipeline.cfg
--job calibrator_branch_regression -d
```

Simply replace the pipeline call with the command you want to run in your job. Example of Alexanders bbs test: /lustre/jhome9/lofar/zdv596/LOFAR-Release-1_14/lofar/release/bin/calibrate-stand-alone -v -n -f L104244_SB200_uv.dppp.MS BBS.parset skymodel.parset One important remark for your working directory. Use the Filesystem mounted under $WORK for your data and jobs.
From the Juropa home page:
$WORK
File system for large temporary files with high I/O bandwidth demands (scratch file system). No backup of files residing here. Files not used for more than 28 days will be automatically deleted!

If you want to try the awimager try another build directly in the home directory /lustre/jhome9/lofar/zdv596/lofar/release.
There are some weird problems with pyrap and this version uses a different compiler. The awimager seems to work but there are other problems which extend has not been analyzed yet.

I hope these information are sufficient for some first tests and experiments.
Good luck and let me know of any problems and feel free to give some feedback.

on holidays for one week... after that more updates

From:
https://www.astron.nl/lofarwiki/ - **LOFAR *Wiki***

Permanent link:
https://www.astron.nl/lofarwiki/doku.php?id=public:processing_at_juropa&rev=1377867862

Last update: **2013-08-30 13:04**