Bioinformatics, 2017, 1–3
doi: 10.1093/bioinformatics/btx010
Advance Access Publication Date: 16 January 2017
Applications Note

Sequence analysis

FASTdoop: a versatile and efficient library for the input of FASTA and FASTQ files for MapReduce Hadoop bioinformatics applications

Umberto Ferraro Petrillo[1,*,†], Gianluca Roscigno[2,†], Giuseppe Cattaneo[2] and Raffaele Giancarlo[3]

[1]Dipartimento di Scienze Statistiche, Università di Roma—“La Sapienza”, 00185 Rome, Italy, [2]Dipartimento di Informatica, Università di Salerno, 84084 Fisciano (SA), Italy and [3]Dipartimento di Matematica ed Informatica, Università di Palermo, 90133 Palermo, Italy

*To whom correspondence should be addressed.
†The authors wish it to be known that, in their opinion, the first two authors should be regarded as Joint First Authors.
Associate Editor: John Hancock

Received on November 25, 2016; revised on December 22, 2016; editorial decision on January 9, 2017; accepted on January 10, 2017

Abstract

Summary: MapReduce Hadoop bioinformatics applications require the availability of special-purpose routines to manage the input of sequence files. Unfortunately, the Hadoop framework does not provide any built-in support for the most popular sequence file formats like FASTA or BAM. Moreover, the development of these routines is not easy, both because of the diversity of these formats and the need for managing efficiently sequence datasets that may count up to billions of characters. We present FASTdoop, a generic Hadoop library for the management of FASTA and FASTQ files. We show that, with respect to analogous input management routines that have appeared in the Literature, it offers versatility and efficiency. That is, it can handle collections of reads, with or without quality scores, as well as long genomic sequences while the existing routines concentrate mainly on NGS sequence data. Moreover, in the domain where a comparison is possible, the routines proposed here are faster than the available ones. In conclusion, FASTdoop is a much needed addition to Hadoop-BAM.
Availability and Implementation: The software and the datasets are available at http://www.di.unisa.it/FASTdoop/.
Contact: umberto.ferraro@uniroma1.it
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

MapReduce (Dean and Ghemawat, 2004), as supported by Apache Hadoop (White, 2015) is one of the most successful paradigms for the effective processing of Big Data in many industrial and scientific environments (for conciseness, we refer to *MapReduce* supported by Hadoop simply as Hadoop). Consequently, Hadoop is also gaining popularity in bioinformatics where, it supports a growing number of applications, reviewed in Cattaneo *et al.* (2016b).

It is also well known that bioinformatics software works by taking as input datasets in specific file formats, such as BAM (Li *et al.*, 2009), FASTQ (Cock *et al.*, 2010) and FASTA (http://blast.ncbi.nlm.nih.gov/blastcgihelp.shtml, accessed May 2016). Therefore, since the standard Hadoop plain text file input routines are not able to handle those formats, specialized software libraries have been developed, in particular for input purposes. Namely, Hadoop-BAM (Niemenmaa *et al.*, 2012) that can be used to read BAM, FASTQ and FASTA files and BioPig (Nordberg *et al.*, 2013), that offers routines which can be used for input of FASTQ and FASTA files.

Those routines have been designed to handle specific datasets: collections of reads coming from sequencing experiments. Therefore,

© The Author 2017. Published by Oxford University Press. All rights reserved. For Permissions, please e-mail: journals.permissions@oup.com

they lack versatility in the sense that they are unable to handle files containing long sequences, e.g. FASTA files whose content substantially differs from a collection of reads. This is a serious shortcoming since the applicability of Hadoop to bioinformatics tasks goes beyond datasets coming from sequencing, e.g. it can be successfully used to speed-up fundamental tasks such as sequence comparison, both based on alignments (Leo *et al.*, 2009) and alignment-free (Cattaneo *et al.*, 2016a). In addition, a test of the routines in Hadoop-BAM shows that they are not able to handle FASTA files larger than 1.5 MB (data not shown and available upon request) and a careful profiling of the BioPig routines reveals different sources of inefficiency, as the timing results presented in Section 3 make evident. For completeness, we mention the other two Hadoop frameworks supporting bioinformatics applications: SeqPig (Schumacher *et al.*, 2014) uses Hadoop-BAM for input management while Biodoop (Leo *et al.*, 2009) uses the standard Hadoop input routines.

FASTdoop, the software library proposed here, addresses the shortcomings of Hadoop-BAM and BioPig input routines by providing FASTA and FASTAQ input routines that are efficient and versatile.

2 Materials and Methods

The access to the input files of an application is managed by Hadoop through the implementation of a proper `InputFormat`. This class organizes a data source in smaller parts, called *input splits*, where each split is processed by a distinct map task. In turns, input splits represent a logical view of data that is typically organized in blocks under the HDFS distributed file system (Shvachko *et al.*, 2010). The `InputFormat` implementation has also the purpose of defining how to extract the <*key, value*> pairs from an input split that will be used as input to the map functions. At the moment, no explicit support is provided by Hadoop for handling FASTA files. Instead, the only ready-to-use class available to this end is `TextInputFormat`, a Java class that can be used by a map task to access the content of a generic text file, using a line-by-line logic.

FASTdoop includes two Java classes for FASTA files, these are: `FASTAshortInputFileFormat` (for conciseness, `Fshort`) and `FASTAlongInputFileFormat` (for conciseness, `Flong`). In addition, it provides `FASTQInputFileFormat` for FASTQ files (for conciseness, `FQ`).

The first routine for FASTA format handles files containing sets of short sequences, such as reads. `Fshort` takes as input a split and it generates <*key, value*> pairs that make that part of data accessible to map functions using it. In particular, it works by initially reading into a memory buffer the whole content of a Hadoop input split to be processed, with the help of some low-level byte-oriented functions provided by the framework. This acquisition is done in one burst, to reduce the overall loading time. Moreover, also all the (potential) characters that are found at the beginning of the subsequent split and that may be the terminal part of a short sequence starting in the current split are loaded. Once available in memory, this buffer is directly processed avoiding any delay due to costly memory copy operations.

When one needs to input a collection of very long sequences, where each sequence is stored in a different file, each of them is handled separately from the others via one instance of `Flong`. Such a strategy, at the level of map tasks, preserves data locality and avoids ambiguities regarding which piece of the input being processed belongs to which sequence. The end result is a saving in costly search operations over input splits, in order to get the identifier of a sequence whose split is being processed. Moreover, also `Flong` makes use of low-level byte-oriented functions as `Fshort`.

FASTQ files are handled by a variant of `Fshort` having a more complex parsing logic, which is needed for identifying the quality scores associated to each sequence in a file.

An example of usage of `Fshort` is reported in Listing 1. Since it has been implemented according to the standard Hadoop `InputFormat` interface, it can be used by just providing its class name by means of the `setInputFormatClass` method. From this point on, the processing of the chosen input FASTA file will be automatically and transparently handled by `Fshort`. The Supplementary Material gives additional details of how the routines work as well as examples of their uses.

Listing 1. An example of usage of `Fshort`. The code reported in the `run` method is the only required to use FASTdoop. The map function processes the input record without any output. The complete version of this code is given in the Supplementary Material.

```
public class Benchmark_Fshort extends Configured
    implements Tool {

 public int run(String[] args) throws Exception {
  ...
  String hdfsInputFile = args[0];
  Path hdfsInputPath = new Path(hdfsInputFile);
  Job job = Job.getInstance(getConf(), jobname);
  FileInputFormat.setInputPaths(job,
    hdfsInputPath);

  // Tell Hadoop to load input files with Fshort
  job.setInputFormatClass(FASTAshortInputFileFormat.class);
  ...
}

 public static class MyMapper extends Mapper<
    NullWritable, Record, NullWritable,
    NullWritable> {

 // Input sequences are automatically loaded
 // and exposed in the form (header, sequence)
  public void map(NullWritable key, Record value,
     Context context) throws IOException,
    InterruptedException {
   String header = value.getKey();
   String sequence = value.getValue();

   process(header, sequence);
  }
 }
}
```

3 Results and discussion

We highlight the results of the experiments conducted in order to assess the performance of the routines in FASTdoop, including a comparison with the other available routines (limited to the domain in which they work). Specifically, BioPig `FASTAInputFormat` and `FASTABlockInputFormat` readers (for conciseness, `BioPig-FIF` and `BioPig-FBIF`, respectively) and the Hadop-BAM reader (for conciseness, `HBAM`). For completeness, we also include in the comparison the standard Hadoop routine `TextInputFormat` (for conciseness, `H-TIF`), despite the fact that it makes data available to a map function using it one line at the time, thus making its use quite cumbersome, to the point that it is useless for files containing long sequences spanning multiple lines. Such a shortcoming is determined because, first of all, `TextInputFormat` splits input files around newline characters. This means that the same sequence spanning

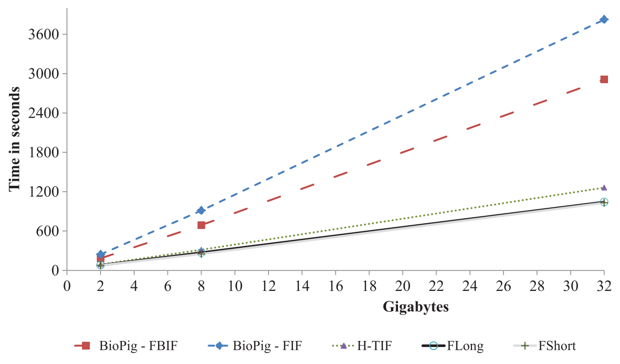

Fig. 1. A benchmarking of the input management routines considered in this study on FASTA files datasets. The integer tallies on the abscissa indicate the dataset size in GB, while the corresponding ordinate indicates the timing in seconds. It is worth remarking that `BioPig-FIF`, `BioPig-FBIF`, `HBAM` and `H-TIF` are not able to support the input of collections of long sequences, while HBAM turned out to have limitation (already mentioned) even on collections of short sequences

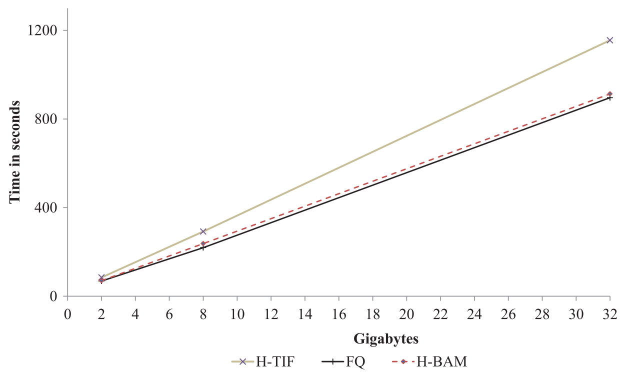

Fig. 2. A benchmarking of the input management routines considered in this study on FASTQ files datasets. The integer tallies on the abscissa indicate the dataset size in GB, while the corresponding ordinate indicates the timing in seconds. It is worth remarking that `BioPig-FIF` and `BioPig-FBIF` showed limitations in handling large collections of short sequences in FASTQ format

multiple lines would be virtually treated by `TextInputFormat` as two or more sequences. It would be up to the application to store in a temporary buffer all the subsequent lines found when scanning a sequence, and then merge them to obtain the original sequence. Similarly, it would be also in charge of the application to handle the scenario where a very long sequence (e.g. gigabytes) spans multiple HDFS blocks. In such a case, the application should examine not only the current HDFS block but also the following one (or the previous one) to look for the lines required to complete a sequence. Moreover, scanning the content of a long input file line-by-line is typically a very time consuming operation since the cost to be paid for reading one single line is often similar to the one paid for a much larger amount of data.

We extract datasets of various sizes for the experiments from the Illumina human genome dataset (http://sra.dnanexus.com/experiments/SRX016231). We use the concatenation of sequences in each dataset, in order to generate a corresponding new one of equal size for the test of `Flong`. Since the goal of the benchmarking is to evaluate the routines in FASTdoop, the Hadoop computational job using them is quite simple: it counts the number of occurrences of the letters $\{A, C, G, T, N\}$ in the input sequences, without producing any output. The hardware that we use is as follows:

A Hadoop *master* node with 2 quad core Intel Xeon processors and 32 GB of RAM;

A computer acting as Hadoop *slave* equipped with 64 GB of RAM and 4 AMD Opteron 6272 processors with 32 total cores.

The Hadoop version is 2.7.1, while the HDFS block size is set to 256 MB. The slave executes map tasks sequentially. Indeed, many concurrent instances of map tasks on the same slave can lead to I/O bottlenecks which would provide misleading results regarding the performance evaluation of our routines. The full set of results, in graphical form, i.e. the abscissa indicates the dataset size and the ordinate the time in seconds, are reported in Figures 1 and 2 for FASTA and FASTQ files, respectively. We include in Figure 1 also the experimentation regarding datasets containing long sequences. It is evident that the routines in FASTdoop represent an advancement of the state of the art both in terms of versatility and efficiency. In particular, we complement quite nicely `HBAM` since we are as efficient in handling FASTQ files but FASTdoop can also efficiently handle FASTA files that are not adequately supported by that library.

4 Conclusions

We have presented FASTdoop, a library allowing for the versatile and efficient input management of FASTA and FASTQ files for Hadoop applications. Based on a wide range of experiments, it can be concluded that it is an advancement in making Hadoop even more suited to handle big quantities of genomic data, not necessarily coming from the NGS pipeline.

Acknowledgements

We are grateful to the Consortium GARR for having made available a cutting edge OpenStack Virtual Datacenter which will turn out to be instrumental in the continuation of this work that tries to establish the impact that MapReduce can have on bioinformatics research.

Conflict of Interest: none declared.

References

Cattaneo,G. *et al.* (2016a) An effective extension of the applicability of alignment-free biological sequence comparison algorithms with Hadoop. *J. Supercomput*. doi: 10.1007/s11227-016-1835-3.

Cattaneo,G. *et al.* (2016b). MapReduce in Computational Biology – A Synopsis. In *Proceedings of the 11th Italian Workshop on Artificial Life and Evolutionary Computation*. Springer.

Cock,P.J. *et al.* (2010) The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Res.*, **38**, 1767–1771.

Dean,J. and Ghemawat,S. (2004). *MapReduce: Simplified Data Processing on Large Clusters*. In: *Proceedings of the Operating Systems Design and Implementation (OSDI)*, San Francisco, CA, pp. 137–150.

Leo,S. *et al.* (2009). Biodoop: bioinformatics on Hadoop. In *ICPPW'09. IEEE International Conference on Parallel Processing Workshops*, 2009, pp. 415–422.

Li,H. *et al.* (2009) The sequence alignment/map format and SAMtools. *Bioinformatics*, **25**, 2078–2079.

Niemenmaa,M. *et al.* (2012) Hadoop-BAM: directly manipulating next generation sequencing data in the cloud. *Bioinformatics*, **28**, 876–877.

Nordberg,H. *et al.* (2013) BioPig: a Hadoop-based analytic toolkit for large-scale sequence data. *Bioinformatics*, **29**, 3014–3019.

Schumacher,A. *et al.* (2014) SeqPig: simple and scalable scripting for large sequencing data sets in Hadoop. *Bioinformatics*, **30**, 119–120.

Shvachko,K. *et al.* (2010). The Hadoop distributed file system. In *IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, pp. 1–10.

White,T. (2015). *Hadoop: The Definitive Guide*, 4th edn. O'Reilly Media. Storage and Analysis at Internet Scale.