SUPERCOMPUTING WALES
UWCHGYFRIFIADURA CYMRU

Guide to Running ***ABySS***

What is ABySS?

ABySS (Assembly By Short Sequences) is a de novo, parallel, paired-end sequence assembler that is designed for short reads. The single-processor version is useful for assembling genomes up to 100 Mbases in size. The parallel version is implemented using MPI and is capable of assembling larger genomes. ABySS is described in detail in the article available at http://genome.cshlp.org/content/19/6/1117.long.

This guide to running ABySS provides instructions on how to run a standard ABySS test case on the Supercomputing Wales systems.

Step 1 - Log in

The examples used in this guide is configured to run on the Cardiff Skylake *Hawk* cluster. Connect to *hawklogin.cf.ac.uk* with your Supercomputing user credentials using your preferred method (e.g. PuTTY from a Windows machine or ssh from any Linux terminal, thus

```
ssh -l your_username hawklogin.cf.ac.uk
```

The steps below involve typing commands (**in bold font**) in the terminal window.

Step 2 - Load an ABySS module

In common with most of the application guides in this series, we assume at the outset that the module of choice would be selected from those originally available on HPC Wales. Thus a necessary first step would be to gain access to that module set by issuing the commands

```
module purge
module load hpcw
```

Note that the “module purge” command is required to prevent any unintentional collision with pre-existing modules. Subsequently issuing the module command

```
module avail
```

will provide visibility of and access to the entire set of HPCW modules.

- List preinstalled ABySS versions:

  ```
  module avail ABySS
  ```

- Load your preferred version (version 1.3.6 is used in this tutorial):

  ```
  module load ABySS/1.3.6
  ```

- Confirm the loaded modules:

  ```
  module list
  ```

  Note: Dependent modules, e.g. compiler and MPI library, Boost, etc. are loaded automatically.

Step 3 - Create a directory

From your home directory, create a directory to hold any user data files. For this tutorial, a directory called ABySS should be created:

```
cd ~
mkdir ABySS
cd ABySS
```

Step 4 - Obtain a test case

A test case for ABySS is provided with the installation at

```
/app/genomics/ABySS-1.3.6/example/
```

Copy the input files and SLURM job script:

```
cp -rp /app/genomics/ABySS-1.3.6/example/* .
```

Step 5 - Submit a parallel job

The supplied SLURM job script runs using four processes, and should run for about one minute.

- Submit the job using:

  ```
  sbatch ABySSExampleSubmit.SLURM.q
  ```

- Check the job queue using:

  ```
  squeue
  ```

- When completed, output can be found in a file called ABySS.Hawk.o.<JobID> and errors, if any, can be found in ABySS.Hawk.e.<JobID> (where <JobID> is the ID generated by the queuing system). Other output files created by ABySS will be found in the directory:

  ```
  /scratch/$USER/run_abyss_id-<JobID>.
  ```

Step 6 - Rerun with other parallel settings

Now open the *ABySSExampleSubmit.SLURM.q* jobscript using your favourite editor, such as nano, emacs or vi.

Lines 5 and 6 of the job script read:

```
#SBATCH --ntasks=4           # number of parallel processes (tasks)
#SBATCH --ntasks-per-node=4 # tasks to run per node,
```

meaning that 4 cores are used to run this job. Change both numbers to, say, 8. You will also need to change the 'np' parameter to the abyss-pe command as follows:

```
abyss-pe -j1 k=35 n=5 np=8 c=5 mpirun=mpirun …
```

Then resubmit the job - what happens to the job time?

Further info

Further information on ABySS can be found at the website http://www.bcgsc.ca/platform/bioinfo/software/abyss/, and documentation may be found at https://github.com/bcgsc/abyss#abyss.