BIOINFORMATICS APPLICATIONS NOTE Vol. 27 no. 15 2011, pages 2159–2160
doi:10.1093/bioinformatics/btr325

Sequence analysis

Advance Access publication June 22, 2011

SEAL: a distributed short read mapping and duplicate removal tool

Luca Pireddu*, Simone Leo and Gianluigi Zanetti
CRS4, Polaris, Ed. 1, I-09010 Pula, Italy

Associate Editor: Alex Bateman

ABSTRACT

Summary: SEAL is a scalable tool for short read pair mapping and duplicate removal. It computes mappings that are consistent with those produced by BWA and removes duplicates according to the same criteria employed by Picard MarkDuplicates. On a 16-node Hadoop cluster, it is capable of processing about 13 GB per hour in map+rmdup mode, while reaching a throughput of 19 GB per hour in mapping-only mode.

Availability: SEAL is available online at http://biodoop-seal.sourceforge.net/.

Contact: luca.pireddu@crs4.it

Received on March 7, 2011; revised on May 9, 2011; accepted on May 26, 2011

1 INTRODUCTION

Deep sequencing experiments read billions of short fragments of DNA, and their throughput is steadily increasing (Metzker, 2010). These reads need to be post-processed after sequencing to prepare the data for further analysis, which implies that the computational steps need to scale their throughput to follow the trend in sequencing technology. Such high data rates imply the need for a distributed architecture that can scale with the number of computational nodes.

Typical post-processing steps include sequence alignment, which is a fundamental step in nearly all applications of deep sequencing technologies, and duplicate read removal, which is a major concern for Illumina sequencing (Kozarewa *et al.*, 2009). The pressure for better and faster tools has recently given rise to the development of new alignment algorithms that outperform traditional ones in terms of both speed and accuracy (Li and Homer, 2010). Distributed alignment tools have also been created, with Crossbow (Langmead *et al.*, 2009a) as one of the most prominent examples. However, Crossbow is based on Bowtie (Langmead *et al.*, 2009b), and thus does not currently support gapped alignment, an important feature for many applications (Li and Homer, 2010).

In this work we describe SEAL, a new distributed alignment tool that combines BWA (Li and Durbin, 2009) with duplicate read detection and removal. SEAL harnesses the Hadoop MapReduce framework (http://hadoop.apache.org) to efficiently distribute I/O and computation across cluster nodes and to guarantee reliability by resisting node failures and transient events such as peaks in cluster load. In its current form, SEAL specializes in the pair-end alignment of sequences read by Illumina sequencing machines. SEAL uses a version of the original BWA code base (version 0.5.8c) that has been refactored to be modular and extended to use shared memory to significantly improve performance on multicore systems.

*To whom correspondence should be addressed.

2 METHODS

SEAL is currently structured in two applications that work in sequence: PairReadsQseq and Seqal. PairReadsQseq is a utility that converts the qseq files (Illumina, Inc., 2009) produced by Illumina sequencing machines into our prq file format that places entire read pairs on a single line. Seqal is the core that implements read alignment and optionally also performs duplicate read removal following the same duplicate criteria used by Picard MarkDuplicates (http://picard.sourceforge.net). Both applications implement MapReduce algorithms (Dean and Ghemawat, 2004) which run on the Hadoop framework.

MapReduce and Hadoop: MapReduce is a programming model prescribing that an algorithm be formed by two distinct functions: *map* and *reduce*. The map function receives one input record and outputs one or more key-value pairs; the reduce function receives a single key and a list of all the values that are associated to that key. Hadoop is the most widespread implementation of MapReduce.

Pairing reads in PairReadsQseq: PairReadsQseq groups mate pairs from qseq data files into the same record, producing prq files where each line consists of five tab-separated fields: id; sequence and ASCII-encoded base qualities for read 1 and 2.

Read alignment and duplicates removal in Seqal: SEAL's second MapReduce application, Seqal, takes input pairs in the prq format and produces mapped reads in SAM format (Li *et al.*, 2009). The read alignment is implemented in the map function. Rather than implementing a read aligner from scratch, we integrated BWA (Li and Durbin, 2009) into our tool. We refactored its functionality into a new library, libbwa, which allows us to use much of the functionality of BWA programmatically. Although it is written in C, it provides a high-level Python interface. To take advantage of this feature, the Seqal mapper is written in Python, and integrates into the Hadoop framework using Pydoop (Leo and Zanetti, 2010).

For each pair of reads, the aligner produces a pair of alignment records. The user can choose to filter these by whether or not the read is mapped and by mapping quality. Then, the reads may be directly output to SAM files, or put through a reduce phase where duplicates are removed; the choice is made through a command line option.

Like Picard MarkDuplicates, Seqal identifies duplicate reads by noting that they are likely to map to the same reference coordinates. The specific criteria we use defines two pairs as duplicates if their alignment coordinates are identical, both for their first and second reads. Likewise, lone reads are considered duplicates if they are aligned to the same position. When a set of duplicate pairs is found, only the one with the highest average base quality is kept; the rest are discarded as duplicates. Moreover, when a lone read is aligned to the same position as a paired read, the lone one is discarded. If, on the other hand, only lone reads are found at a specific position then, as for pairs, only the one with the highest average base quality is kept.

2.1 Evaluation

Correctness: we verified the correctness of SEAL by performing the alignment of the 5M dataset (Table 1) to the UCSC HG18 reference genome

Table 1. SEAL evaluation: input datasets

Dataset	No. of lanes	No. of pairs	Size (GB)	Read length
5M	0	5.0×10^6	2.3	91
DS1	1	1.2×10^8	51	100
DS3	3	3.3×10^8	147	100
DS8	8	9.2×10^8	406	100

The 5M dataset consists of the first 5M pairs from run id ERR020229 of the 1000 Genomes Project (Durbin *et al.*, 2010). The three DS datasets are from a production sequencing run on an Illumina HiSeq 2000.

Table 2. Comparison of running time in hours between BWA on a single node with 8 cores and SEAL running on 32 nodes without duplicates removal

Dataset	BWA time (h, 1 node)	SEAL time (h, 32 nodes)
5M	0.49	0.04
DS1	11.26[a]	0.63
DS3	32.39[a]	1.72
DS8	89.35[a]	4.78

Note that the SEAL running time includes qseq to prq format conversion. [a]Time is predicted as a linear extrapolation of the throughput observed on the 5M dataset.

(Fujita *et al.*, 2010) with both SEAL and BWA ver. 0.5.8c and then comparing their output. With BWA, we ran bwa aln and bwa sampe, while with SEAL we ran the PairReadsQseq and Seqal applications.

The result was identical for 99.5% of the reads. The remaining 0.5% had slightly different map quality scores (mapq), while the mapping coordinates were identical for all but two reads. Both of the latter two cases had multiple best hits but resulted in different alignment choices probably due to insert size statistics, in turn due to the particular input read batch. Slight differences in mapq scores are expected because their calculation takes into account the insert size statistics, which are calculated on sample windows on the input stream of sequences. Since the sample windows seen by the command line version of BWA and SEAL are different for each read, a slight change in the mapq value is expected. To verify this hypothesis, we ran BWA with varying input datasets while keeping 3000 of those reads that produced mapq variations in the original experiment. We observed that the mapq values for those reads varied between runs.

Speed and scalability: we tested SEAL with varying input size (DS datasets from Table 1) and cluster size (16, 32, 64 and 96 nodes). Each node is equipped with dual quad-core Intel Xeon CPUs @ 2.83 GHz, 16 GB of RAM, two 250 GB SATA disks, one of which is used for Hadoop storage. Nodes are connected via Gigabit Ethernet. For each cluster size, we allocated a Hadoop cluster (ver. 0.20.2) and copied the input data and a tarball of the indexed reference sequence onto the Hadoop file system. The SEAL application was run on all the DS datasets in both alignment-only and alignment plus remove duplicate modes. The runs were repeated three times, with the exception of DS8 which was run only once. The runtimes for the different datasets are reported in Table 2, while the throughput is shown in Figure 1.

Looking at Figure 1, we see that SEAL is generally capable of throughput levels comparable to single-node operation, meaning that the application and Hadoop keep the distribution overhead to a minimum. As the cluster size increases, we would ideally see a constant throughput per node, giving a linear increase in overall throughput. In practice, when the input is too small with respect to the computational capacity, nodes are often underutilized. Therefore, the throughput per node with DS1 at 96 nodes is much lower than the other configurations. On the other hand, we see that SEAL is capable of utilizing available resources efficiently when more data are available, although while scaling up from 64 to 96 nodes, the system achieved better throughput on the small DS3 dataset as opposed to the larger DS8. We suspect this is due to network congestion, which can be alleviated by informing Hadoop about the cluster network topology.

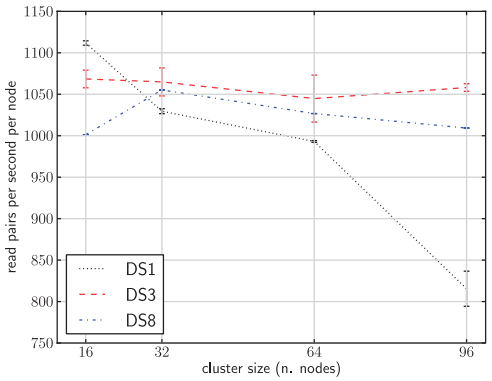

Fig. 1. Throughput per node of the entire SEAL workflow: finding paired reads in different files; computing the alignment; and removing duplicate reads. An ideal system would produce a flat line, scaling perfectly as the cluster size grows. The three datasets used are described in Table 1. By comparison, a single-node workflow we wrote for testing—performing the same work as SEAL but using the standard multithreaded BWA and Picard—reaches a throughput of ∼1100 pairs/s on the 5M dataset.

SEAL is able to achieve such scalability rates principally thanks to libbwa's efficient use of memory. In fact, libbwa stores the reference in shared memory, allowing all libbwa instances running on the same system to share the same memory space. In practical terms, this feature makes it possible to run in parallel 8 alignments on a system with 8 cores and 16 GB of memory, fully operating in parallel. While BWA does have a multithreaded mode of operation, it only applies to the bwa aln step. On the contrary, SEAL is able to parallelize all steps in the alignment.

ACKNOWLEDGEMENTS

We would like to thank our colleagues R. Berutti, M. Muggiri, C. Podda and F. Reinier for their feedback and technical support.

Conflict of Interest: none declared.

REFERENCES

Dean,J. and Ghemawat,S. (2004) MapReduce: simplified data processing on large clusters. In *OSDI '04: 6th Symposium on Operating Systems Design and Impl.*, USENIX Association.

Durbin,R.M. *et al.* (2010) A map of human genome variation from population-scale sequencing. *Nature*, **467**, 1061–1073.

Fujita,P.A. *et al.* (2010) The UCSC Genome Browser database: update 2011. *Nucleic Acids Res.*, **39** (Suppl. 1) D876–D882.

Illumina, Inc. (2009) *Sequencing Analysis Software User Guide For Pipeline Version 1.4 and CASAVA Version 1.0*. Illumina.

Kozarewa,I. *et al.* (2009) Amplification-free Illumina sequencing-library preparation facilitates improved mapping and assembly of (G+C)-biased genomes. *Nat. Methods*, **6**, 291–295.

Langmead,B. *et al.* (2009a) Searching for SNPs with cloud computing. *Genome Biol.*, **10**, 134.

Langmead,B. *et al.* (2009b) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol.*, **10**, 25.

Leo,S. and Zanetti,G. (2010) Pydoop: a Python MapReduce and HDFS API for Hadoop. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, ACM, New York, NY, USA, pp. 819–825.

Li,H. and Durbin,R. (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, **25**, 1754–1760.

Li,H. and Homer,N. (2010) A survey of sequence alignment algorithms for next-generation sequencing. *Brief. Bioinformatics*, **11**, 473–483.

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

Metzker,M.L. (2010) Sequencing technologies — the next generation. *Nat. Rev. Genet.*, **11**, 31–46.