SeqPig: simple and scalable scripting for large sequencing data sets in Hadoop

André Schumacher, Luca Pireddu, Matti Niemenmaa, Aleksi Kallio, Eija Korpelainen, Gianluigi Zanetti and Keijo Heljanko

Abstract

Summary: Hadoop MapReduce-based approaches have become increasingly popular due to their scalability in processing large sequencing data sets. However, as these methods typically require in-depth expertise in Hadoop and Java, they are still out of reach of many bioinformaticians. In order to solve this problem, we have created SeqPig, a library and a collection of tools to manipulate, analyze and query sequencing data sets in a scalable and simple manner. SeqPig scripts use the Hadoop-based distributed scripting engine Apache Pig, which automatically parallelizes and distributes data processing tasks. We demonstrate SeqPig's scalability over many computing nodes and illustrate its use with example scripts.[1]

Availability and Implementation: Available under the open source MIT license at `http://sourceforge.net/projects/seqpig/`

Contact: `andre.schumacher@yahoo.com`

1 Introduction

Novel computational approaches are required to cope with the increasing data volumes of large-scale sequencing projects, since the growth in processing power and storage access speed is unable to keep pace with them [11, 4]. Several innovative tools and technologies have been proposed to tackle these challenges. Some are based on MapReduce, which is a distributed computing paradigm that is based on the idea of splitting input data into chunks which can be processed largely independently (via a *Map* function). Subresults can later be merged after grouping related subresults (by a *Reduce* function). MapReduce permits automatic parallelization and scalable data distribution across many computers. The most popular implementation available as open source software is Apache Hadoop, which also comes with its own distributed filesystem. The validity of

[1]This is the author's version of the article that appeared in *Bioinformatics* 30 (1): 119-120, 2014 and is available online via open access at `http://bioinformatics.oxfordjournals.org/content/30/1/119.abstract`.

Hadoop as a data processing platform is demonstrated by the level of adoption in major data-intensive companies, e.g., Twitter, Facebook and Amazon.

There are an increasing number of Hadoop-based tools for processing sequencing data [12], ranging from quality control [9] and alignment [3, 8] to SNP calling [3], variant annotation [7] and structural variant detection [13], including general purpose workflow management [10]. Note the recent publication of independent and complementary work in [6].

While Hadoop does simplify writing scalable, distributed software, it does not make it trivial. Such a task still requires specialized skills and a significant amount of work, particularly if the solution involves sequences of MapReduce jobs. This effort can be reduced significantly by using high-level tools such as Apache Pig, which implements an SQL-like scripting language that is automatically translated into a sequence of MapReduce jobs. Given its flexibility and simplicity for developing data processing pipelines, it is not surprising that a large fraction of computing jobs in contemporary Hadoop deployments originate from Apache Pig or similar high-level tools [2]. SeqPig brings the benefits of Apache Pig to sequencing data analysis. It allows users to integrate their own analysis components with existing MapReduce programs in order to create full NGS pipelines based on Hadoop.

2 Methods

SeqPig extends Pig with a number of features and functionalities specialized for processing sequencing data. Specifically, it provides: 1) data input and output components, 2) functions to access fields and transform data and 3) a collection of scripts for frequent tasks (e.g., pile-up, QC statistics).

Apache Pig provides an extension mechanism through the definition of new library functions, implemented in one of several supported programming languages (Java, Python, Ruby, JavaScript); these functions can then be called from Pig scripts. SeqPig uses this feature to augment the set of operators provided by plain Pig with a number of custom sequencing-specific functions.

SeqPig supports ad hoc (scripted and interactive) distributed manipulation and analysis of large sequencing datasets so that processing speed scales with the number of available computing nodes. It provides import and export functions for file formats commonly used for sequencing data: Fastq, Qseq, FASTA, SAM and BAM. These components, implemented with the help of Hadoop-BAM [5], allow the user to load and export sequencing data in the Pig environment. All available fields, such as BAM/SAM optional read attributes for example, can then be accessed and modified from within Pig. SeqPig also includes functions to access SAM flags, split reads by base (for computing base-level statistics), reverse-complement reads, calculate read reference positions in a mapping (for pileups, extracting SNP positions), and more. It comes packaged with scripts that calculate various statistics and manipulations on read data, which also serve as examples. The growing library of functions and scripts is documented in the SeqPig manual. Contributions from the community are welcome and

```
reads = LOAD 'in.qseq' USING QseqLoader();
STORE reads INTO 'out.fq' USING FastqStorer();
```

Figure 1: Converting Qseq into Fastq; the data set is simply read and then written using the appropriate load/store functions.

```
reads = LOAD 'in.fq' USING FastqLoader();
read_bases = FOREACH reads GENERATE
  UnalignedReadSplit(sequence, quality);
read_gc = FOREACH read_bases {
  only_gc = FILTER $0 BY readbase == 'G'
    OR readbase == 'C';
  GENERATE COUNT(only_gc) AS count;
}
gc_counts = FOREACH (GROUP read_gc BY count)
  GENERATE group AS gc_cnt, COUNT($1) AS cnt;
DUMP gc_counts;
```

Figure 2: Script that generates a histogram for GC content of reads. The script loads a Fastq file, splits each read into separate bases and for each read coordinate filters only bases that are either G or C. The filtered bases are then counted and counts are grouped. Finally, the script prints records which contain the GC count and the count of reads that have the given GC count.

encouraged. Figures 1 and 2 show script examples. For a more detailed list of features and more examples please see the project web site: `http://seqpig.sourceforge.net/`.

To evaluate SeqPig, we implemented a script which calculates most of the read quality statistics that are collected by the popular FastQC tool [1]. The script is included with the examples (`fast_fastqc.pig`). We ran a set of experiments which measured the speed-up gained by using SeqPig on Hadoop clusters of different sizes compared to using a single-node FastQC run. We used a set of Illumina reads as input (read length: 101 bases; file size: 61.4 GB; format: Fastq). Software versions were as follows: FastQC 0.10.1; Hadoop 1.0.4; Pig 0.11.1. All tests were run on nodes equipped with dual quad-core Intel Xeon CPUs @ 2.83 GHz, 16 GB of RAM and one 250 GB SATA disk available to Hadoop. Nodes are connected via Gigabit Ethernet. FastQC read its data from a high-performance shared parallel file system by DDN. SeqPig used the Hadoop file system (HDFS) which uses each node's local disk drive.

We first ran five different SeqPig read statistics for a different number of computing nodes: the sample distribution of a) the average base quality of the reads; b) the length of reads; c) bases by position inside the reads; d) the GC contents of reads. Finally, we combined them into a single script. Each of the executions results in a single MapReduce job and thus a single scan through the data. All runs were repeated three times and averaged (deviation from average $<$ 7%). From Figure 3 one can see that it is possible to achieve a significant speed-up by exploiting the parallelism in read and base statistics computation using

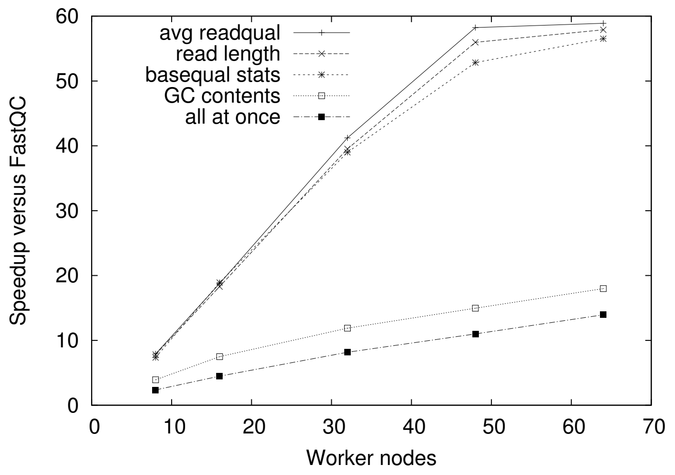

Figure 3: Results of an experiment with an input file of 61.4 GB and a different number of Hadoop worker nodes. The script does not currently implement all FastQC statistics (we expect the missing ones to scale similarly in SeqPig), whereas the per-cycle quality distribution is not computed by FastQC.

Hadoop. Further, the total runtime of the script that computes all statistics is mostly determined by the slowest of the individual ones, since the complete script is compiled into a single Map-only job. A different observation is that for most of the statistics computed we are able to achieve a close to linear speedup compared to FastQC until 48 nodes. We assume that the levelling off is due to the Hadoop job overhead eventually dominating over speedup due to parallelization, depending on input file size.

SeqPig enables simple and scalable manipulation and analysis of sequencing data on the Hadoop platform. At CRS4 SeqPig is already used routinely for several steps in the production workflow; in addition, it has been successfully used for ad hoc investigations into data quality issues, comparison of alignment tools, and reformatting and packaging data. We have also tested SeqPig on Amazon's Elastic MapReduce service, where users may rent computing time on the cloud to run their SeqPig scripts and even share their S3 storage buckets with other cloud-enabled software. Instructions are provided in the supplementary material.

Acknowledgements This work was supported by the Cloud Software and D2I Programs of the Finnish Strategic Centre for Science, Technology and Innovation DIGILE; Academy of Finland [139402]; the Sardinian (Italy) Regional Grant [L7-2010/COBIK]; COST Action BM1006 Next Generation Sequencing Data Analysis Network, SeqAhead. Computational resources were provided by CRS4 and ELIXIR node hosted at CSCIT Center for Science.

References

[1] S. Andrews. Fastqc. a quality control tool for high throughput sequence data. *http://www.bioinformatics.babraham.ac.uk/projects/fastqc*, 2010.

[2] Y. Chen, S. Alspaugh, et al. Interactive analytical processing in big data systems: A cross-industry study of MapReduce workloads. In *Proceedings of the VLDB Endowment*, volume 5, pages 1802–1813, Aug. 2012.

[3] B. Langmead, M. Schatz, et al. Searching for SNPs with cloud computing. *Genome Biology*, 10(11):R134, 2009.

[4] V. Marx. Biology: The big challenges of big data. *Nature*, 498:255–260, June 2013.

[5] M. Niemenmaa, A. Kallio, et al. Hadoop-BAM: Directly manipulating next generation sequencing data in the cloud. *Bioinformatics*, 28(6):876–877, 2012.

[6] H. Nordberg, K. Bhatia, et al. Biopig: a Hadoop-based analytic toolkit for large-scale sequence data. *Bioinformatics*, 2013. Epub ahead of print.

[7] B. O'Connor, B. Merriman, et al. SeqWare Query Engine: storing and searching sequence data in the cloud. *BMC Bioinformatics*, 11(Suppl 12):S2, 2010.

[8] L. Pireddu, S. Leo, et al. SEAL: a distributed short read mapping and duplicate removal tool. *Bioinformatics*, 27(15):2159–2160, 2011.

[9] T. Robinson, S. Killcoyne, et al. SAMQA: Error classification and validation of high-throughput sequenced read data. *BMC Genomics*, 12(1):419, 2011.

[10] S. Schönherr, L. Forer, et al. Cloudgene: A graphical execution platform for MapReduce programs on private and public clouds. *BMC Bioinformatics*, 13(1), 2012.

[11] L. Stein. The case for cloud computing in genome informatics. *Genome Biology*, 11(5):207, 2010.

[12] R. Taylor. An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics. *BMC Bioinformatics*, 11(Suppl 12):S1, 2010.

[13] C. W. Whelan, J. Tyner, et al. Cloudbreak: Accurate and Scalable Genomic Structural Variation Detection in the Cloud with MapReduce. *arXiv:1307.2331*, 2013.