SAR ARD code summary

Eric L.

Main code files

For *<task>* ∈ {*backsc*, *dualpol*, *intcoh*}:

- Python script to generate jobs: *<task>*_proc_qsub.py
- Bash shell script executed by each job: *<task>*_proc.sh
- GPT Graph file(s) used by the job shell script: *<task>*_proc_graph*N*.xml, *N* = 1, 2, ...

Ancillary code files:

- Python script to generate DEM data (used by job shell script): make_DEM.py
- Shell script to automatically (NCI) or manually (VDI) fix dodgy log files (for *<task>* = *backsc* only!): postproc_out_file.sh
- Shell script to download DEM tiles (shouldn't need to re-use): get_DEM.sh
- Shell script to download orbit files (re-use regularly): get_orbits.sh

Main code files

→ Get latest versions of these files from:
/g/data1a/qd04/Eric/proc

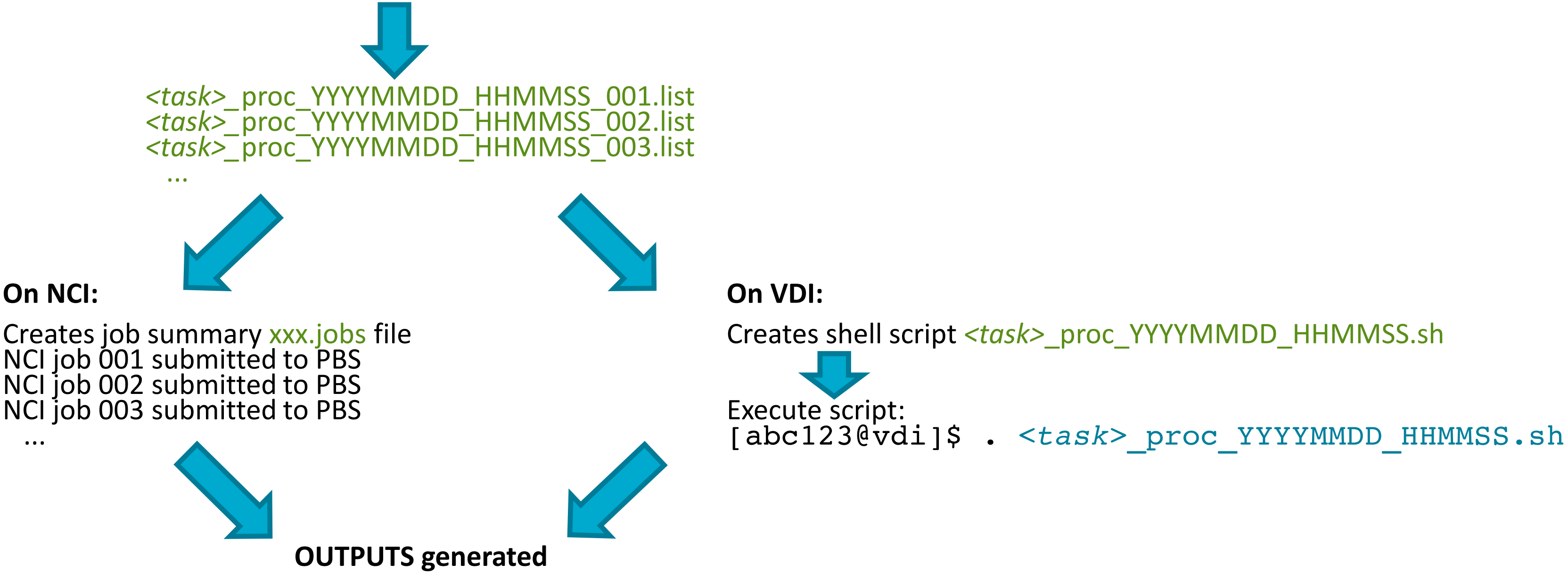
→ Work *still* in progress... (more tweaks possible)

→ Will ultimately archive on GitHub...

General approach

1. Log on to NCI or VDI, create and/or cd to directory, copy **all** files across
2. Then in a terminal, execute:

```
module load python3/3.4.3
module load python3/3.4.3-matplotlib
python3.4 <task>_proc_qsub.py --startdate ... --bbox ... --reprocess_existing ...
```



*<task>*_proc_YYYYMMDD_HHMMSS_001.list
*<task>*_proc_YYYYMMDD_HHMMSS_002.list
*<task>*_proc_YYYYMMDD_HHMMSS_003.list
...

On NCI:

Creates job summary xxx.jobs file
NCI job 001 submitted to PBS
NCI job 002 submitted to PBS
NCI job 003 submitted to PBS
...

On VDI:

Creates shell script *<task>*_proc_YYYYMMDD_HHMMSS.sh

Execute script:

```
[abc123@vdi]$ . <task>_proc_YYYYMMDD_HHMMSS.sh
```

OUTPUTS generated

NCI example

```
python3.4 backsc_proc_qsub.py --reprocess_existing --express_queue  \
            --bbox 146.5 147.5 -38.2 -37.8 --startdate 2018-01-01 --enddate 2018-06-01  \
            --jobs_basename ./log_backsc_test2/ --base_save_dir  \
            /g/data1a/dz56/ga/ga_s1a_c_ard/1-0-0/Copernicus_backsc_TESTING2
```

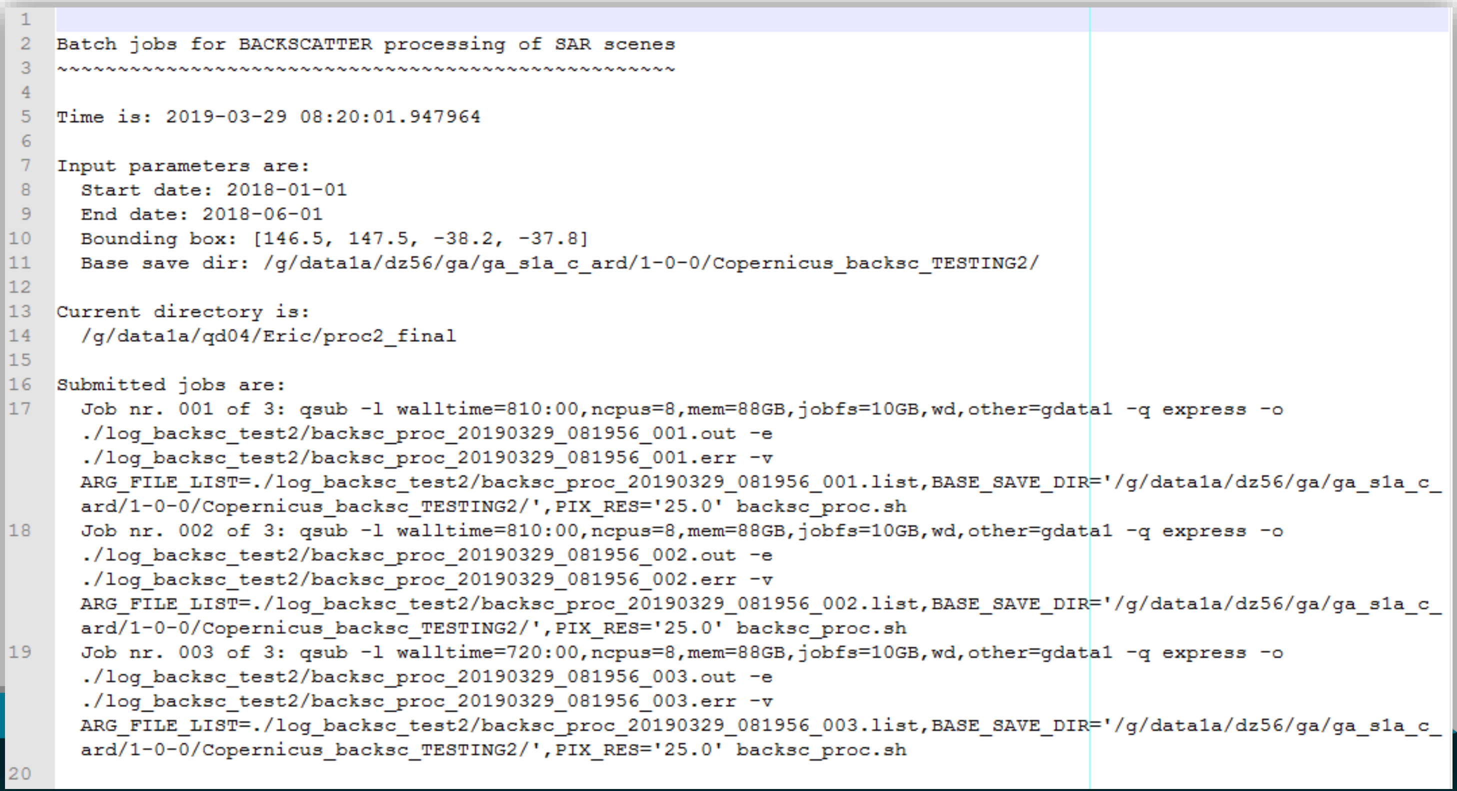
This will (automatically) submit PBS jobs such as:

```
qsub -l ncpus=8,mem=88GB,... -q express ...  \
      -v ARG_FILE_LIST=<task>_proc_yyyymmdd_hhmmss_001.list,  \
      BASE_SAVE_DIR='/g/data1a/dz56/...',PIX_RES='25.0' backsc_proc.sh
```

And creates a summary <task>_proc_YYYYMMDD_HHMMSS.jobs file...

NCI example

Summary <task>_proc_YYYYMMDD_HHMMSS.jobs file:



```

Batch jobs for BACKSCATTER processing of SAR scenes
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Time is: 2019-03-29 08:20:01.947964

Input parameters are:
  Start date: 2018-01-01
  End date: 2018-06-01
  Bounding box: [146.5, 147.5, -38.2, -37.8]
  Base save dir: /g/data1a/dz56/ga/ga_s1a_c_ard/1-0-0/Copernicus_backsc_TESTING2/

Current directory is:
  /g/data1a/qd04/Eric/proc2_final

Submitted jobs are:
  Job nr. 001 of 3: qsub -l walltime=810:00,ncpus=8,mem=88GB,jobfs=10GB,wd,other=gdata1 -q express -o
  ./log_backsc_test2/backsc_proc_20190329_081956_001.out -e
  ./log_backsc_test2/backsc_proc_20190329_081956_001.err -v
  ARG_FILE_LIST=./log_backsc_test2/backsc_proc_20190329_081956_001.list,BASE_SAVE_DIR='/g/data1a/dz56/ga/ga_s1a_c_
  ard/1-0-0/Copernicus_backsc_TESTING2/',PIX_RES='25.0' backsc_proc.sh
  Job nr. 002 of 3: qsub -l walltime=810:00,ncpus=8,mem=88GB,jobfs=10GB,wd,other=gdata1 -q express -o
  ./log_backsc_test2/backsc_proc_20190329_081956_002.out -e
  ./log_backsc_test2/backsc_proc_20190329_081956_002.err -v
  ARG_FILE_LIST=./log_backsc_test2/backsc_proc_20190329_081956_002.list,BASE_SAVE_DIR='/g/data1a/dz56/ga/ga_s1a_c_
  ard/1-0-0/Copernicus_backsc_TESTING2/',PIX_RES='25.0' backsc_proc.sh
  Job nr. 003 of 3: qsub -l walltime=720:00,ncpus=8,mem=88GB,jobfs=10GB,wd,other=gdata1 -q express -o
  ./log_backsc_test2/backsc_proc_20190329_081956_003.out -e
  ./log_backsc_test2/backsc_proc_20190329_081956_003.err -v
  ARG_FILE_LIST=./log_backsc_test2/backsc_proc_20190329_081956_003.list,BASE_SAVE_DIR='/g/data1a/dz56/ga/ga_s1a_c_
  ard/1-0-0/Copernicus_backsc_TESTING2/',PIX_RES='25.0' backsc_proc.sh

```

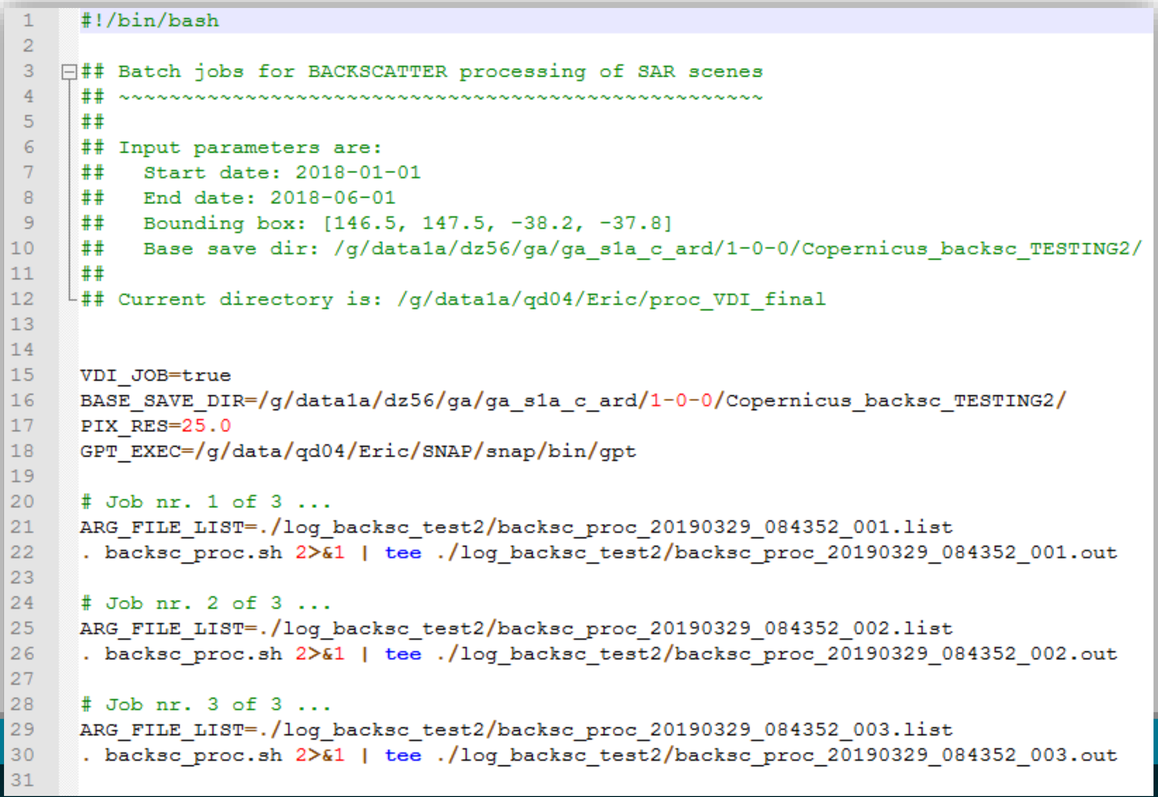
VDI example

```
python3.4 dualpol_proc_qsub.py --reprocess_existing  \
                --bbox 146.5 147.5 -38.2 -37.8  --startdate 2018-01-01  \
                --enddate 2018-06-01  --jobs_basename ./log_dualpol_test2/  \
                --base_save_dir  \
                /g/data1a/dz56/ga/ga_s1a_c_ard/1-0-0/Copernicus_backsc_TESTING2  \
                --VDI_jobs  --gpt_exec /g/data/qd04/Eric/SNAP/snap/bin/gpt
```

This will create the shell script <*task*>_proc_YYYYMMDD_HHMMSS.sh...

VDI example

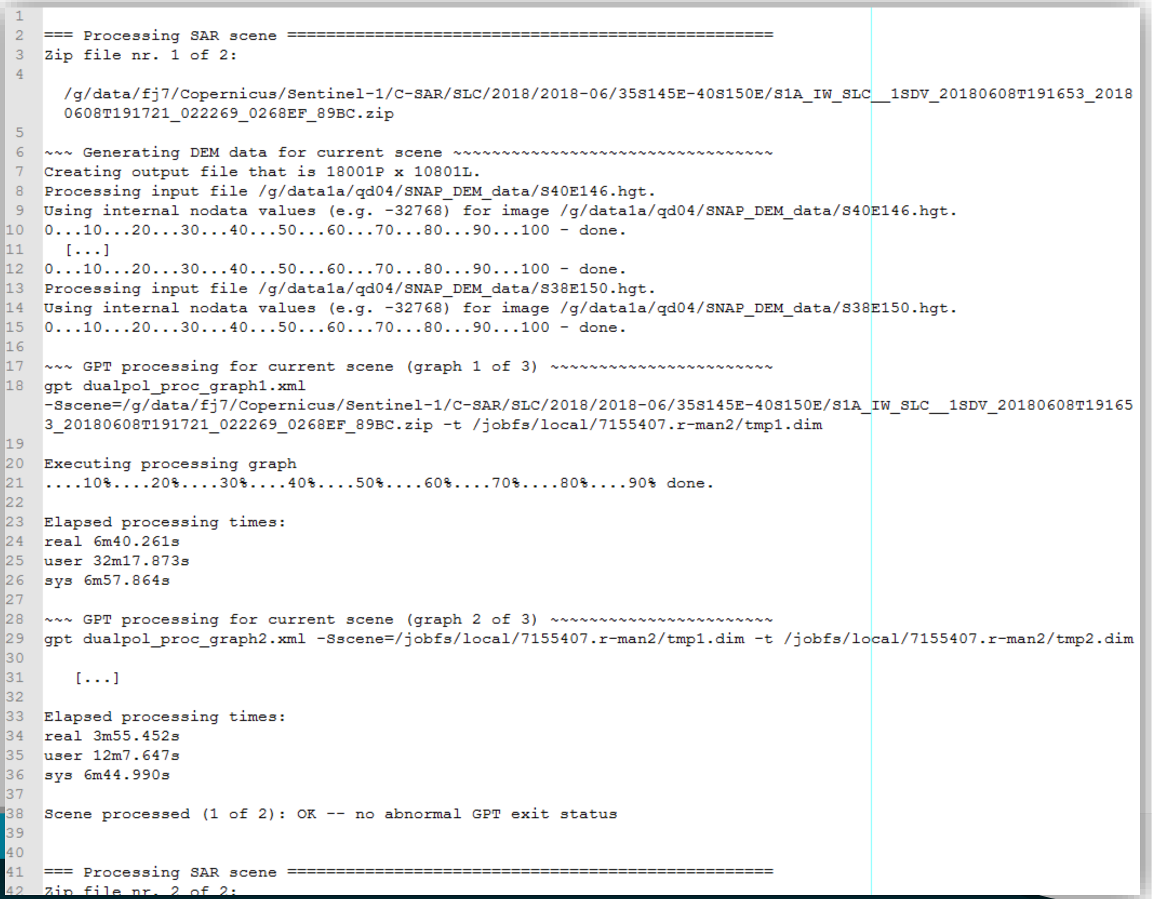
Shell script *<task>*_proc_YYYYMMDD_HHMMSS.sh:



```
#!/bin/bash

## Batch jobs for BACKSCATTER processing of SAR scenes
## ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
##
## Input parameters are:
##   Start date: 2018-01-01
##   End date: 2018-06-01
##   Bounding box: [146.5, 147.5, -38.2, -37.8]
##   Base save dir: /g/data1a/dz56/ga/ga_s1a_c_ard/1-0-0/Copernicus_backsc_TESTING2/
##
## Current directory is: /g/data1a/qd04/Eric/proc_VDI_final


VDI_JOB=true
BASE_SAVE_DIR=/g/data1a/dz56/ga/ga_s1a_c_ard/1-0-0/Copernicus_backsc_TESTING2/
PIX_RES=25.0
GPT_EXEC=/g/data/qd04/Eric/SNAP/snap/bin/gpt

# Job nr. 1 of 3 ...
ARG_FILE_LIST=./log_backsc_test2/backsc_proc_20190329_084352_001.list
. backsc_proc.sh 2>&1 | tee ./log_backsc_test2/backsc_proc_20190329_084352_001.out

# Job nr. 2 of 3 ...
ARG_FILE_LIST=./log_backsc_test2/backsc_proc_20190329_084352_002.list
. backsc_proc.sh 2>&1 | tee ./log_backsc_test2/backsc_proc_20190329_084352_002.out

# Job nr. 3 of 3 ...
ARG_FILE_LIST=./log_backsc_test2/backsc_proc_20190329_084352_003.list
. backsc_proc.sh 2>&1 | tee ./log_backsc_test2/backsc_proc_20190329_084352_003.out

```

Processing outputs

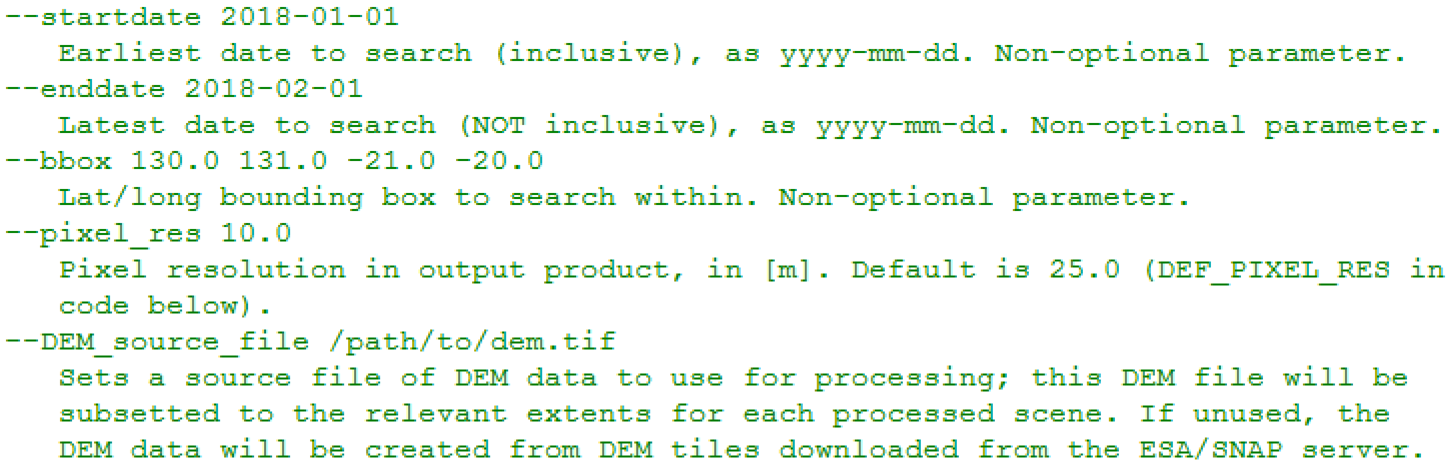
Output file:
<task>_proc_YYYYMMDD_HHMMSS_NNN.out
provides summary information about executed job...



```

=== Processing SAR scene ==================================================
Zip file nr. 1 of 2:

  /g/data/fj7/Copernicus/Sentinel-1/C-SAR/SLC/2018/2018-06/35S145E-40S150E/S1A_IW_SLC__1SDV_20180608T191653_20180608T191721_022269_0268EF_89BC.zip

~~~ Generating DEM data for current scene ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Creating output file that is 18001P x 10801L.
Processing input file /g/data1a/qd04/SNAP_DEM_data/S40E146.hgt.
Using internal nodata values (e.g. -32768) for image /g/data1a/qd04/SNAP_DEM_data/S40E146.hgt.
0...10...20...30...40...50...60...70...80...90...100 - done.
  [...]
0...10...20...30...40...50...60...70...80...90...100 - done.
Processing input file /g/data1a/qd04/SNAP_DEM_data/S38E150.hgt.
Using internal nodata values (e.g. -32768) for image /g/data1a/qd04/SNAP_DEM_data/S38E150.hgt.
0...10...20...30...40...50...60...70...80...90...100 - done.

~~~ GPT processing for current scene (graph 1 of 3) ~~~~~~~~~~~~~~~~~~~~~~~
gpt dualpol_proc_graph1.xml -Sscene=/g/data/fj7/Copernicus/Sentinel-1/C-SAR/SLC/2018/2018-06/35S145E-40S150E/S1A_IW_SLC__1SDV_20180608T191653_20180608T191721_022269_0268EF_89BC.zip -t /jobfs/local/7155407.r-man2/tmp1.dim

Executing processing graph
....10%....20%....30%....40%....50%....60%....70%....80%....90% done.

Elapsed processing times:
real 6m40.261s
user 32m17.873s
sys 6m57.864s

~~~ GPT processing for current scene (graph 2 of 3) ~~~~~~~~~~~~~~~~~~~~~~~
gpt dualpol_proc_graph2.xml -Sscene=/jobfs/local/7155407.r-man2/tmp1.dim -t /jobfs/local/7155407.r-man2/tmp2.dim

   [...]

Elapsed processing times:
real 3m55.452s
user 12m7.647s
sys 6m44.990s

Scene processed (1 of 2): OK -- no abnormal GPT exit status


=== Processing SAR scene ==================================================
Zip file nr. 2 of 2:
```

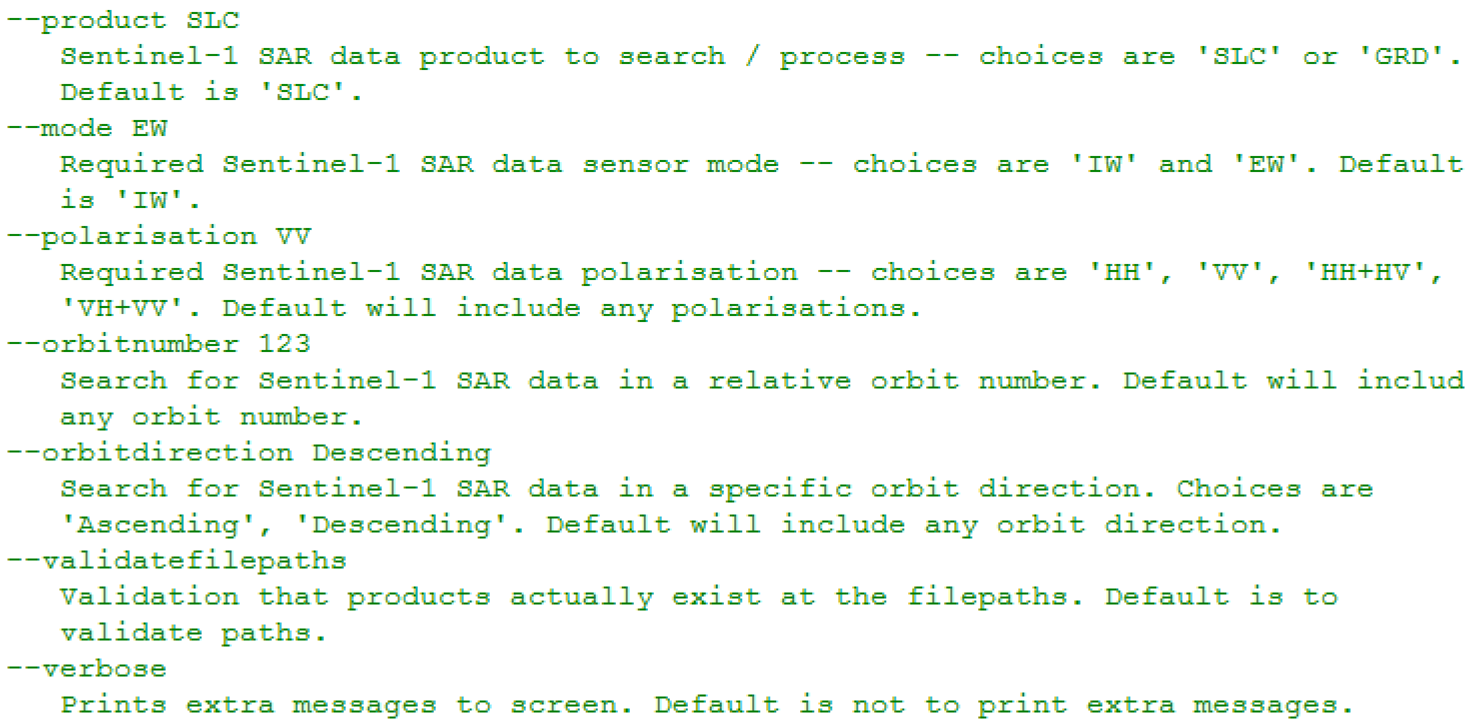
<*task*>_proc_qsub.py: List of parameters

- Documented in .py file itself
- May vary slightly with <*task*> = *intcoh* ...



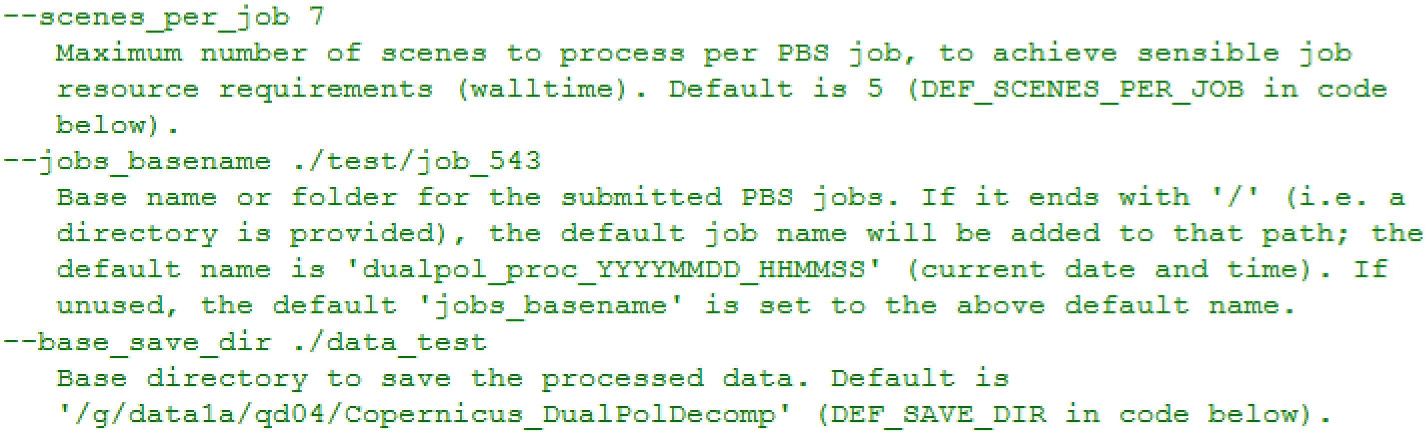
```
--startdate 2018-01-01
   Earliest date to search (inclusive), as yyyy-mm-dd. Non-optional parameter.
--enddate 2018-02-01
   Latest date to search (NOT inclusive), as yyyy-mm-dd. Non-optional parameter.
--bbox 130.0 131.0 -21.0 -20.0
   Lat/long bounding box to search within. Non-optional parameter.
--pixel_res 10.0
   Pixel resolution in output product, in [m]. Default is 25.0 (DEF_PIXEL_RES in
   code below).
--DEM_source_file /path/to/dem.tif
   Sets a source file of DEM data to use for processing; this DEM file will be
   subsetted to the relevant extents for each processed scene. If unused, the
   DEM data will be created from DEM tiles downloaded from the ESA/SNAP server.
```

<*task*>_proc_qsub.py: List of parameters



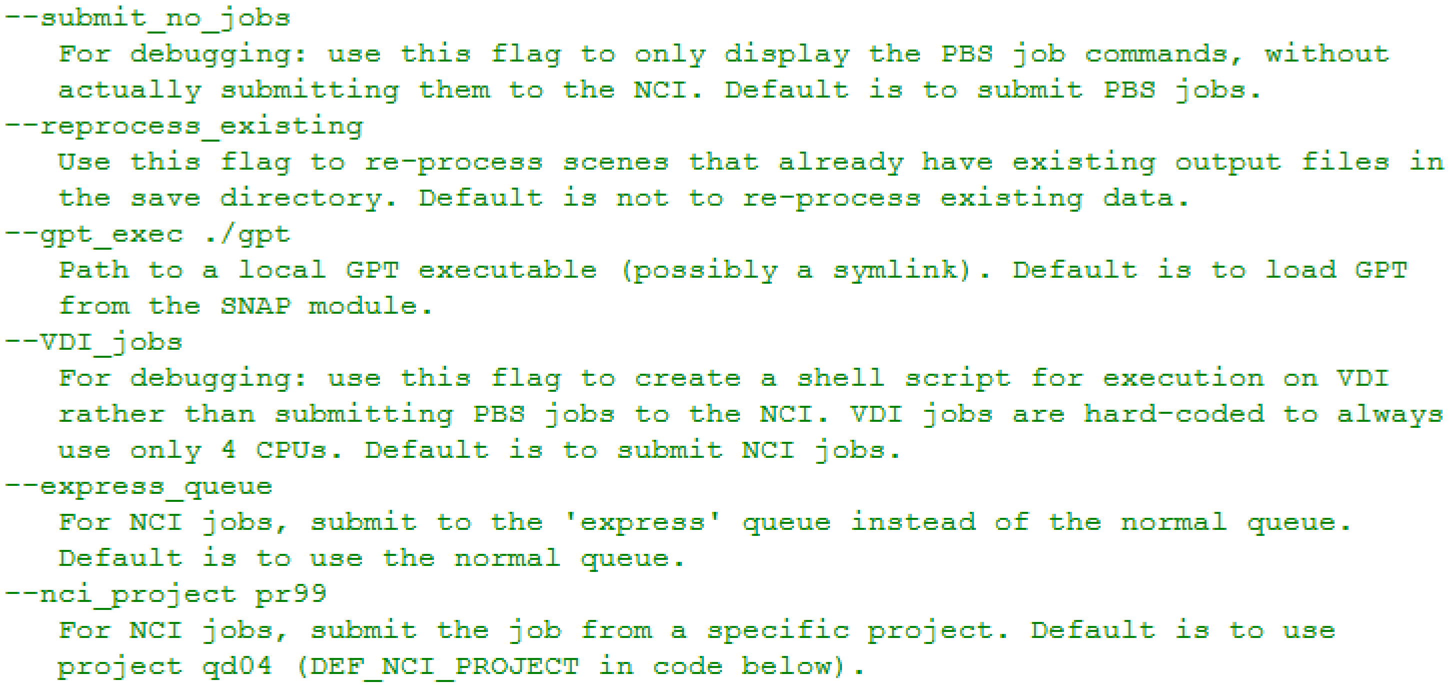
```
--product SLC
   Sentinel-1 SAR data product to search / process -- choices are 'SLC' or 'GRD'.
   Default is 'SLC'.
--mode EW
   Required Sentinel-1 SAR data sensor mode -- choices are 'IW' and 'EW'. Default
   is 'IW'.
--polarisation VV
   Required Sentinel-1 SAR data polarisation -- choices are 'HH', 'VV', 'HH+HV',
   'VH+VV'. Default will include any polarisations.
--orbitnumber 123
   Search for Sentinel-1 SAR data in a relative orbit number. Default will includ
   any orbit number.
--orbitdirection Descending
   Search for Sentinel-1 SAR data in a specific orbit direction. Choices are
   'Ascending', 'Descending'. Default will include any orbit direction.
--validatefilepaths
   Validation that products actually exist at the filepaths. Default is to
   validate paths.
--verbose
   Prints extra messages to screen. Default is not to print extra messages.
```

<*task*>_proc_qsub.py: List of parameters



```
--scenes_per_job 7
   Maximum number of scenes to process per PBS job, to achieve sensible job
   resource requirements (walltime). Default is 5 (DEF_SCENES_PER_JOB in code
   below).
--jobs_basename ./test/job_543
   Base name or folder for the submitted PBS jobs. If it ends with '/' (i.e. a
   directory is provided), the default job name will be added to that path; the
   default name is 'dualpol_proc_YYYYMMDD_HHMMSS' (current date and time). If
   unused, the default 'jobs_basename' is set to the above default name.
--base_save_dir ./data_test
   Base directory to save the processed data. Default is
   '/g/data1a/qd04/Copernicus_DualPolDecomp' (DEF_SAVE_DIR in code below).
```

*<task>*_proc_qsub.py: List of parameters



```
--submit_no_jobs
   For debugging: use this flag to only display the PBS job commands, without
   actually submitting them to the NCI. Default is to submit PBS jobs.
--reprocess_existing
   Use this flag to re-process scenes that already have existing output files in
   the save directory. Default is not to re-process existing data.
--gpt_exec ./gpt
   Path to a local GPT executable (possibly a symlink). Default is to load GPT
   from the SNAP module.
--VDI_jobs
   For debugging: use this flag to create a shell script for execution on VDI
   rather than submitting PBS jobs to the NCI. VDI jobs are hard-coded to always
   use only 4 CPUs. Default is to submit NCI jobs.
--express_queue
   For NCI jobs, submit to the 'express' queue instead of the normal queue.
   Default is to use the normal queue.
--nci_project pr99
   For NCI jobs, submit the job from a specific project. Default is to use
   project qd04 (DEF_NCI_PROJECT in code below).
```

Miscellaneous

- All members of qd04 are able to log on to VDI. Access it as per: https://opus.nci.org.au/display/Help/VDI+User+Guide

SNAP installation

- VDI doesn't have SNAP installed → need manual install
  - download .sh installer from http://step.esa.int/main/download/ (or get it from /g/data/qd04 ... *old version perhaps?*)
  - only 1 or 2 GB in VDI 'home' directory → need to install SNAP in e.g. /g/data1a/qd04/*<user>*/SNAP
  - change SNAP's user dir by editing the file /g/data1a/qd04/*<user>*/SNAP/bin/gpt.vmoptions:
    -Dsnap.userdir=/g/data1a/qd04/*<user>*/SNAP/.snap
  - NCI only: need symlink to orbit files: .../.snap/auxdata/Orbits -> /g/data/qd04/SNAP_Orbits_data
    (on VDI, can also do this, though orbit files can also be automatically downloaded during processing)
- You can also do the same install on the NCI! → Make sure you choose a different install dir. E.g. /g/data1a/qd04/*<user>*/SNAP_NCI vs. /g/data1a/qd04/*<user>*/SNAP_VDI

Miscellaneous

SNAP update

- Manually updating SNAP on VDI: update via the SNAP GUI software...
- Manually updating SNAP on NCI (own local SNAP install):
  - start e.g. Xming locally, log on to Raijin (X11 forwarding enabled)
  - cd to /g/data1a/qd04/*<user>*/SNAP/bin
  - execute: [abc123@raijin]$ snap --nosplash --nogui –modules --update-all
- Updating SNAP module on NCI: need to ask NCI-helpers!...

Miscellaneous

- Currently 120GB in /short, (112GB of) 200GB in /g/data/qd04 – check with ‘nci_account –P qd04’ or ‘lquota’
  → use /short/<*user*> to write big (**test**) datasets (though apparently can’t access /short from VDI...)
  → get in touch with Paul.Ryan@csiro.au to discuss / request more storage and/or compute resources...

- Typical SNAP install:
  - VDI: -Xmx=21G in SNAP; available memory ~32GB ... *beware of other users!*
  - NCI: -Xmx=65G in SNAP (manual install or ‘module load’); available MEM set by job (currently 88GB)
- → there are max. MEM limits ... anything larger required by jobs will fail!
  → e.g. <*task*> = *backsc* appears to require >32GB MEM (at least for some scenes) and will thus consistently fail on the VDI!

- To update orbit files:
  - log on to VDI
  - execute: `[abc123@vdi]$ . get_orbits.sh`

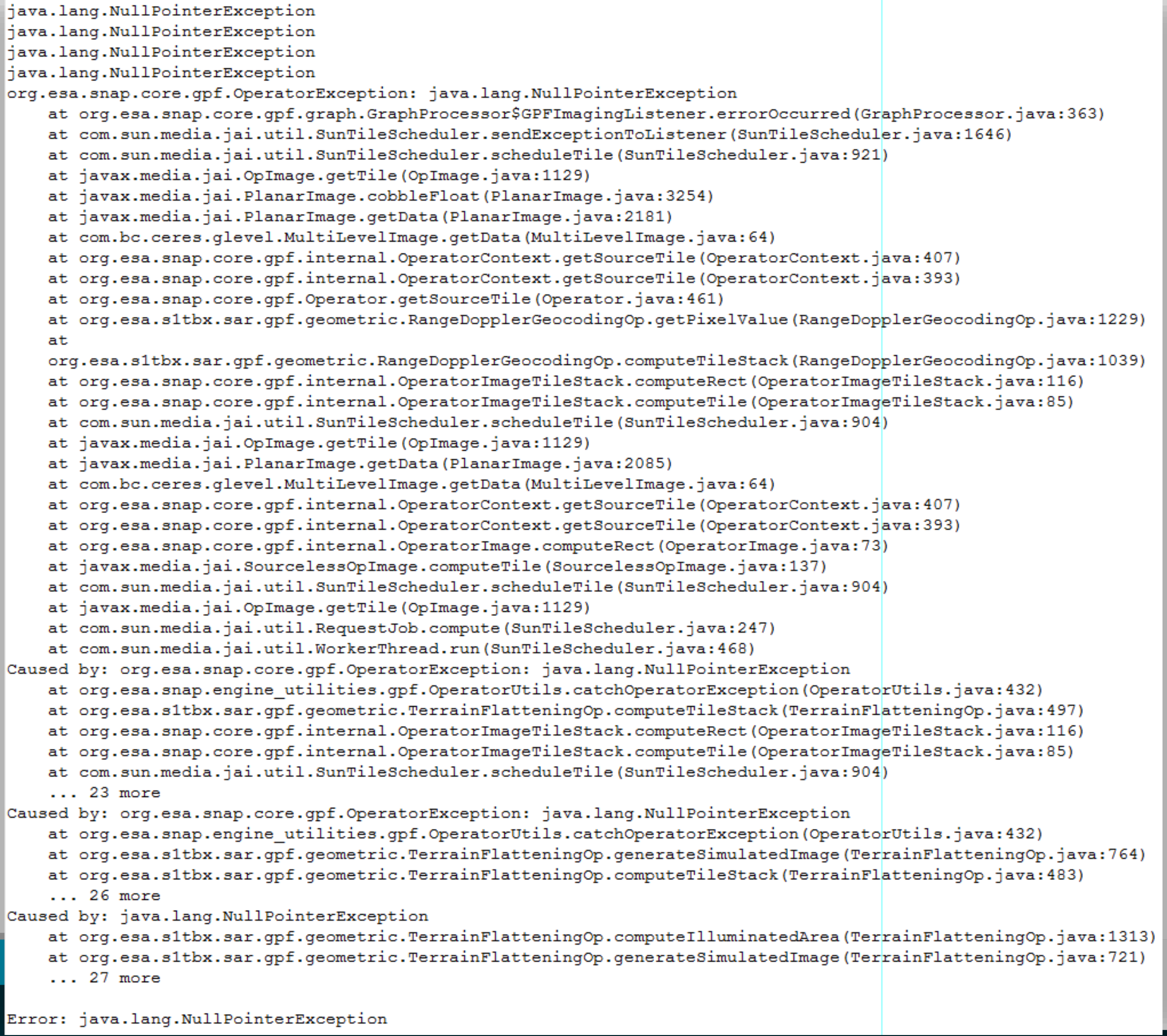
Miscellaneous

You might / will frequently encounter the following messages – I’ve been told to just ignore them!

- *INFO: org.esa.snap.core.gpf.operators.tooladapter.ToolAdapterIO: Initializing external tool adapters*
- *SEVERE: org.esa.s2tbx.dataio.gdal.activator.GDALDistributionInstaller: The environment variable LD_LIBRARY_PATH does not contain the current folder '.'. [...]*
- *INFO: org.hsqldb.persist.Logger: dataFileCache open start*
- *INFO: org.hsqldb.persist.Logger: Database closed* ... even **during** GPT processing!
- […]
  ```
  getGeoPos: Maximum number of iterations reached for pixel (26684.0, 0.0)
  getGeoPos: Maximum number of iterations reached for pixel (26704.0, 0.0)
  x = 26144.0, y = 20.0, lat = -33.56962942107605, lon = 146.9941624206096, iter = 1,
  x’ = 26144.656455993652, y’ = 19.995193234967033
  x = 26184.0, y = 20.0, lat = -33.56857201643739, lon = 146.9899748135287, iter = 1,
  x’ = 26184.444358825684, y’ = 19.99455628985618
  […]
  ```

Miscellaneous

... but can’t ignore *that* one! (your processing will crash – likely points to MEM issues, though feel free to enlighten me!...)



```
java.lang.NullPointerException
java.lang.NullPointerException
java.lang.NullPointerException
java.lang.NullPointerException
org.esa.snap.core.gpf.OperatorException: java.lang.NullPointerException
    at org.esa.snap.core.gpf.graph.GraphProcessor$GPFImagingListener.errorOccurred(GraphProcessor.java:363)
    at com.sun.media.jai.util.SunTileScheduler.sendExceptionToListener(SunTileScheduler.java:1646)
    at com.sun.media.jai.util.SunTileScheduler.scheduleTile(SunTileScheduler.java:921)
    at javax.media.jai.OpImage.getTile(OpImage.java:1129)
    at javax.media.jai.PlanarImage.cobbleFloat(PlanarImage.java:3254)
    at javax.media.jai.PlanarImage.getData(PlanarImage.java:2181)
    at com.bc.ceres.glevel.MultiLevelImage.getData(MultiLevelImage.java:64)
    at org.esa.snap.core.gpf.internal.OperatorContext.getSourceTile(OperatorContext.java:407)
    at org.esa.snap.core.gpf.internal.OperatorContext.getSourceTile(OperatorContext.java:393)
    at org.esa.snap.core.gpf.Operator.getSourceTile(Operator.java:461)
    at org.esa.s1tbx.sar.gpf.geometric.RangeDopplerGeocodingOp.getPixelValue(RangeDopplerGeocodingOp.java:1229)
    at
    org.esa.s1tbx.sar.gpf.geometric.RangeDopplerGeocodingOp.computeTileStack(RangeDopplerGeocodingOp.java:1039)
    at org.esa.snap.core.gpf.internal.OperatorImageTileStack.computeRect(OperatorImageTileStack.java:116)
    at org.esa.snap.core.gpf.internal.OperatorImageTileStack.computeTile(OperatorImageTileStack.java:85)
    at com.sun.media.jai.util.SunTileScheduler.scheduleTile(SunTileScheduler.java:904)
    at javax.media.jai.OpImage.getTile(OpImage.java:1129)
    at javax.media.jai.PlanarImage.getData(PlanarImage.java:2085)
    at com.bc.ceres.glevel.MultiLevelImage.getData(MultiLevelImage.java:64)
    at org.esa.snap.core.gpf.internal.OperatorContext.getSourceTile(OperatorContext.java:407)
    at org.esa.snap.core.gpf.internal.OperatorContext.getSourceTile(OperatorContext.java:393)
    at org.esa.snap.core.gpf.internal.OperatorImage.computeRect(OperatorImage.java:73)
    at javax.media.jai.SourcelessOpImage.computeTile(SourcelessOpImage.java:137)
    at com.sun.media.jai.util.SunTileScheduler.scheduleTile(SunTileScheduler.java:904)
    at javax.media.jai.OpImage.getTile(OpImage.java:1129)
    at com.sun.media.jai.util.RequestJob.compute(SunTileScheduler.java:247)
    at com.sun.media.jai.util.WorkerThread.run(SunTileScheduler.java:468)
Caused by: org.esa.snap.core.gpf.OperatorException: java.lang.NullPointerException
    at org.esa.snap.engine_utilities.gpf.OperatorUtils.catchOperatorException(OperatorUtils.java:432)
    at org.esa.s1tbx.sar.gpf.geometric.TerrainFlatteningOp.computeTileStack(TerrainFlatteningOp.java:497)
    at org.esa.snap.core.gpf.internal.OperatorImageTileStack.computeRect(OperatorImageTileStack.java:116)
    at org.esa.snap.core.gpf.internal.OperatorImageTileStack.computeTile(OperatorImageTileStack.java:85)
    at com.sun.media.jai.util.SunTileScheduler.scheduleTile(SunTileScheduler.java:904)
    ... 23 more
Caused by: org.esa.snap.core.gpf.OperatorException: java.lang.NullPointerException
    at org.esa.snap.engine_utilities.gpf.OperatorUtils.catchOperatorException(OperatorUtils.java:432)
    at org.esa.s1tbx.sar.gpf.geometric.TerrainFlatteningOp.generateSimulatedImage(TerrainFlatteningOp.java:764)
    at org.esa.s1tbx.sar.gpf.geometric.TerrainFlatteningOp.computeTileStack(TerrainFlatteningOp.java:483)
    ... 26 more
Caused by: java.lang.NullPointerException
    at org.esa.s1tbx.sar.gpf.geometric.TerrainFlatteningOp.computeIlluminatedArea(TerrainFlatteningOp.java:1313)
    at org.esa.s1tbx.sar.gpf.geometric.TerrainFlatteningOp.generateSimulatedImage(TerrainFlatteningOp.java:721)
    ... 27 more

Error: java.lang.NullPointerException
```