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

André Schumacher1,2,†, Luca Pireddu3, Matti Niemenmaa1, Aleksi Kallio4, Eija Korpelainen4, Gianluigi Zanetti3 and Keijo Heljanko1

1Aalto University School of Science and Helsinki Institute for Information Technology HIIT, Finland
2International Computer Science Institute, Berkeley, CA, USA
3CRS4 — Center for Advanced Studies, Research and Development in Sardinia, Italy
4CSC — IT Center for Science, Finland

Associate Editor: Prof. Alfonso Valencia

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.

Availability and Implementation: Available under the open source MIT license at http://sourceforge.net/projects/seqpig/
Contact: andre.schumacher@yahoo.com
Supplementary information: Instructions and examples for SeqPig

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 (Stein, 2010; Marx, 2013). 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 Hadoop as a data processing platform is demonstrated by the level of adoption in major data-intensive companies, e.g., Twitter, Facebook and Amazon.

Motivated by the potential scalability and throughput offered by Hadoop, there are an increasing number of Hadoop-based tools for processing sequencing data (Taylor, 2010), ranging from quality control (Robinson et al., 2011) and alignment (Langmead et al., 2009; Pireddu et al., 2011) to SNP calling (Langmead et al., 2009), variant annotation (O’Connor et al., 2010) and structural variant detection (Whelan et al., 2013), including also general purpose workflow management (Schönherr et al., 2012).

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 (Chen et al., 2012). 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 (Niemenmaa et al., 2012), 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

© The Author(s) 2013. Published by Oxford University Press.
This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0/), which permits unrestricted reuse, distribution, and reproduction in any medium, provided the original work is properly cited.
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: We acknowledge support by the Cloud Software and D2I Programs of the Finnish Strategic Centre for Science, Technology and Innovation DIGILE, Academy of Finland, and by the Sardinian (Italy) Regional Grant [L7-2010/COBIK]. Computational resources were provided by CRS4 and ELIXIR node hosted at CSC – IT Center for Science. The collaboration originated thanks to meetings organized by SeqAhead COST Action BM1006.

REFERENCES


