Prefactor tutorial

the data

The data for this tutorial come from a standard LOFAR surveys observation (i.e. an 8hr observation of two target pointings with two simultaneous beams each with 48 MHz bandwidth, bookended by two 10 min calibrator observations)
we will look at one beam and consider only a subset of the data (in the interest of processing times)
we will use one of the calibrator observations for the calibrration

The calibrator is 3C196 with obsid L232873
The target is field P23 with obsid L232875

The RO preprocessed data (i.e. that which can be retrieved from the LTA) can be found on CEP3 in

```
/data010/scratch/wwilliams/lds18_pft/data/L232873
```

containing 100 subbands from the calibrator (25GB) and

```
/data010/scratch/wwilliams/lds18_pft/data/L232875
```

containing 20 subbands from the target (228GB).

Parsets are available in

```
/home/williams/lds18/parsets
```

STEP 1: setup environment

This tutorial is written in bash (equivalents can be done in tcsh) to be run on CEP3.

make a directory for working in

```
> mkdir -p /data/scratch/<username>/pf_tutorial/
> cd /data/scratch/<username>/pf_tutorial/
```

make a directory for the pipeline to run in

```
> mkdir pipeline
```

make sure you load the required packages

```
> module load lsmtool
> module load lofar
```

prefactor, losoto and rmextract are available on CEP3 but are frequently updated, so we will use a standard set of these packages which are located in /home/williams/lds18/git/.

for prefactor we use a version sourced from the master branch in the git repository (with one or two minor changes). Instead of specifying these directly in the pipeline parsets, we will set some environment variables (Note no trailing slashes)

```
> export PREFACTOR_PATH=/home/williams/lds18/git/prefactor-master/prefactor
> export LOSOTO_PATH=/home/williams/lds18/git/losoto
> export PYTHONPATH=/home/williams/lds18/git/losoto/lib/python2.7/site-
```

```
packages:${PYTHONPATH}
```

rmextract needs to be added to your PYTHONPATH

```
> export PYTHONPATH=/home/williams/lds18/git/RMextract/lib64/python2.7/site-
packages:${PYTHONPATH}
```

STEP 2: setup pipeline

the genericpipeline is part of the lofar installation. After having done

```
> module load lofar
```

you can find where the particular lofar installation you are using is located with the LOFARROOT environment variable:

```
> echo $LOFARROOT
/opt/cep/lofar/lofar_versions/LOFAR-Release-
3_1_2/lofar_build/install/gnucxx11_opt
```

make a copy of the default pipeline config file
> cp $LOFARROOT/share/pipeline/pipeline.cfg .
> cat pipeline.cfg

```
[DEFAULT]
lofarroot = /opt/cep/lofar/lofar_versions/LOFAR-Release-
3_1_2/lofar_build/install/gnucxx11_opt
casaroot = /opt/cep/casacore/builds/casacore-2.3.0/build/gnucxx11_opt
pyraproot = /opt/cep/casacore/builds/python-casacore-2.1.2-2.3.0
hdf5root =
wcsroot =
aoflaggerroot=/opt/cep/aoflagger/aoflagger-2.10.0/build
pythonpath = /opt/cep/lofar/lofar_versions/LOFAR-Release-
3_1_2/lofar_build/install/gnucxx11_opt/lib64/python2.7/site-packages
runtime_directory = %(lofarroot)s/var/run/pipeline
recipe_directories = [%(pythonpath)s/lofarpipe/recipes]
working_directory = /data/scratch/lofarbuild
task_files = [%(lofarroot)s/share/pipeline/tasks.cfg]

[layout]
job_directory = %(runtime_directory)s/%(job_name)s

[cluster]
clusterdesc = %(lofarroot)s/share/cep2.clusterdesc

[deploy]
engine_ppath = %(pythonpath)s:%
(pyraproot)s/lib:/opt/cep/pythonlibs/lib/python/site-packages
engine_lpath = %(lofarroot)s/lib:%(casaroot)s/lib:%(pyraproot)s/lib:%
(hdf5root)s/lib:%(wcsroot)s/lib

[logging]
log_file = %(lofarroot)s/var/log/pipeline-%(job_name)s-%(start_time)s.log
xml_stat_file = %(lofarroot)s/var/log/pipeline-%(job_name)s-%(start_time)s-
statistics.xml

[feedback]
# Method of providing feedback to LOFAR.
# Valid options:
#    messagebus    Send feedback and status using LCS/MessageBus
#    none          Do NOT send feedback and status
```

```
method = messagebus
```

Some of these parameters don't really matter, but there are a few important things to change.

Since we are sharing CEP3 nodes we need to make sure that the pipeline restricts it's use of the cpus available (in addition to settings in the pipeline parsets we will be using). Add the lines:

```
[remote]
method = local
max_per_node = 8
```

The method is local to work on a local/single machine (default). Other methods that can be used when running on other clusters, e.g. pbs_ssh for multinode jobs using the torque system (provided the data are accessible via a shared filesystem).

Change the feedback method to none

```
[feedback]
method = none
```

Add the prefactor recipes directory to the line

```
recipe_directories = [%
(pythonpath)s/lofarpipe/recipes,/home/williams/lds18/git/prefactor-
master/prefactor]
```

this allows the additional pipeline steps defined in the prefactor plugins directory to be used.

Set the working and runtime directories:

```
working_directory = /data/scratch/<username>/pf_tutorial/pipeline
runtime_directory = /data/scratch/<username>/pf_tutorial/pipeline
```

Note that these can be separate directories, if you like, but for simplicity let's keep them the same. The working directory stores things like the pipeline logs (in a logs subdirectory) and the mapfiles (in a mapfiles subdirectory) that are used to tell the pipeline where data is stored.

Also change the logging lines so they don't point to the lofarroot area where you can't write anything...

```
[logging]
log_file = %(runtime_directory)s/%(job_name)s/logs/%(start_time)s/pipeline.log
xml_stat_file = %(runtime_directory)s/%(job_name)s/logs/%
(start_time)s/statistics.xml
```

STEP 3: run calibrator pipeline

Get a copy of the calibrator parset (this is a slightly modified version of the one in the prefactor master branch):

```
> cp /home/williams/lds18/parsets/Pre-Facet-Calibrator-L232873.parset .
```

(Note that there are many interesting advances in the version3 branch for the calibrator parset that are beyond the scope of this tutorial)

Then start the pipeline… and sit back have a coffee or write a paper (realistically though, check that it starts off and doesn't crash and, for now, let's investigate what it is doing)

```
> genericpipeline.py Pre-Facet-Calibrator-L232873.parset -v -d -c pipeline.cfg
```

the -c pipeline.cfg allows you to specify the pipeline configuration file. Note that restarting the pipeline with a different config file will lead to errors.

-v is for verbose output
-d is for debugging output

Run in this way the pipeline will produce a LOT of output in your terminal screen [run in background, screen etc. …]

This will run for some time (~30 min on my test run). So in the meantime, let's have a look at what exactly it is doing.

It will create a directory with same name as the parset you are running in your runtime and working directories, so in this case it will create

```
/data010/scratch/wwilliams/pf_tutorial/pipeline/Pre-Facet-Calibrator-L232873
```

and subdirectories (in the working_dir)

```
parsets
logs
mapfiles
statefile
```

and later will populate dataproducts/ouputs (in the runtime_dir)

It will end with

```
2018-09-17 15:47:12 INFO    genericpipeline: LOFAR Pipeline finished
succesfully.
2018-09-17 15:47:12 INFO    genericpipeline: recipe genericpipeline completed
```

Inspect the results… in results/inspection

```
cal_phases_polXX.png
cal_phases_polYY.png
cal_amplitude_polXX.png
cal_amplitude_polYY.png
losoto_clock.png
losoto_tec.png
losoto_bandpasspolXX.png
losoto_bandpasspolYY.png
losoto_xyoffsetpolYY.png
```

The main final product of this pipeline is the h5parm file containing the calibrator solutions in results/cal_values

```
instrument.h5imp_cal
```

STEP 4: run target pipeline

```
> cp /home/williams/lds18/parsets/Pre-Facet-Target-L2328735.parset .
```

A few changes have been made to the parset in the prefactor master branch to speed up the data processing with some slight reduction in quality of calibration (in Ateam clipping, in the tgss

skymodel used and additional averaging)

Edit the cal_values_directory path to point to your results from the calibrator pipeline

```
! cal_values_directory     =
/data/scratch/<username>/pf_tutorial/pipeline/Pre-Facet-Calibrator-
L232873/results/cal_values
```

And start the pipeline

```
> genericpipeline.py Pre-Facet-Target-L2328735.parset -v -d -c pipeline.cfg
```

This will run for quite some time (~3 hours on my test run).
[monitoring… results… processing times… should take about 3 hrs]

It will end with

```
2018-09-17 15:47:12 INFO    genericpipeline: LOFAR Pipeline finished
succesfully.
2018-09-17 15:47:12 INFO    genericpipeline: recipe genericpipeline completed
```

Inspect the results… in results/inspection

```
L232875_SB120_uv.dppp_124B2FCD4t_144MHz.msdpppconcat_structure.png
L232875_SB120_uv.dppp_124B2FCD4t_146MHz.msdpppconcat_structure.png
L232875_SB120_uv.dppp_124B2FCD4t_144MHz.msdpppconcat_phase.png
L232875_SB120_uv.dppp_124B2FCD4t_146MHz.msdpppconcat_phase.png
```

The main final products of this pipeline are the calibrated target bands… in results

```
L232875_SB120_uv.dppp_124B2FCD4t_144MHz.pre-cal.ms
L232875_SB120_uv.dppp_124B2FCD4t_146MHz.pre-cal.ms
```

STEP 5: make an image

The initial subtract parsets in prefactor can be used to prepare the data for FACTOR, which will make some direction-independent calibrated images and prepare the required residual data. For now though we will simply use wsclean to make an image of our calibrated Target field

```
> module load wsclean
> wsclean -size 4200 4480 -maxuv-l 7000 -baseline-averaging 6.72164158179
-local-rms-method rms-with-min -mgain 0.8 -auto-mask 3.3 -pol I  -padding 1.4
-weighting-rank-filter 3 -auto-threshold 0.5 -j 8 -local-rms-window 50 -mem 20
-weight briggs 0.0 -name /data/scratch/<username>/pf_tutorial/P23-wsclean
-scale 0.00208 -threshold 0.0 -niter 50000 -no-update-model-required -reorder
-local-rms -fit-beam /data/scratch/<username>/pf_tutorial/pipeline/Pre-Facet-
Target-L232875/results/L232875_*.pre-cal.ms
```

This should take about 30 min.

Then open the image in ds9

```
> module load ds9
> ds9 -scale limits -0.01 0.1 /data/scratch/<username>/pf_tutorial/P23-
wsclean-image.fits
```