HSRA: User's Guide

Authors:
Roberto R. Expósito, Jorge González-Domínguez and Juan Touriño

January 23, 2019

Contents

1 Introduction

The Hadoop Spliced Read Aligner (HSRA) [1, 2] is a MapReduce-based [3] parallel tool for mapping reads from RNA sequencing (RNA-seq) experiments [4] to a reference genome. HSRA currently runs on all major 64-bit Linux distributions, supporting single-end and paired-end read alignments from FASTQ/FASTA datasets.

RNA-seq analyses typically begin by mapping reads to a reference genome in order to determine the location from which the reads were originated. The rapid progress of Next Generation Sequencing (NGS) technologies has led to an explosion in the amount of sequencing data available, which makes the read alignment step very time-consuming in bioinformatics pipelines. This tool allows bioinformatics researchers to efficiently distribute their mapping tasks over the nodes of a computer cluster by combining a fast multithreaded spliced aligner (HISAT2 [5, 6]) with the Apache Hadoop project [7], which is the most popular distributed computing framework for scalable Big Data processing.

This tool is distributed as free software and is publicly available at [2] under the GPLv3 license [8].

1.1 Citation

If you have used HSRA in your research, please cite our work using reference [1].

2 Prerequisites

In order to use HSRA, the prerequisites are:

1. Make sure you have Java Runtime Environment (JRE) version 1.8 or above
2. Make sure you have a working Hadoop distribution version 2.2 or above [9]
   - HADOOP_HOME environmental variable must be set accordingly
3. Make sure you have a working HISAT2 distribution
   - https://ccb.jhu.edu/software/hisat2/manual.shtml
4. Untar the downloaded HSRA distribution
   - On Linux, just follow the instructions below:

```
[user@host ~]$ tar xzf HSRA-v1.1.tar.gz
```

   - Alternatively use your preferred archive extraction tool
5. Set HSRA_HOME, HISAT_HOME and PATH environmental variables
   - On Linux, you can set them in your profile or your shell configuration files (e.g., .bashrc). Follow the instructions below:

```
[user@host ~]$ export HSRA_HOME=/path/to/hsra
[user@host ~]$ export HISAT_HOME=/path/to/hisat
[user@host ~]$ export PATH=$HSRA_HOME/bin:$PATH
```

3 Execution

HSRA can be executed by using the provided *hsrarun* command, which launches the MapReduce jobs to the Hadoop cluster. Three compulsory arguments are needed for single-end read alignments: 1) the path to the reference genome index; 2) the input dataset containing unpaired reads to be aligned; and 3) the memory requirements for the alignments tasks. In case of paired-end read alignments, a fourth argument is needed that indicates the second dataset. Note that the input FASTQ/FASTA datasets must be stored in the Hadoop Distributed File System (HDFS) [10], while the genome index must be accessible from all the nodes in the cluster either in a local or shared path. HSRA uses the Hadoop Sequence Parser (HSP) library [11] to efficiently read the input datasets from HDFS. The datasets can be in compressed format (see Section 3.3 for more information about this topic).

HSRA distributes the input dataset(s) over the nodes of the Hadoop cluster. The number of alignment tasks to execute can be specified via a command-line option, as will be shown in Section 3.1. The underlying aligners are executed independently to map the reads assigned to them to the reference genome. Optionally, their corresponding output files can be merged into a single SAM output file. Note that this merge step can be enabled via a configurable option (see Section 3.4). Nevertheless, avoiding the merge operation can be very time-saving in those scenarios where further downstream analysis of the SAM output is going to be performed directly on HDFS [12], thus avoiding significant disk I/O overhead.

3.1 Command-line arguments

The available command-line arguments are:

- -x. Compulsory both in single-end and paired-end scenarios. String with the path to the reference genome index, which must be accessible from all the nodes in the cluster.
- -s. Compulsory both in single-end and paired-end scenarios. String with the HDFS path to the first sequence file in FASTQ/FASTA format.
- -m. Compulsory both in single-end and paired-end scenarios. Integer with the memory requirements in MiBytes for the alignments tasks.
- -p. Compulsory in paired-end scenarios. String with the HDFS path to the second sequence file in FASTQ/FASTA format.
- -q. Specify that input sequence files are in FASTQ format. By default, HSRA tries to autodetect the input format, but if input files are compressed the user must specify the appropriate argument.
- -f. Specify that input sequence files are in FASTA format. By default, HSRA tries to autodetect the input format, but if input files are compressed the user must specify the appropriate argument.
- -o. Optional. String with the HDFS path to the output file in SAM format. The default value is the same as the first input file followed by the `.sam` extension.

- -na. Optional. Integer with the number of alignment tasks per node. The default value is 1.
- -nt. Optional. Integer with the number of parallel threads per alignment task. The default value is 1.
- -multi-wave. Optional. Specifying this argument allows HSRA to execute the alignment tasks in multiple waves of map/reduce tasks.
- -args. Optional. Quoted-string with any other arguments to be passed directly to the alignment tasks.
- -h. Print out the usage of the program and exit.
- -v. Print out the version of the program and exit.

3.2 Examples

The following command maps a set of unpaired reads to a reference genome using 1 alignment task per node, requesting 6 GiB of memory (i.e., 6144 MiB) and 8 threads per task:

```
[user@host ~]$ hsrarun -x /genomes/hg38/hg38_tran -s dataset.
   fastq -m 6144 -nt 8
```

The following command shows a similar example but for paired-end alignment, using 2 alignment tasks per node, requesting 6 GiB of memory (i.e., 6144 MiB) and 4 threads per task:

```
[user@host ~]$ hsrarun -x /genomes/hg38/hg38_tran -s dataset_1.
   fastq -p dataset_2.fastq -na 2 -m 6144 -nt 4
```

3.3 Compression support

HSRA supports the processing of input datasets compressed with Gzip (i.e., .gz extension) and BZip2 (i.e., .bz2 extension) codecs using the HSP library [11]. However, when considering compressed data that will be processed by Hadoop, it is important to understand whether the underlying compression format supports splitting, as many codecs need the whole input stream to uncompress successfully.

On the one hand, it is impossible to start reading at an arbitrary point in a Gzip file and therefore impossible for a map task to read its input split independently of the others. For this reason, Gzip does not support splitting and Hadoop will not split the gzipped input dataset. This will work, but at the expense of performance: a single map task will parse the whole input dataset, which prevents parallelism. In terms of performance, it would be probably better to first uncompress the gzipped dataset before storing it in HDFS. On the other hand, BZip2 does compression on blocks of data and later these compressed blocks can be decompressed independent of each other, so it does support splitting. Therefore, BZip2 is the recommended codec to use with Hadoop for best performance and parallelism. Note that if you are processing compressed input datasets, HSRA will not compress the SAM output files.

Finally, even if you are using uncompressed input datasets, Hadoop may benefit from compressing the intermediate output of the map phase in paired-end mode when using the reduce-side join approach (see next section). Since the map output is written to disk and transferred across the network to the reducer nodes, by using a fast compressor such as snappy, you may get performance gains simply because the volume of network data to transfer is reduced. HSRA supports the compression of the intermediate map output using the snappy codec, which can be configured by setting the COMPRESS_MAP_OUTPUT option, as shown next.

3.4 Configuration

HSRA can be configured by means of the *hsra.conf* file located at the *etc* directory. The main parameters are:

- MERGE_OUTPUT (boolean). Merge output files into a single SAM file stored in HDFS. The default value is false.
- DELETE_TEMP (boolean). Delete intermediate files created by Hadoop (if any). The default value is true.
- PAIRED_END_MAP_JOIN (boolean). Enable the use of a map-side join in paired-end scenarios when set to true, otherwise a reduce-side join is performed. The default value is true, which usually leads to a better performance.
- HDFS_BASE_PATH (string). Base path on HDFS where HSRA stores the output files as well as temporary intermediate files. The user running HSRA must have write permissions on this path. The default value is blank, which means to use the HDFS user's home directory.
- HDFS_BLOCK_REPLICATION (short). HDFS block replication factor for SAM output files. The default value is 1.
- INPUT_BUFFER_SIZE (integer). Buffer size in bytes used for input read operations. It should probably be a multiple of the hardware page size (e.g., 4096). The default value is 65536 bytes.
- OUTPUT_BUFFER_SIZE (integer). Buffer size in bytes used for output write operations. It should probably be a multiple of the hardware page size (e.g., 4096). The default value is 8192 bytes.
- PIPE_BUFFER_SIZE (integer). Size in bytes used for the internal buffers when creating named pipes in paired-end mode. The default value is 1048576 bytes. Maximum value is system dependent.
- MAP_HEAP_FACTOR (double). This setting manages the maximum JVM heap size (i.e., -Xmx) of map tasks, expressed as a factor of the HDFS block size. The default value is 4.0.

The following parameters are only significant in paired-end mode using a reduce-side join (i.e., PAIRED_END_MAP_JOIN=false). Most of them are considered advanced Hadoop parameters that should only be changed by experts users:

- TASK_HEAP_RATIO (double). This setting manages the maximum JVM heap size (i.e., -Xmx) of map/reduce tasks, expressed as a ratio of the total JVM memory. The default value is 0.8.
- COMPRESS_MAP_OUTPUT (boolean). Enable the compression of the intermediate map output using the snappy codec. It requires the snappy library installed on the system and the Hadoop native library (i.e., libhadoop) compiled with snappy support. The default value is false.
- MAP_IO_SORT_RATIO (double). This setting manages the size of the circular in-memory buffer used to store serialized key/value records emitted from map tasks (i.e., serialized map outputs), expressed as a ratio of the JVM heap size. The default value is 0.4.
- MAP_SORT_SPILL_PERCENT (double). The soft limit in the circular in-memory buffer used to store serialized map outputs. Once reached, a thread begins to spill contents to disk in the background. The default value is 0.95.
- MAP_IO_SORT_MB_METADATA_OVERHEAD (integer). When map output is being sorted, 16 bytes of metadata are added immediately before each key/value record. This parameter specifies the buffer space in MiBytes dedicated to store these metadata. It can be estimated by: (16 * N) * 1048576, being N calculated by dividing map input records by the number of map tasks. The default value is 75 (i.e., 75 MiB), which can be used for map tasks that emit up to 5 million of key/value records.
- SHUFFLE_PARALLEL_COPIES (integer). Number of parallel transfer run by reducers during the shuffle phase to fetch map outputs. The default value is 10.
- SHUFFLE_INPUT_BUFFER_PERCENT (double). Percentage of memory relative to the maximum JVM heap size that can be allocated to storing map outputs during the shuffle phase. The default value is 0.6.
- SHUFFLE_MERGE_PERCENT (double). Memory threshold for fetched map outputs before an in-memory merge is started, expressed as a percentage of memory allocated to storing map outputs (managed by the previous parameter). The default value is 0.8.
- REDUCE_INPUT_BUFFER_PERCENT (double). Percentage of memory relative to the memory allocated to storing map outputs in which map outputs can be retained during the reduce. The default value is 0.7.
- COMPLETED_MAPS_FOR_REDUCE_SLOWSTART (double). Fraction of the number of maps in the job which should be completed before reduces are scheduled. The default value is 0.8.

3.4.1 HISAT configuration

As a general rule, HSRA users are not required to do anything in particular to configure the HISAT aligner except setting the HISAT_HOME environmental variable as mentioned in Section 2. However, in order to properly interact with the underlying aligner, HSRA uses some default command-line options of HISAT, which do not usually change between

different releases. In the unlikely event that any of the options needed by HSRA was changed in future HISAT releases, HSRA provides the *hisat.conf* file located at the *etc* directory. The parameters included in this file together with their default values are:

- HISAT_EXEC=hisat2. Name of the HISAT executable file.
- HISAT_INDEX=-x. Argument to set the basename of the index for the reference genome.
- HISAT_FASTQ=-q. Argument that sets input reads in FASTQ format.
- HISAT_FASTA=-f. Argument that sets input reads in FASTA format.
- HISAT_NTHREADS=-p. Argument to set the number or parallel threads.
- HISAT_INPUT_UNPAIRED=-U. Argument to set the input file that contains unpaired reads.
- HISAT_INPUT_PAIRED_1=-1. Argument to set the first input file that contains paired reads.
- HISAT_INPUT_PAIRED_2=-2. Argument to set the second input file that contains paired reads.
- HISAT_PRINT_TIMES=-t. Argument that prints the wall-clock time required to load the index files and align the reads.
- HISAT_NO_HEADERS=–no-hd. Argument that removes any header lines (starting with @) in SAM output files.

4 Compilation

In case you need to recompile the HSRA distribution, the prerequisites are:

1. Make sure you have Java Development Kit (JDK) version 1.8 or above. You must set the JAVA_HOME environmental variable accordingly. On Linux, you can set this variable in your profile or your shell configuration files (e.g., .bashrc). Follow the instructions below:

```
[user@host ~]$ export JAVA_HOME=/path/to/jdk
```

2. Make sure you have a working C compiler for your system. If you use the GNU Compiler Collection (GCC), you are not required to do anything in particular. Otherwise, you must set the CC variable in the first line of the *Makefile.common* file located at *src/main/native* with the name of your compiler executable (e.g., icc for Intel compiler).
3. Make sure you have a working Apache Maven distribution version 3 or above
   - https://maven.apache.org/install.html

In order to build the JAR distribution, just execute the following Maven command from within the HSRA root directory:

```
[user@host hsra]$ mvn package
```

The first time you execute this command, Maven will download all the plugins and related dependencies it needs to fulfill the command. From a clean installation of Maven, this can take quite a while. If you execute the command again, Maven will now have what it needs, so it will be able to execute the command much more quickly.

5 Contact

HSRA has been developed in the Computer Architecture Group [13] at the University of A Coruña [14] by the following authors:

- Roberto R. Expósito: http://gac.udc.es/~rreye
- Jorge González-Domínguez: http://gac.udc.es/~jgonzalezd
- Juan Touriño: http://gac.udc.es/~juan

To report any question, bug, requirement or information about HSRA, feel free to contact us at [2].

References

[1] Roberto R. Expósito, Jorge González-Domínguez, and Juan Touriño. HSRA: Hadoop-based spliced read aligner for RNA sequencing data. *PLoS ONE*, 13(7):e0201483, 2018.

[2] HSRA webpage. http://hsra.dec.udc.es.

[3] Jeffrey Dean and Sanjay Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[4] Mihaela Pertea, Daehwan Kim, Geo M Pertea, Jeffrey T Leek, and Steven L Salzberg. Transcript-level expression analysis of RNA-seq experiments with HISAT, StringTie and Ballgown. *Nature Protocols*, 11(9):1650–1667, 2016.

[5] Daehwan Kim, Ben Langmead, and Steven L Salzberg. HISAT: a fast spliced aligner with low memory requirements. *Nature Methods*, 12(4):357–360, 2015.

[6] HISAT2 webpage. https://ccb.jhu.edu/software/hisat2.

[7] Apache Hadoop. http://hadoop.apache.org.

[8] GNU General Public License version 3 (GPLv3). https://www.gnu.org/licenses/gpl-3.0.en.html.

[9] Apache Hadoop Cluster Setup. http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/ClusterSetup.html.

[10] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The Hadoop distributed file system. In *Proceedings of the 26th IEEE Symposium on Mass Storage Systems and Technologies (MSST'10)*, pages 1–10, Incline Village, NV, USA, 2010.

[11] HSP: Hadoop Sequence Parser library for FASTQ/FASTA datasets. https://github.com/rreye/hsp.

[12] Bjørn Fjukstad and Lars Ailo Bongo. A review of scalable bioinformatics pipelines. *Data Science and Engineering*, 2(3):245–251, 2017.

[13] Computer Architecture Group. http://gac.udc.es/english.

[14] University of A Coruña. http://www.udc.gal/index.html?language=en.