ABySS for Intel® Xeon® Processor

This code recipe describes how to get, build, and use the ABySS de novo assembly code for the Intel® Xeon® processor.

Introduction

ABySS (Assembly By Short Sequences) is a de novo, parallel, paired-end sequence assembler that is designed for short reads.

The single-node version is useful for assembling genomes up to 100 Mbases in size. This version is threaded for concurrent execution. There is also a parallel version of ABySS implemented using MPI and capable of assembling larger genomes. The script **abyss-pe** will run a more comprehensive set of tools to process paired-end data.

The most current version of ABySS source files can be downloaded from http://www.bcgsc.ca/platform/bioinfo/software/abyss.

Support Software

To build the ABySS application requires the installation of the Boost Graph Library and the Google Sparsehash library. The current version of the Boost library can be found at www.boost.org; the current version of the Sparsehash library can be found at https://code.google.com/p/sparsehash/.

Build and Install Boost Graph Library

1. Download the current version of the software from www.boost.org.
2. Unzip and untar the downloaded file into a desired directory; change to the expanded Boost directory.
3. Execute the **bootstrap.sh** shell script with the **--prefix** parameter for the root directory in which to install the library files. If you have admin privileges, you may wish to put this in some global directory such as **/usr/local**.

```
$> ./bootstrap.sh --prefix=/usr/local
```

4. Install the library files using the **b2** script.

```
$> sudo ./b2 install
```

Build and Install Sparsehash Library

1. Download the current version of the software from https://code.google.com/p/sparsehash/.
2. Unzip and untar the downloaded file into a desired directory; change to the expanded Sparsehash directory.
3. Use **configure** to set up build environment. The compiler and root directory for installation, among other options, can be given as command line arguments. For example,

```
$> configure CC=icc CXX=icpc —-prefix=
/usr/local
```

4. Use **make** to build the library.

```
$> make
```

5. Install the library.

```
$> sudo make install
```

Build and Install the ABySS Software

1. Download the current version of the software from http://www.bcgsc.ca/platform/bioinfo/software/abyss.
2. Unzip and untar the downloaded file into a desired directory; change to the expanded ABySS directory.
3. Use **configure** to set up build environment; include your compiler choice, the location of the include files for the Boost library, any compiler flags you want to use, and the root directory in which you want to install the ABySS built programs and documentation.

```
$> ./configure CC=icc CXX=icpc                \
      --with-boost=/usr/local/include    \
      CPPFLAGS=-I/usr/local/include      \
      --prefix=/usr/local
```

4. Use **make** to build the library.

```
$> make
```

5. Install the library.

```
$> sudo make install
```

Running ABySS on a Human Genome Data Set

As an example, the following steps describe how to download a large human genome data set (ERR194147) and run the ABySS genome assembly software on the data set.

1. Download the two paired-end data files from http://www.ebi.ac.uk/ena/data/view/ERR194147&display=html
   a. Direct URLS for the files are:

      ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR194/ERR194147/ERR194147_1.fastq.gz

      ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR194/ERR194147/ERR194147_2.fastq.gz

   b. The files can be left as compressed FASTQ files or uncompressed for input to ABySS.
2. The following command will run the ABYSS executable with the data files from the previous step. This assumes that the command is run from the directory in which the data files reside; if this is not the case, add directory information to the file name parameters to locate these files from the execution directory.

```
$> ABYSS -k57 —j 16 \
    --coverage-hist=coverage.hist \
    -s err194147-bubbles.fa \
    -o err194147-1.fa \
    ERR194147_1.fastq.gz ERR194147_2.fastq.gz
```

   a. The **-k** parameter sets the length of the kmers to be used in the de Bruijn graph. The default maximum is 64. A good setting is some value over half the length of the input reads.
   b. The number of threads is specified with the **-j** flag.
   c. The above command will generate the files **coverage.hist, err194147-bubbles.fa**, and **err194147-1.fa**.

Running ABySS Paired-End Analysis

An alternate execution that will utilize the ABYSS executable and many other tools in the ABySS suite is available when you have paired end data (as given above). This uses the **abyss-pe** script file.

1. The following command will run the abyss-pe script with the data files from the previous steps. This assumes that the command is run from the directory in which the data files reside; if this is not the case, add directory information to the file name parameters to locate these files from the execution directory.

```
$> abyss-pe name=err194147 k=57 j=16 \
in='ERR194147_1.fastq.gz ERR194147_2.fastq.gz'
```

a. The assembled contigs output will be stored in the file **err194147-contigs.fa** (using the **name** parameter from the command line).

b. The parameter **in** specifies the input files. The pair of read files must be named with the suffixes **1** and **2** to identify the first and second read. A single file with the paired reads interleaved could be used.

Build and Install the ABySS Software for Distributed Memory Execution

1. Download the current version of the software from http://www.bcgsc.ca/platform/bioinfo/software/abyss.
2. Unzip and untar the downloaded file into a desired directory; change to the expanded ABySS directory.
3. Use **configure** to set up build environment; include your compiler choice, the location of the include files for the Boost library, the path to the home of the MPI files, any compiler flags you want to use, and the root directory in which you want to install the ABySS built programs and documentation.

```
$> ./configure CC=mpiicc CXX=mpiicpc        \
    --with-boost=/usr/local/include \
    --with-MPI=$MPI_HOME_DIR         \
    CPPFLAGS=-I/usr/local/include   \
    --prefix=/usr/local
```

   a. The above will use the Intel MPI compiler script that employs the Intel compiler. (The Open MPI library is the version recommended by the code authors.)

4. Use **make** to build the library.

```
$> make
```

5. Install the library.

```
$> sudo make install
```

Running the Distributed Memory Version of ABySS

1. The following instructions assume that you have downloaded the ERR194147 data files or some other appropriate data set.
2. The following command will run the ABYSS executable in a distributed memory fashion. The launch is done via the appropriate "run" command used by the MPI library that was used to build the application. The command assumes that it is run from the directory in which the data files reside; if this is not the case, add directory information to the file name parameters to locate these files from the execution directory.

```
$> mpirun -np 8 ABYSS-P -k57 \
    --coverage-hist=coverage.hist \
    -s err194147-bubbles.fa \
    -o err194147-1.fa \
    ERR194147_1.fastq.gz ERR194147_2.fastq.gz
```

   a. The executable **ABYSS-P** is the MPI-enabled version of the ABYSS application.

   b. The above will start 8 processes (**-np 8**) and divide up the input reads from the data files. Each process will use its set of reads to construct the overall de Bruijn graph. Adding a **-j** parameter to the **ABYSS-P** call will use the given number of threads within each process.

Running ABySS Paired-End Analysis on Distributed Memory Platform

An alternate execution that will utilize the **ABYSS-P** executable and many other tools in the ABySS suite is available when you have paired end data (as given above). This uses the **abyss-pe** script file.

1. The following command will run the **abyss-pe** script using the **ABYSS-P** executable. (At the time of this writing, none of the other ABySS tools were written to run on multiple processes.)

```
$> abyss-pe name=err194147 k=57 np=8 \
    in='ERR194147_1.fastq.gz ERR194147_2.
    fastq.gz'
```

   a. If compiled for distributed execution, you MUST NOT launch the script with something like "**mpirun -np 8 abyss-pe**". The **abyss-pe** driver script will launch the MPI processes.