Phase Picking Workflow

Lachlan Adams

February 4, 2022

Contents

Introduction

This brief document describes the parallel arrival picking workflow, the interaction of the various subsystems in HiPerSeis and how an ensemble set of P and S arrivals are aggregated for generating body-wave tomography products.

While some of the steps described in this document can be carried out on their own, experience suggests that in order to interact with SeisComp3 in a sustainable manner, it is advisable to run the workflow from start-to-end. It is critical that any steps from section 6 onwards are not repeated without performing the cleanup described in section 5.

1 Retrieve Data

The catalogues used come from a variety of sources. These are:

- Gary Gibson hypocentre catalogue - this is a catalogue in comma separated values (.csv) format containing hypocentres of seismic events in Australia from 1840 until the early 2000s.
- Bob Engdahl catalogue - these are extensible markup language (.xml) files in the SC3ML format containing information about events in the Australian region from the 1990s to late 2000s.
- ISC catalogue - these are .xml files in the QuakeML format which are downloaded from the International Seismological Centre web services. We only use the reviewed ISC catalogue.
- GA catalogue - these are .xml files in the QuakeML format which contain events which have been reviewed by analysts from Geoscience Australia.
- USGS catalogue - these are .xml files in the QuakeML format downloaded from a ftp server provided by the United States Geological Survey. This server contains both reviewed and un-reviewed events.

These catalogues are merged and filtered using the following criteria:

- Events in the GA database have their hypocentre replaced with those in the Gary Gibson files, if a match is found.
- Pairs of events where the hypocentre differs by less than 50km and the origin times are within 15 seconds of one another are considered to be duplicates. In this case, preferred hypocentres are Bob Engdahl $>$ ISC $>$ GA $>$ USGS.
- ISC and USGS events in the Australian bounding box ($\phi \in [105, 160]$ and $\theta \in [-40, -13]$ with $\phi, \theta =$ longitude and latitude) are filtered so that only events with an azimuthal gap of less than 180 degrees are retained.
- ISC and USGS events in the Indian Ocean ($\phi \in [15, 90]$ and $\theta \in [-60, 30]$, or $\phi \in [90, 180]$ and $\theta \in [-60, -45]$) are retained only if their magnitude is larger than 5.0.
- ISC and USGS events outside these regions are kept if their magnitude is above 5.5.

- GA events are filtered to have an azimuthal gap of less than 180°.

The events from the Geoscience Australia catalogue may be downloaded on the NCI using the following console command.

```
python hiperseis/seismic/catalogue/download_catalogues/download_GA/
↪  downloadquakeml.py --agency <agency name> --start_time
↪  <YYYY-MM-DDThh:mm:ss> --end_time <YYYY-MM-DDThh:mm:ss>
↪  --min_magnitude <min_mag> --max_magnitude <max_mag> --bounding_box
↪  <[min_lat, min_lon, max_lat, max_lon]> --output <.../output_path/>
```

The events from the ISC web services may be downloaded onto the NCI using the following console command. The 'split_by_day' command is a flag telling the algorithm to save one .xml file for each day instead of each month (default). Use this flag if there are expected to be a large amount of events over the time period specified.

```
python hiperseis/seismic/catalogue/download_catalogues/
↪  download_ISC/download_isc.py --output_path <.../output_path>
↪  --start_date <YYYY-MM-DD> --start_time <hh:mm:ss> --end_date
↪  <YYYY-MM-DD> --end_time <hh:mm:ss> --bbox <min_lon min_lat max_lon
↪  max_lat> --mag_range <min_mag max_mag> --split_by_day <True>
```

Events included in the USGS database may be downloaded using the following console command. These must be preprocessed in the next step separately to those in the other catalogues.

```
python hiperseis/seismic/catalogue/download_catalogues/
↪  download_USGS/download_usgs_catalogue.py --output_path
↪  <.../output_path> --start_date <YYYY-MM-DD> --end_date <YYYY-MM-DD>
```

2 Merge Catalogues

Merging and filtering of the catalogues can be performed on a single processor, however it is recommended that some parts of the process are performed on multiple processors. If there is a large amount of event .xml files, then the time taken to merge the catalogues may be reduced by first producing binary files which contain pickled versions of the catalogues using the pbs script in section 2.1.

2.1 Preprocessing Input Data

If desired, preprocessing of input .xml files should be performed with the following pbs script. Note that hiperseis is contained in /g/data/ha3/la8536/hiperseis/, and /g/data/ha3/la8536/VENV/para_h5py/lib/python3.7/site-packages/ contains some required python libraries. The .xml files are in /g/data/ha3/la8536/Catalogue/, and the flag –split_smaller is telling the algorithm to produce one binary file per input .xml file (otherwise one is produced per processor).

If preprocessing the USGS catgalogue, as no filters can be applied in the download step (apart from date filters), Read_USGS_XML_Multiprocessing.py can be used to apply the filters described in section 1. Arguments are the same arguments as the script included below.

```
1 #!/bin/bash
2
3 # PBS directives
4 #PBS -P vy72
5 #PBS -N make_event_binaries
6 #PBS -q normal
```

```
#PBS -l walltime=24:00:00
#PBS -l mem=100GB
#PBS -l ncpus=240
#PBS -l jobfs=100GB
#PBS -l storage=scratch/la8536+gdata/ha3
#PBS -l wd
#PBS -j oe
#PBS -M lachlan.adams@ga.gov.au
#PBS -m bae


export PATH=/g/data/ha3/la8536/.local/bin:$PATH
export PYTHONPATH=/g/data/ha3/la8536/hiperseis/:
↪ /g/data/ha3/la8536/VENV/para_h5py/lib/python3.7/site-packages/:
↪ $PYTHONPATH
export LC_ALL=en_AU.UTF-8
export LANG=en_AU.UTF-8

module purge

module load pbs
module load python3/3.7.4
module load hdf5/1.10.5p
module load openmpi/4.0.1

module list

mpirun -np $PBS_NCPUS python3
↪ /g/data/ha3/la8536/hiperseis/seismic/catalogue/merge_catalogues/
↪ Read_XML_Multiprocessing.py --paths /g/data/ha3/la8536/Catalogue/
↪ --output_path /g/data/ha3/la8536/Catalogue/event_binary_files/
↪ --split_smaller True
```

2.2 Filtering Catalogue

On a single processor, the script Filter_Catalogues.py can be run in the command line to filter all input catalogues. The algorithm can either read the output of Read_XML_Multiprocessing.py to save time, or can read the event .xml files directly (if the dataset is small).

Criteria currently used for filtering are described in section 1. If using the preprocessed pickled catalogues (.pkl files), set the flag –read_from_preprocessed to True and leave –GA_path (GA catalogue), –ISC_path (ISC catalogue), –USGS_path (USGS catalogue)), and –EHB_path (Engdahl catalogue) blank. Otherwise leave –preprocessed_path blank. –GG_path is the directory containing the Gary Gibson catalogue.

```
python3 hiperseis/seismic/catalogue/merge_catalogues/
↪ Filter_Catalogues.py --preprocessed_path <.../path/> --GG_path
↪ <.../path/> --output_path <.../path/> --read_from_preprocessed True
```

Or

```
python3 hiperseis/seismic/catalogue/merge_catalogues/
↪ Filter_Catalogues.py --GA_path <.../path/> --ISC-path <.../path/>
↪ --EHB_path <.../path/> --GG_path <.../path/> --USGS_path <.../path/>
↪ --output_path <.../path/>
```

2.3 Converting Catalogue to CSV

The conversion from the Obspy earthquake catalogue format to a format usable by the software for later parts of this workflow is a computationally expensive operation. For this reason it should be run on multiple processors.

The Convert_Catalogues_To_CSV.py script reads the output from section 2.2 on multiple processors, converts each event into a list format containing all required information, and saves each event as a separate .csv file. This is because the time taken to collate the output of each processor into a single object within the multiprocessing environment is larger than the time taken to save the output to disk and read it again using the script in section 2.4. In this step, station coordinates are matched with each pick, if an inventory of stations is available. In this case, –station_path points to the directory in which the FDSN station .xml inventory and ISC International Registry (IR) .kml inventory are stored. To perform this step of the workflow, run the following PBS script.

```
#!/bin/bash

# PBS directives
#PBS -P vy72
#PBS -N make_csv
#PBS -q hugemem
#PBS -l walltime=10:00:00
#PBS -l mem=500GB
#PBS -l ncpus=48
#PBS -l jobfs=200GB
#PBS -l storage=scratch/la8536+gdata/ha3
#PBS -l wd
#PBS -j oe
#PBS -M lachlan.adams@ga.gov.au
#PBS -m bae


# this is tested working in NCI gadi system.
export PATH=/g/data/ha3/la8536/.local/bin:$PATH
export PYTHONPATH=/g/data/ha3/la8536/hiperseis/:
↪ /g/data/ha3/la8536/VENV/para_h5py/lib/python3.7/site-packages/:
↪ $PYTHONPATH
export LC_ALL=en_AU.UTF-8
export LANG=en_AU.UTF-8

module purge

module load pbs
module load python3/3.7.4
module load hdf5/1.10.5p
module load openmpi/4.0.1

module list

mpirun -np $PBS_NCPUS python3 /g/data/ha3/la8536/Catalogue/
↪ merge_catalogues/Convert_Catalogues_To_CSV.py --preprocessed_path
↪ <.../path/> --station_path <.../path/> --output_path <.../path/>
```

2.4 Merging Catalogues

On a single processor, the following script reads in all output files from section 2.3, and merges them into a single file after sorting the events by origin time. Perform this step by running the following console command.

```
python3 /g/data/ha3/la8536/Catalogue/merge_catalogues/
Merge_Catalogues.py --input_path <.../path> --output_file
<.../path/merge_catalogues_output.csv>
```

The output will be a .csv file with the following structure. Every pick row between event rows is considered as belonging to the event above it.

```
event_rows = [#source, YYYY, MM, DD, hh, mm, ss, lon, lat, depth,
↪  n_phase, mb, ms, ml, mw, event_id, azim_gap, index]
pick_rows = [sta, cha, loc, net, lon, lat, elev, phase, YYYY, MM, DD, hh,
↪  mm, ss, ang_dist]
```

3 Parallel Picker for Automatic Arrival Detection

To increase the amount of data available, arrivals from Adaptable Seismic Data Format (ASDF) files are scoured to attempt to match these arrivals with events in the merged catalogue. This can increase the ray path coverage of the region we wish to analyse. A list of available ASDF files is contained on the NCI in /g/data/ha3/Passive/SHARED_DATA/Index/asdf_files.txt.

The output of the algorithms in section 2.4 or 2.3 is read by the script /hiperseis/seismic/pick_harvester/pick.py and the picks in the files listed within asdf_files.txt are paired with these events, if a match is found. The script outputs two files, p_combined.txt and s_combined.txt, which contain P and S arrivals detected on all stations, for all events in the CSV events catalog.

To perform this step of the workflow, run the following pbs script. Arguments are (1) file containing asdf file names, (2) output path from section 2, (3) output path.

```
#!/bin/bash

# PBS directives
#PBS -P vy72
#PBS -N pick_pbs_job
#PBS -q hugemem
#PBS -l walltime=24:00:00,mem=1470GB,ncpus=96,jobfs=1000GB
#PBS -l storage=scratch/la8536+gdata/ha3
#PBS -l wd
#PBS -j oe
#PBS -M lachlan.adams@ga.gov.au
#PBS -m bae


export PATH=/g/data/ha3/la8536/.local/bin:$PATH
export PYTHONPATH=/g/data/ha3/la8536/hiperseis/:
↪  /g/data/ha3/la8536/VENV/para_h5py/lib/python3.7/site-packages/:
↪  $PYTHONPATH
export LC_ALL=en_AU.UTF-8
export LANG=en_AU.UTF-8

module purge

```

```
module load pbs
module load python3/3.7.4
module load hdf5/1.10.5p
module load openmpi/4.0.1

module list

mpirun -np $PBS_NCPUS python3
↪ /g/data/ha3/la8536/hiperseis/seismic/pick_harvester/pick.py
↪ /g/data/ha3/Passive/SHARED_DATA/Index/asdf_files.txt
↪ /g/data/ha3/la8536/Catalogue/Outputs/
↪ /g/data/ha3/la8536/Picking/Step1 >
↪ /g/data/ha3/la8536/Picking/Step1/run_output.txt
```

The output will be two .txt files with the following structure. One file contains 'P' picks, while the other contains 'S' picks.

```
pick_rows = [event_id, origin_time, mag, elon, elat, edepth (km), net,
↪ sta, cha, arrival_time, slon, slat, az, baz, dist, residual, snr,
↪ qualityMeasureCWT, domFreq, qualityMeasureSlope, bandIndex, nsigma]
```

4 Creating Ensemble .xml Files

For usage, the .csv event catalogue must be converted into .xml catalogues in the SC3ML format. The algorithms in section 2 will reassign network codes to the picks used to match those contained in the station catalogue files used (if commanded to). These still need to be added to the extra picks found in section 3. In addition, some stations may not be contained in the station catalogue; in particular ISC events have all networks set to IR (international registry). To find any network codes for stations not included in the station catalogue, the ISC station coordinate database is also searched to check if the station is included there.

To redefine the network codes in the new picks added in section 3 and to produce the required .xml files, the following pbs script should be run. Inputs are (1) output path from section 2, (2) station file, (3) ISC station coordinates file, (4) output path, (5) p wave arrival file output from section 3, (6) p wave arrival file output from section 3.

```
#!/bin/bash

# PBS directives
#PBS -P vy72
#PBS -N create_ensemble_pbs_job
#PBS -q hugemem
#PBS -l walltime=01:00:00,mem=2000GB,ncpus=192,jobfs=200GB
#PBS -l storage=scratch/la8536+gdata/ha3
#PBS -l wd
#PBS -j oe
#PBS -M lachlan.adams@ga.gov.au
#PBS -m bae


export PATH=/g/data/ha3/la8536/.local/bin:$PATH
export PYTHONPATH=/g/data/ha3/la8536/hiperseis/:
/g/data/ha3/la8536/VENV/para_h5py/lib/python3.7/site-packages/:
$PYTHONPATH
```

```
export LC_ALL=en_AU.UTF-8
export LANG=en_AU.UTF-8

module purge

module load pbs
module load python3/3.7.4
module load hdf5/1.10.5p
module load openmpi/4.0.1

module list

mpirun -np $PBS_NCPUS python3 /g/data/ha3/la8536/hiperseis/
seismic/pick_harvester/createEnsembleXML.py
/g/data/ha3/la8536/Catalogue/Outputs/
/g/data/ha3/la8536/Catalogue/stations.xml
/g/data/ha3/Passive/SHARED_DATA/Inventory/station_coords/
stations.kml /g/data/ha3/la8536/Step4/
--p-arrivals /g/data/ha3/la8536/Step3/p_combined.txt
--s-arrivals /g/data/ha3/la8536/Step3/s_combined.txt
```

Once the run completes, the output folder should contain the following.

- Event .xml files which now contain both the arrivals from the original input catalogue and the automatically detected arrivals, named according to MPI ranks used.
- Log files containing stations that were not found and the number of arrivals associated with them. Note that we expect to have some missing stations.
- A text file, ensemble.txt, which is a flat representation of all the arrivals in the output .xml files. This file is useful for generating various diagnostic plots. Further details on this are presented in the latter sections.

5 Preparing SeisComp3 for iLoc Processing

In order to use iLoc, SeisComp3 is used for data management. The steps for preparing a SeisComp3 instance for ingesting XML events are: (a) Delete all existing inventories, (b) Adapt and ingest all available inventories, (c) Create bindings for inventories, (d) Delete all preexisting events and arrivals.

5.1 Resetting Database

If starting from scratch, or if the database needs to be reset from scratch, the following must be performed. If the database already exists, then this step is unnecessary.

5.1.1 Drop the Database

Run mysql in the command line as user root, using password seiscomp3. Then, perform the following.

```
mysql > DROP DATABASE seiscomp3;
mysql > CREATE DATABASE seiscomp3 CHARACTER SET utf8 COLLATE
utf8_bin;
mysql > grant usage on seiscomp3.* to sysop@localhost identified
by ‘sysop’
```

```
mysql > grant all privileges on seiscomp3.* to sysop@localhost;
mysql > grant usage on seiscomp3.* to sysop@'%' identified by
'sysop';
mysql > grant all privileges on seiscomp3.* to sysop@'%';
mysql > flush privileges
```

Exit the database.

5.1.2 Setup New Database

Run the following script (contained within the seiscomp3 distribution) in the command line to add all tables and information to the database needed to run seiscomp3.

```
$ mysql < /opt/seiscomp3/share/db/mysql.sql
```

5.1.3 Setup and Configure SeisComp3

Run the following in the command line.

```
$ /opt/seiscomp3/bin/seiscomp stop
$ /opt/seiscomp3/bin/seiscomp setup
```

Enter the default values until it asks you "Create database [yes]:". Select no and continue with the default values until the completion of setup.

In the command line, run the following.

```
$ /opt/seiscomp3/bin/seiscomp enable seedlink scautopick scautoloc
scamp scmag scevent
```

Start with an empty inventory in /opt/seiscomp3/etc/inventory/, and then create an empty file "inventory_empty.xml" in this folder. Import a dataless seed volume into this file, then update the configuration by running the following in the command line.

```
$ /opt/seiscomp3/bin/seiscomp exec import_inv dlsv
inventory_empty.xml
$ /opt/seiscomp3/bin/seiscomp update-config
```

5.1.4 Start SeisComp3

Finally, restart SeisComp3 by running the following in the command line.

```
$ /opt/seiscomp3/bin/seiscomp start
```

5.2 Delete All Existing Inventories

To start off with a clean slate, log into the instance and remove all preexisting inventories from /opt/seiscomp3/etc/inventory. Note that this step is not necessary for subsequent iterations of processing, if the existing inventories have not changed or when some new inventories need to be added. When new inventories need to be added to the existing pool, simply copy them into this folder and follow the steps below.

In the command line, run the following.

```
$ scinv sync
$ scinv sync keys
$ scinv sync
$ scinv sync keys
```

The repetition of both commands is needed due to a SeisComp3 oddity. This will remove all keys from the database corresponding to stations no longer included in it.

5.3 Adapt and Ingest All Available Inventories

There are two sources of inventories:

- The folder /g/data/ha3/Passive/SHARED_DATA/Inventory/networks_sc3ml/ on the NCI contains SC3ML inventories. These files are copied to the folder /opt/seiscomp3/etc/inventory on the AWS instance.
- FDSNStationXML inventories for temporal deployments (provided by Jason) which are generally found in the folders /g/data/ha3/Passive/_ANU/ and /g/data/ha3/Passive/_AusArray/ on the NCI.

FDSNStationXML inventories in the above folders do not contain valid response objects. After attaching 'bogus' response objects to each channel we convert each inventory into SC3ML format as follows.

- Make a new folder on the AWS. Change directory into this folder and create two sub-folders named 'input' and 'output'. Copy all the FDSNStationXML inventories from the above folders on the NCI to the 'input' folder.
- From the command line, run the following.

  ```
  $ export PYTHONPATH=\$PYTHONPATH:/home/centos/hiperseis/
  ```

- Read all FDSNStationXML inventories in the input folder, insert valid response objects and produce the required SC3ML inventories in the output folder above. This is performed by running the following in the command line.

  ```
  $ python
  /home/centos/hiperseis/seismic/inventory/fdsnxml_convert.py
  input/ output/ --response-fdsnxml /home/centos/hiperseis/
  seismic/inventory/responses/LE3dLite.xml
  ```

  These files are then copied to the folder /opt/seiscomp3/etc/inventory on the AWS instance.

Run the following from the command line to synchronise the new inventory.

```
$ scinv sync
```

With the completion of this step, one is now able to directly download FDSNStationXMLs from the SeisComp3 server through a webservice query as follows.

```
$ wget http://ip_address:port/fdsnws/station/1/query -O
fdsn_inventory.xml
```

The IP address will change each time an AWS instance is restarted. Also note that SeisComp3 runs the fdsnws webservice on port 8081, configurable through /opt/seiscomp3/etc/fdsnws.cfg.

5.4 Create Bindings for Inventories

For each station ingested into the SeisComp3 inventory, a "binding" must be created. See https://www.seiscomp.de/seiscomp3/doc/seattle/2013.046/apps/global.html for more details. To do this, perform the following procedure.

1. Enter the command "scconfig" into the command line. This will open up the configuration window GUI, so X-forwarding must be enabled to perform this step.
2. On the left side of the GUI, click on bindings.

3. On the right side of the GUI, enter the drop down list for the "global" folder, and find the binding "__SHZ".
4. Drag and drop this onto the "Networks" folder on the left side of the screen. See figure 1 for a guide.
5. Exit the GUI. Click "yes" when asked if you wish to save the updated configuration.

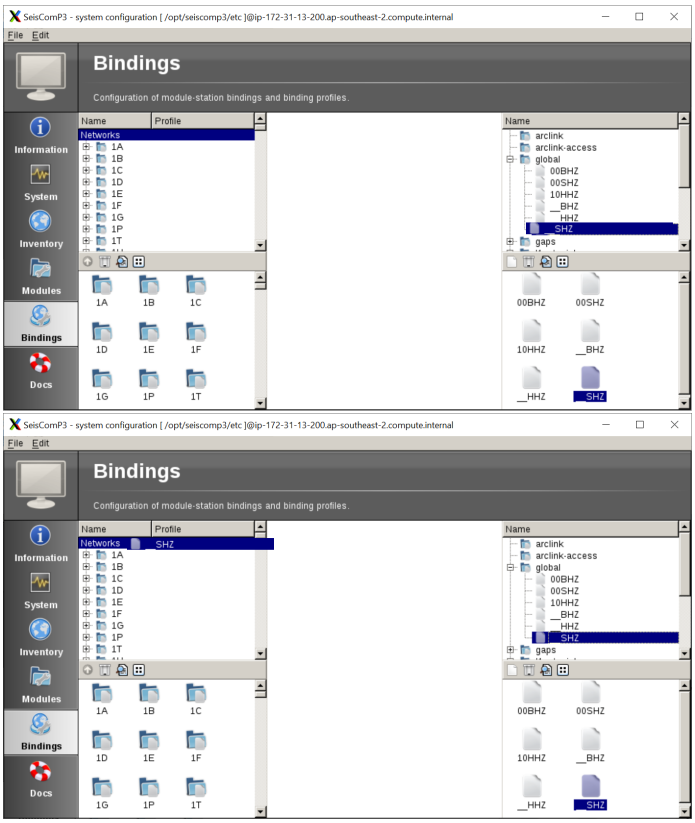

Figure 1: Drag and drop the "SHZ" binding onto the "Networks" folder.

5.5 Delete all Pre-Existing Events and Arrivals

On the AWS instance, delete all preexisting events and arrivals as follows.

```
$ scdbstrip --days 1 -d mysql://sysop:sysop@localhost/seiscomp3
```

It might take a while (hours). Running the command in a screen session will allow the user to disconnect from the AWS during execution of the command. In case the above fails (another SeisComp3 oddity), manually cleanse the database of all preexisting arrivals as follows.

```
$ mysql -e"delete Object from Object, Pick where Object._oid=
Pick._oid"
```

6 Ingest .xml Event Files Into SeisComp3

We now ingest the XML files (output of section 4) on the AWS instance as follows.

- We now copy all the XML files (output in Step 2) to the AWS instance, in the folder /home/centos/picking_workflow/step_4/. While copying files to AWS, gadi-dm.nci.org.au (instead of gadi) can support 10× network transfer rates.
- Change directory to the above folder on the AWS instance and run the following from the command line.

```
for axml in `ls *.xml`; do sed -i 's%<seiscomp
xmlns="http://geofon.gfz-potsdam.de/ns/seiscomp3-schema/0.10"
version="0.10">%<seiscomp
xmlns="http://geofon.gfz-potsdam.de/ns/seiscomp3-schema/0.9"
version="0.9">%g' $axml; done

for i in `ls *.xml`; do scdb -i $i -d
mysql://sysop:sysop@localhost/seiscomp3 -b 1000; done
```

  This may take a while (several hours). Running the above in a screen session will allow a user to disconnect from the AWS during execution of the command.

7 Run iLoc to Identify/Redefine Phases

iLoc is an earthquake phase identification/redefinition code that updates each event origin location by iteratively optimizing some misfit function that factors in a number of variables including travel-time residuals, etc. iLoc works off of parametric data from a SeisComp3 server and while iLoc also runs off of text input, the extreme slowness of execution renders it almost unusable. It is this dependence of iLoc on SeisComp3 and that SeisComp3 cannot be installed on NCI (with a mysql backend) that stops us from being able to run the entire Picking workflow on the NCI. iLoc requires all parametric data and station inventories within SeisComp3 to be self-consistent (e.g. all arrivals should have corresponding stations in the inventory) – a hard constraint that cannot be fulfilled, given neither our catalogue aggregation/curation nor our inventory curation workflows are perfect. Consequently, the iLoc codebase needed to be patched in order to relax these hard internal constraints. These code changes allow iLoc to pre-process parametric data and compute solutions in cases in which a tiny percentage of stations may be missing.

The changes made to iLoc can be found in the mercurial repository in the folder /home/centos/iLocRelease1.60/src on the AWS instance.

The iLoc phase identification code is launched using an MPI-parallel script as follows.

- From a terminal on the AWS instance, run the following.

```
$ export OPENBLAS_NUM_THREADS=1
$ export PYTHONPATH=$PYTHONPATH:/home/centos/hiperseis/
$ export nprocessors=(number of processors on AWS)
```

- Create a new folder on the AWS and change directory into it.
- Launch iloc_phase_ident.py using the following command.

```
$ mpirun -np $nprocessors python
/home/centos/hiperseis/iloc_rstt/iloc_phase_ident.py
1900-01-01T00:00:00 2021-01-01T00:00:00 ./ --update-db 1 >
./phase_ident.out
```

Upon successful completion, the output folder should contain a bunch of two-columned text files (one for each processor). The first column lists event-ids and the second column shows the iLoc convergence status for each event. Note that we expect a small percentage of events not to converge not only due to some stations metadata being missing, as discussed above, but also due to potential inconsistencies in the arrivals in the curated catalogue. The file, phase_ident.out, contains verbose output from iLoc that may be useful for debugging purposes.

8 Extract Updated Events

We now extract only those events for which the iLoc solutions in step 7 converged and output them in text format. An MPI-parallel script, sc3_extract_events.py, is launched on the AWS instance as follows.

- From a terminal on the AWS instance, run the following.

```
$ export OPENBLAS_NUM_THREADS=1
$ export PYTHONPATH=$PYTHONPATH:/home/centos/hiperseis/
```

- Create a new folder on the AWS instance and change directory into it.
- Run the following in the command line. Assume that <input> is the directory from which iloc_phase_ident.py was run,

```
$ mpirun -np $nprocessors python
/home/centos/hiperseis/iloc_rstt/sc3_extract_events.py
<input> /tmp/ ensemble
```

Upon successful completion, the current folder should contain three files:

- ensemble.all.txt: all arrivals
- ensemble.p.txt: only P, Pg arrivals
- ensemble.s.txt: only S, Sg arrivals

9 Compare Arrivals Before/After Applying iLoc

In this step, we generate various diagnostic plots before and after applying iLoc on the arrivals (using two separate iPython notebooks) and compare them visually. Any first-order processing missteps should reveal themselves in these plots. The notebooks also generate maps for local ($< 10°$) surface-projected raypath coverage within the Australian continent, which are useful in gauging the effective model resolution for tomographic inversion products produced using the output of section 8. Furthermore, plots of distributions of identified phases (after iLoc) over local distances provide clues to the effectiveness of iLoc in filtering out undesired phases e.g. Pn and Sn phases that bottom out in the uppermost mantle.

We generate these diagnostic plots as follows.

- We now copy the ensemble.p.txt and ensemble.s.txt files output in step 8 (on the AWS) back to the NCI.
- Note that ensemble.p.txt and ensemble.s.txt in the above folder are the final artefacts that go into the tomographic inversion workflow. These files are fed into sort_rays.py script in order to prepare input for the Inversion Program.
- Run the ipython diagnostic notebooks diagnostics_before_iloc.ipynb and diagnostics_after_iloc.ipynb.