Instruction Manual for the Ray *de novo* genome assembler software

Sébastien Boisvert

June 27, 2014

Ray version 1.6.2-devel

Website: `http://denovoassembler.sf.net`

Git source tree: `http://github.com/sebhtml/ray`

Reference to cite:

Sébastien Boisvert, François Laviolette & Jacques Corbeil.
Ray: simultaneous assembly of reads from a mix of high-throughput sequencing technologies.
Journal of Computational Biology (Mary Ann Liebert, Inc. publishers, New York, U.S.A.).
November 2010, Volume 17, Issue 11, Pages 1519-1533.
doi:10.1089/cmb.2009.0238
`http://dx.doi.org/doi:10.1089/cmb.2009.0238`

Contents

1 Abstract **3**

2 Installation **3**
2.1 Use longer k-mers . . . 4

3 General guide lines for launching Ray **4**
3.1 Communication . . . 4
3.2 Scheduling a MPI job . . . 4

4 Inputs **5**
4.1 Unknown nucleotides (not A, T, C, G) . . . 5

5 Parameters **5**
5.1 -debug-bubbles . . . 5
5.2 -write-kmers . . . 5
5.3 -debug-seeds . . . 5
5.4 -show-memory-usage (only available in GNU/Linux) . . . 6
5.5 -show-ending-context . . . 6
5.6 -run-profiler . . . 6
5.7 -amos . . . 6
5.8 -help . . . 6
5.9 -minimumCoverage minimumCoverage . . . 6
5.10 -peakCoverage peakCoverage . . . 6
5.11 -repeatCoverage repeatCoverage . . . 6
5.12 *k*-mer length with -k . . . 6
5.13 Output prefix with -o . . . 6
5.14 Single-end reads with -s . . . 6
5.15 Paired-end reads and mate-pair reads with -p . . . 7
5.16 Paired-end reads and mate-pair reads with -i (interleaved sequences) . . . 7

6 Outputs **7**

7 Validation of assemblies **8**

8 Examples **8**
8.1 Bacterial genome with paired-end and mate-pair short reads . . . 8

9 Virtual sequencer & simulations **9**

10 Exploring the source code **9**

11 Submitting patches **9**

12 Reporting bugs & crashes **10**

1 Abstract

Ray is documented in the manual (InstructionManual.tex; available online and in Portable Document Format), on `http://github.com/sebhtml/ray` (with the README.md), in the C++ code using Doxygen and in the journal paper as well. Enjoy the manual.

2 Installation

To install Ray, you need a C++ compiler (standard 1998), make and a message-passing-interface (MPI) implementation compliant with the MPI standard 2.2.

The software below are readily (and freely) available in most GNU/Linux distributions (I use Ubuntu, CentOS, and Fedora); see Table 1).

Table 1: Suggested software

Software	Name
C++ compiler	GNU g++
MPI implementation	Open-MPI
make	GNU make

There are some compilation options that can be changed. These are listed below in Table 2. They can be changed in the Makefile or provided on the command line.

Table 2: Compilation options

Option	Value	Description
PREFIX	path	where 'make install' will install Ray
MAXKMERLENGTH	integer	maximum k-mer length
HAVE_LIBZ	y/n	enable support for .gz files
HAVE_LIBBZ2	y/n	enable support for .bz2 files
FORCE_PACKING	y/n	enable packing of structures and classes
ASSERT	y/n	enable assertions in the code
HAVE_CLOCK_GETTIME	y/n	get time at a nanosecond precision – real-time
VIRTUAL_SEQUENCER	y/n	will generate a program called VirtualNextGenSequencer
INTEL_COMPILER	y/n	use mpiicpc

To compile Ray, enter these commands.

```
make PREFIX=build
make install
ls build/Ray
```

You need to have mpic++ in your path, otherwise edit the Makefile to change MPICXX. You may also provide MPICXX on the command line.

```
make PREFIX=build MPICXX=/software/openmpi-1.4.3/bin/mpic++
make install
ls build/Ray
```

You can also provide all compilation options as follows.

```
make  PREFIX=build MPICXX=/software/openmpi-1.4.3/bin/mpic++ \
MAXKMERLENGTH=64 HAVE_LIBZ=n HAVE_LIBBZ2=n \
```

```
HAVE_CLOCK_GETTIME=n INTEL_COMPILER=n ASSERT=n FORCE_PACKING=y
#wait
make install
```

No configure script exists because it is too much of a mess to maintain. Furthermore, autoconf and automake just don't work on a Lustre filesystem (used on most super computers) when the network file system disallows file locking. The Makefile of Ray utilises the Linux kernel syntax (obj-y += object.o).

2.1 Use longer k-mers

```
make PREFIX=Ray-Large-k-mers MAXKMERLENGTH=64
# wait
make install
mpirun -np 512 Ray-Large-k-mers/Ray -k 63 -p lib1_1.fastq lib1_2.fastq \
-p lib2_1.fastq lib2_2.fastq -o DeadlyBug,Assembler=Ray,K=63
# wait
ls DeadlyBug,Assembler=Ray,K=63.Scaffolds.fasta
```

Note: longer k-mers utilise more memory.

3 General guide lines for launching Ray

3.1 Communication

Ray is a high-performance peer-to-peer software. It runs on numerous computers simultaneously. Ray instances are called MPI ranks. The MPI ranks are usually mapped to processes and they communicate using available bit transfer layers. Bit transfer layers are (in Open-MPI):

- **Direct memory** copy is used when an MPI rank communicates with itself;
- **Shared memory** is used when an MPI rank communicates with another MPI rank located on the same computer;
- **TCP/IP** is used when an MPI rank communicates with another MPI rank located on a different computer.

If you use a high-performance network:

- **Infiniband** is used when an MPI rank communicates with another MPI rank located on a different computer.

3.2 Scheduling a MPI job

One thing that should be avoided is sharing cluster nodes between users for any given MPI job. Otherwise, your MPI job may fail simply because other users sharing your job nodes may make one of your job nodes crash (load factor is too high or memory usage is not respectful).

If you use Oracle Grid Engine (previously known as Sun Grid Engine), you should submit your MPI jobs to a queue with an allocation rule (allocation_rule) equal to the number of cores per node. This simple configuration will ensure that no cluster node is shared between users for any MPI job.

4 Inputs

For a most up-to-date list of Ray parameters, run:

```
Ray -help|less
```

The input files for Ray contain sequences. The files must be formatted in one of the supported formats. These formats are listed in Table 3. Note that the file extension is obligatory and Ray uses it to select the file format.

Table 3: File formats compatible with the Ray *de novo* genome assembler software

Format	Mandatory extension
Fasta format	.fasta
Fasta format, compressed with GNU zip (gzip)	.fasta.gz
Fasta format, compressed with bzip2	.fasta.bz2
Fastq format	.fastq
Fastq format, compressed with GNU zip (gzip)	.fastq.gz
Fastq format, compressed with bzip2	.fastq.bz2
Standard flowgram format	.sff
Color-space	.csfasta

Keep in mind that these formats are *just* containers. To use 454 mate-pairs in an SFF file, you must extract them and provide Ray with the 2 resulting fastq files.

Ray assembles color-space reads and generate color-space contigs. Files must have the .csfasta extension. Nucleotide reads can not be mixed with color-space reads. This is an experimental feature.

4.1 Unknown nucleotides (not A, T, C, G)

Unknown nucleotides at the ends of reads are removed and unknown nucleotides inside reads are converted to A.

5 Parameters

Ray assembles reads (paired or not) to produce an assembly. Paired reads must be on opposite strands (forward & reverse or reverse & forward).

For a paired library, paired reads can be provided as two files (with -p) or as one file containing interleaved sequences (with -i). With both, the average outer distance and the standard deviation can be provided by the user. Otherwise, these values are computed by Ray. Ray may fail to compute and average outer distance when the later is too high. In that case, provide the value. The maximum number of paired libraries allowed by Ray is 499.

5.1 -debug-bubbles

Debugs bubble code.

5.2 -write-kmers

Ray will write k-mers to PREFIX.kmers.txt.

5.3 -debug-seeds

Debugs seed code.

5.4 -show-memory-usage (only available in GNU/Linux)

Shows memory usage. Data is fetched from /proc on GNU/Linux

5.5 -show-ending-context

Shows the ending context of each extension.

5.6 -run-profiler

Runs the profiler as the code runs. Needs a real-time POSIX system (HAVE_CLOCK_GETTIME = y)

5.7 -amos

Generates an AMOS file containing read locations on contigs. This file can be opened with AMOS Hawkeye, Tablet and other assembly browser software. The generated file is PREFIX.AMOS.afg.

5.8 -help

Shows options available.

5.9 -minimumCoverage minimumCoverage

Sets manually the minimum coverage. If not provided, it is computed by Ray automatically.

5.10 -peakCoverage peakCoverage

Sets manually the peak coverage. If not provided, it is computed by Ray automatically.

5.11 -repeatCoverage repeatCoverage

Sets manually the repeat coverage. If not provided, it is computed by Ray automatically.

5.12 k-mer length with -k

Ray builds a distributed catalog of all occuring k-mers in the reads and their reverse-complement. k must be greater or equal to 15 and lower or equal to MAXKMERLENGTH (provided at compilation). The k-mer length must be an odd number.

5.13 Output prefix with -o

Output files are named according to the prefix provided by the option -o.

5.14 Single-end reads with -s

```
-s <sequencesFile>
```

5.15 Paired-end reads and mate-pair reads with -p

Average outer distance and standard deviation are computed by Ray if omitted.

```
-p <leftSequencesFile> <rightSequencesFile>
```

OR

```
-p <leftSequencesFile> <rightSequencesFile>  <averageFragmentLength> <standardDeviation>
```

Example for paired-end reads (the ends of DNA fragments):

```
-p lib1_1.fastq lib1_2.fastq -p lib2_1.fastq lib2_2.fastq
```

Example for mate-pair reads:

```
-p s_20000_1.fastq s_20000_2.fastq
```

Example with metagenomic data and user-provided average outer distance and standard deviation

```
-p s_1.fastq s_2.fastq 300 30
```

5.16 Paired-end reads and mate-pair reads with -i (interleaved sequences)

Average outer distance and standard deviation are computed by Ray if omitted.

```
-i <sequencesFile>
```

OR

```
-i <sequencesFile>  <averageFragmentLength> <standardDeviation>
```

In the interleaved file (example is for a fasta file):

```
>200_1_1234/1
ATCGATCGATCGACTCAGACACGTACG
>200_1_1234/2
ACTGACGACGTACGACGTCATGCAACT
...
```

6 Outputs

Outputted files are listed in Table 4.

The file PREFIX.CoverageDistributionAnalysis.txt contains an estimation of the genome length.

Table 4: Files generated by the Ray *de novo* genome assembler software

File	Description
PREFIX.Contigs.fasta	Contiguous sequences in FASTA format
PREFIX.ContigLengths.txt	The lengths of contiguous sequences
PREFIX.Scaffolds.fasta	The scaffold sequences in FASTA format
PREFIX.ScaffoldComponents.txt	The components of each scaffold
PREFIX.ScaffoldLengths.txt	The length of each scaffold
PREFIX.ScaffoldLinks.txt	Scaffold links
PREFIX.RayCommand.txt	The exact same command provided
PREFIX.RayVersion.txt	The version of Ray
PREFIX.CoverageDistribution.txt	The distribution of coverage values
PREFIX.CoverageDistributionAnalysis.txt	Analysis of the coverage distribution
PREFIX.LibraryStatistics.txt	Number of reads in each file and estimation of outer distances for paired reads
PREFIX.SeedLengthDistribution.txt	The distribution of seed lengths
PREFIX.OutputNumbers.txt	Overall numbers for the assembly
PREFIX.AMOS.afg	Assembly representation in AMOS format (-amos)
PREFIX.kmers.txt	K-mer graph (-write-kmers)
PREFIX.degreeDistribution.txt	Distribution of ingoing and outgoing degrees
PREFIX.MessagePassingInterface.txt	Contains the number of messages sent
PREFIX.NetworkTest.txt	Contains the result of the network test

7 Validation of assemblies

In the scripts/ directory, the script ValidateGenomeAssembly.sh can validate an assembly against the corresponding reference. To run it, you will need MUMmer and some other programs from AMOS (Google them).

8 Examples

Examples are available in system-tests/tests

8.1 Bacterial genome with paired-end and mate-pair short reads

The command:

```
mpirun -np 32 ./build-99/Ray \
-p /home/boiseb01/nuccore/Large-Ecoli/200_1.fastq \
   /home/boiseb01/nuccore/Large-Ecoli/200_2.fastq \
-p /home/boiseb01/nuccore/Large-Ecoli/1000_1.fastq \
   /home/boiseb01/nuccore/Large-Ecoli/1000_2.fastq \
-p /home/boiseb01/nuccore/Large-Ecoli/4000_1.fastq \
   /home/boiseb01/nuccore/Large-Ecoli/4000_2.fastq \
-p /home/boiseb01/nuccore/Large-Ecoli/10000_1.fastq \
   /home/boiseb01/nuccore/Large-Ecoli/10000_2.fastq \
-o BacterialGenome | tee RayLog
```

9 Virtual sequencer & simulations

The Ray package includes a simulator for paired reads. To build it, you must provide VIRTUAL_SEQUENCER=y. Note that you need boost to compile this tool.

To compile it, type these commands:

```
make PREFIX=build VIRTUAL_SEQUENCER=y
# wait
make install
./build/VirtualNextGenSequencer
```

To use it:

```
N=6000000
readLength=50
errorRate=0.005
ref=~/nuccore/Ecoli-k12-mg1655.fasta

./build/VirtualNextGenSequencer $ref $errorRate 200 20 $N $readLength L1_1.fasta L1_2.fasta
```

You can then assemble these reads with Ray.

```
mpirun -np 64 ./build/Ray -k 31 -p L1_1.fasta L2_2.fasta -o DeadlyBug
```

10 Exploring the source code

The source code of Ray (and all the changes since its inception) is on github. github is a place where people code socially. There, you can browse the source code easily.

`http://github.com/sebhtml/ray`

Feel free to fork my tree and send me a pull request if you have some interesting changes to the code base.

The Ray source code is documented with Doxygen tags.

The command below generates code documentation.

```
cd code
doxygen DoxygenConfigurationFile
```

To browse it, use this command:

```
firefox DoxygenDocumentation/html/index.html
```

11 Submitting patches

See code/Submit-a-patch.txt

12 Reporting bugs & crashes

If Ray crashed, you should send a bug report. However, be sure that you can reproduce the bug when Ray is compiled with ASSERT=y. Assertions are useful to catch those pesky bugs.

```
On the mailing list:

denovoassembler-users # lists DOT sourceforge.net

(replace # with @)
```

Elements to send:

1. Ray command including mpirun
2. Ray standard output: mpirun -np 32 /path/to/Ray -p p_1.fastq p_2.fastq -o Out |& tee Log; gzip Log; ls -lh Log.gz
3. C++ compiler name (examples: GNU g++, Intel ICC, or other) and version (usually −−version)
4. MPI library name (examples: Open-MPI, MPICH2, or other) and version (usually −−version)
5. Sequencer vendor and model that generated reads
6. Total physical memory of the system and physical memory available for each MPI ranks
7. Job scheduler (examples: Grid Engine, PBS, none or other)

Also, if you have access to compute nodes:

8. Memory page size: getconf PAGESIZE > pagesize
9. Central processing unit: cat /proc/cpuinfo—gzip> cpuinfo.gz
10. Memory information: cat /proc/meminfo—gzip> meminfo.gz
11. Kernel: uname -a> uname-a

THE END