**HybPiper 2.0 Set up and User Guide For NCI Gadi**

Theo Allnutt, RBGV, 2022.

This guide is intended for users who already have some experience with Unix systems. Gadi is a very high-power HPC and its use is best suited to large data sets (e.g. >50 samples of >1M reads each). If you have smaller data sets than this it may be more convenient to use a local computing resource if available. You must read the Gadi introduction (https://opus.nci.org.au/display/Help/0.+Welcome+to+Gadi) before using this guide and be familiar with the Gadi PBS job queue system. All commands from section 3 and after, should be run on the PBS job system, unless you are only testing small numbers of samples to make sure commands work (e.g. <=8 samples), in which case they could be run directly on the login node, but be aware that login node processes can be killed at any time and cpu / memory intensive work must be carried out in PBS jobs.

All work must be carried out in Gadi storage (/g/data/nm31/<user>) as described below. Data must not be stored in the login node or scratch disks. Please keep any data storage to short term and limit the number of HybPiper outputs you store - HybPiper creates many 100s of files per sample and this can fill our inode quota quickly - killing jobs. Gadi storage is not backed up. You must backup / move important data to your local resources.

**1. Dependencies**

**1.1 Editing your PATH variable**

The PATH variable - this tells the system where to look for programs. We need to add these programs because they are HybPiper2 dependencies. Use the command line text editor 'nano' (or you can use any text editor of your choice).

nano ~/.bash\_profile

Navigate with cursor keys to a blank line at the bottom of the file and paste the following two lines:

module load python3

export PATH="$PATH:/g/data/nm31/scripts/:/g/data/nm31/bin:/g/data/nm31/bin/bbmap:/g/data/nm31/bin/ncbi-blast-2.11.0+/bin:/g/data/nm31/bin/SPAdes-3.15.4-Linux/bin:/g/data/nm31/bin/exonerate/bin:/g/data/nm31/bin/parallel/bin:/g/data/nm31/bin/bwa-0.7.17:/g/data/nm31/bin/samtools/bin:/g/data/nm31/bin/Yang-and-Smith-paralogy-resolution:"

Then save and exit nano:

ctrl-X

"Y" <return>

It is a good idea to open your ~/.bash\_profile file to make sure the lines have been saved.

Then reload the .bash\_profile file:

source ~/.bash\_profile

**1.2 Install biopython and python libraries.**

Make a directory for your Biopython 1.8 installation:

mkdir -p /g/data/nm31/<user>/bin

cd /g/data/nm31/<user>/bin

Download the Biopython 1.8 repository:

git clone https://github.com/biopython/biopython.git

Install Bipython 1.8:

cd /g/data/nm31/<user>/bin/biopython

pip3 install .

Install python dependencies:

pip3 install progressbar2 seaborn pebble psutil

**2. Installing HybPiper2**

Navigate to your bin directory as above:

cd /g/data/nm31/<user>/bin

Download HybPiper2.

git clone https://github.com/mossmatters/HybPiper

cd HybPiper

pip3 install --force-reinstall .

HybPiper2 should now run. The following should print a help message:

hybpiper assemble --help

**3. Preparing files**

**3.1 File names**

You should have your read files for all samples in a single directory. The read files for each sample must be in a single file for unpaired reads, or in two files (R1 and R2) for paired reads. Paired read file names must end in "\_R1.fastq" / "\_R2.fastq", or "\_R1.fastq.gz" / "\_R2.fastq.gz" (only the --diamond option in 'hybpiper assemble' can read .gz files, otherwise they must be uncompressed). The string before the first underscore in the file name will be used as the sample ID for HybPiper2 results.

**3.2 Concatenating files**

If read files are from multiple sequencing runs or lanes (more than two per sample for paired reads, usually denoted in Ilummina sequencing by 'L1.. L2..' etc.), then they must be concatenated into two files prior to HybPiper2 analysis. There is a script for this on Gadi nm31:

cat-turbo.py reads/ cat/ 12

Where "reads/" is the directory of reads containing multiple runs / lanes per sample; "cat/" is the output directory; and '12' is the number of threads to use (<48). The infolder can contain a mixture of multiple and single run/lane files, the script will detect if files need to be concatenated or not. The script uses the sample ID before the first underscore in the name and the pair identifier, 'R1' or 'R2', to concatenate files. All the files of a sample must be either .gz or uncompressed, not a mixture of both.

**3.3 Quality filtering reads**

Once files are concatenated, they can be filtered. Users can do this themselves with any program they prefer. We use a 'fastp' wrapper for filtering:

fastp.py "cat/\*" filt/ 50 8 6 paired

Where "reads/\*" contains your concatenated read files (this must be within quotation marks as shown and have the '\*' included), and "filt/" is the output directory. '50' is the minimum read length to retain; '8' is the number of samples to process in parallel; and '6' is the number of parallel threads. Note, the maximum number of threads on the Gadi 'normal' queue and login nodes is 48, so samples X threads cannot exceed 48.

**3.4 Entropy filtering reads**

This is not essential, however, HybPiper assembly can become very slow and prone to errors if read files contain many low entropy reads with simple sequence, such as homopolymers and short repeats. These reads do not contribute to target capture assembly and their removal will reduce file sizes and increase speed. The following script will do this.

shannons-filter.py "filt/\*" shan/ 0.8 12 paired 50000000

Where "filt/\*" is the input directory as previously; "shan/" is the output directory; '0.8' is the Shannon's information index limit for reads to be passed (we have found that 0.8 is a good threshold to use); '12' is the number of threads; 'paired' specifies paired reads; and '50000000" is the number of reads to pass (most read files will be much smaller than this so it effectively processes all reads, however you can use this option to limit read numbers if required, which speeds up processing if read files are very large - we usually limit HybPiper to 20M reads without loss in gene recovery).

**4. Running HybPiper2**

**4.1 Submitting HybPiper2 to the job queue**

The following command examples must be run from the directory where you want your results to be stored. Change <username> to your Gadi username.

e.g. make a run directory and move to it.

cd /g/data/nm31/<user>/

mkdir hp2run

cd hp2run

Copy your target file to the directory using your preferred method.

Make a text file of the HybPiper2 run script to be submitted to the job queue. The text file should look like below and be called 'hp2.sh'.

#!/bin/bash

#PBS -P nm31

#PBS -q normal

#PBS -l walltime=4:00:00

#PBS -l ncpus=48

#PBS -l mem=128GB

#PBS -l jobfs=100GB

#PBS -l wd

#PBS -l storage=gdata/nm31+scratch/nm31

module load python3

#this copies done.txt to FS - if first run then gives error that can be ignored

cp ./done.txt $PBS\_JOBFS/

cd $PBS\_JOBFS

hp2.py /g/data/nm31/hp2run/<user>/shan/ /g/data/nm31/<user>/hp2run/Angiosperms353.FAA aa <home-path> "--diamond --cov\_cutoff 5" 8 6

rsync -rut ./ $PBS\_O\_WORKDIR/

The lines starting with '#PBS' set the resources requested for the job. Adjust these as required. Limits can be found here: https://opus.nci.org.au/display/Help/Queue+Limits. The next line loads the Gadi python3 environment. PBS jobs should be run so they I/O to the job node storage (=$PBS\_JOBFS), therefore 'done.txt' is first copied to this location, and the job is cd'd to it. The next line runs the HybPiper2 Gadi wrapper script, hp2.py. This script will submit all the samples in your (filtered) reads directory until they have all completed in manageable batches of eight at a time. In this example "shan/" is the directory of your filtered reads. You must use the full path to the reads and target file directory. "Angiosperms353.FAA" is the target file in amino acid sequence format; "aa" tells the script the targets are amino acid. If you are using a DNA sequence target file, change this to "dna". This is followed by your NCI login node path, <home-path>. To get this path, login and type 'pwd', it will be something like, "/home/554/xy1234". The text in quotes ("--diamond--cov\_cutoff 5 ") contains all other 'hybpiper assemble' options - which you can see in the help message as shown above. "--diamond " alone should be sufficient for most purposes. '8' is the number of samples to process at a time, and '6' is the number of threads per sample. As mentioned above, their product should not exceed 48. '8' and '6' have been optimised for Gadi and you should not need to change them unless you are running fewer than eight samples and you wish to use more threads per sample.

IMPORTANT: While your job is running, do not delete, rename, or move any files in your run or results directories

Submit the job script:

qsub hp2.sh

In this example, walltime is four hours. Approximately one to two hours is required per eight samples but this could increase if read files are very large. Maximum walltime for a Gadi job is 48 hours.

Using PBS\_JOBFS is much faster than using /g/data/, but results are not copied to, or visible in, your run folder (the directory from where you ran the job script) until the job has finished.

**4.2 Tips on watching run progress**

To see how your PBS batch job is progressing, use:

watch -n 60 qstat -u <username>

This will update every 60 seconds, ctrl-c to exit the watch command.

**4.3 Checking results**

hp2.py saves a file listing of completed samples in your run directory, called 'done.txt'. If a job timeouts before all samples have been completed, you can check this file to see which samples finished. If you resubmit the job, hp2.py will read done.txt and continue for your remaining samples automatically. If you want to repeat a sample(s), delete their ID(s) from done.txt. To repeat the whole job, delete done.txt.

All results are stored in a directory in your run directory, called 'results'. Within this, each sample ID will have its own directory containing outputs of HybPiper. In addition, the full output of HybPiper is stored in a compressed file e.g. 12345.tar.gz.

The "\_stats.tsv" file in each result directory provides the best indication of whether a sample's run has been successful or not. This file shows the stats of gene recovery for each sample.

A log directory is created in your run directory. This contains HybPiper stdout and stderr for all samples (stderr has the output normally printed to screen when not run in a job queue, stdout normally contains no information). If you suspect a run has failed or has a problem, its .err file is the first place to check.

**4.4 Collating all samples' results**

Once all your samples have run successfully. The paralog, stats, and gene length files can be concatenated. From your run directory, run the following (there is no need to submit this as a job):

join\_hp\_results\_para.py results/ 12

where 'results/' is your result directory and '12' is the number of threads. This will create (in the results directory) a directory for all samples' paralogs for all target genes ("paralogs\_no\_chimeras\_joined" and "all\_paralogs\_joined"), and files containing all samples' stats and gene lengths.

**Important - Citation**

**Please cite ABLeS in any publications resulting from use of NCI resources, see citation formats here: https://australianbiocommons.github.io/ables/acknowledgements/**

**Citation ensures the continued availability of NCI resources to the community.**