Bioinformatics Advance Access published February 2, 2012

Hadoop-BAM: Directly manipulating next generation sequencing data in the cloud

Matti Niemenmaa [1,*], Aleksi Kallio [2], André Schumacher [1], Petri Klemelä [2], Eija Korpelainen [2], and Keijo Heljanko [1]

[1]Aalto University School of Science, Department of Information and Computer Science
[2]CSC — IT Center for Science

Associate Editor: Dr. Alex Bateman

ABSTRACT
Summary: Hadoop-BAM is a novel library for the scalable manipulation of aligned next generation sequencing data in the Hadoop distributed computing framework. It acts as an integration layer between analysis applications and BAM files that are processed using Hadoop. Hadoop-BAM solves the issues related to BAM data access by presenting a convenient API for implementing map and reduce functions that can directly operate on BAM records. It builds on top of the Picard SAM JDK, so tools that rely on the Picard API are expected to be easily convertible to support large scale distributed processing. In this paper we demonstrate the use of Hadoop-BAM by building a coverage summarizing tool for the Chipster genome browser. Our results show that Hadoop offers good scalability, and one should avoid moving data in and out of Hadoop between analysis steps.
Availability: Available under the open-source MIT license at
`http://sourceforge.net/projects/hadoop-bam/`
Contact: matti.niemenmaa@aalto.fi

1 INTRODUCTION

Next generation sequencing (NGS) technologies provide unprecedented opportunities for life science research. In order to exploit this potential to its full extent, new computational approaches are needed for the efficient processing of large data sets. Nearly all NGS applications rely on sequence alignment as the first analysis step. The alignment data is commonly stored in the standardized, compact and indexed BAM (Binary Alignment/Map) format (Li *et al.*, 2009), which is then used for further analysis such as SNP genotyping, peak calling, or detecting differential gene expression. As data sizes increase more rapidly than processing power and disk-read speed, many of these bioinformatics tasks have been ported to utilize the map-reduce distributed processing framework (Taylor, 2010). Existing solutions, such as GATK (McKenna *et al.*, 2010), SeqWare Query Engine (O'Connor *et al.*, 2010) and Seal (Pireddu *et al.*, 2011), provide useful parts for NGS data analysis pipelines. However, they do not allow efficient parallel access to BAM files.

Map-reduce is a distributed computing paradigm that has been designed for processing collections of relatively independent data items, and is therefore well suited for sequencing reads (Dean and Ghemawat, 2008). It divides data between processing nodes by splitting the files into chunks, which are then processed separately. The user has to write map and reduce functions, where the map function does the actual processing of a chunk, and the reduce function combines partial results. The most popular open source implementation of map-reduce is Apache Hadoop (White, 2009).

BAM files are conceptually a good fit for map-reduce style chunk processing, but their low level structure hinders adoption. Typically map-reduce jobs process data chunks in line-based text format, where identifying entries is simple as line boundaries are denoted by newline characters. Detecting entry boundaries and accessing the binary content of (compressed) BAM files, however, is nontrivial. On the other hand, using plain Hadoop with text-based SAM files results in several times greater disk and network loads. Text formats also complicate the pipeline as data is typically stored in BAM files. We developed the Hadoop-BAM Java library to act as an integration layer between analysis applications and BAM files stored in the Hadoop Distributed File System (HDFS).

2 METHODS

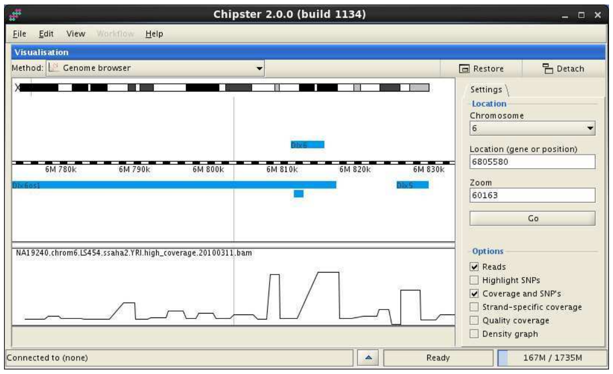
Hadoop-BAM solves the issues related to BAM splitting, presenting a convenient API for implementing map and reduce functions for Hadoop. The library supports two modes of access to BAM files. The first mode relies on a precomputed index that maps byte offsets to BAM records and thus allows random access, which is required to process chunks that can result from Hadoop splitting the BAM data arbitrarily. The second mode does not use an index and instead relies on a two-level detection routine. The higher level locates boundaries between compressed blocks via BGZF magic numbers, while the lower level detects BAM block boundaries via redundancies in the BAM file format. For details we refer to the Supplementary material.

The library exposes a Picard compatible Java API to programmers. Hence, Hadoop code can be written without considering the issues of BGZF compression, block boundary detection, BAM record boundary detection, or parsing of raw binary data. Tools developed upon the Picard API can be easily converted to support large scale distributed computing with Hadoop-BAM.

2.1 Evaluation

To demonstrate the library, we use it for calculating coverage summaries for the Chipster genome browser. Chipster is a biologist-friendly analysis software for high-throughput data (Kallio *et al.*, 2011), and its genome browser allows users to zoom smoothly from whole chromosome to nucleotide

*corresponding author



Method: Genome browser

Chromosome: 6

Location (gene or position): 6805580

Zoom: 60163

NA19240.chrom6.LS454.ssaha2.YRI.high_coverage.20100311.bam

☑ Reads
☐ Highlight SNPs
☑ Coverage and SNP's
☐ Strand-specific coverage
☐ Quality coverage
☐ Density graph

Connected to (none)　Ready　167M / 1735M

Fig. 1. Chipster genome browser using preprocessed data to show an interactive high level overview of coverage profile.

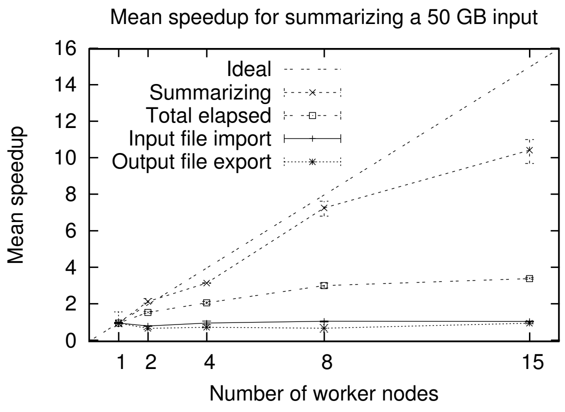


Fig. 2. Mean speedups for summarizing a 50 gigabyte BAM file with Hadoop, using heuristic splitting. Due to the cluster usage policy, the maximum number of parallel worker nodes was restricted to 15.

level. Good interactive performance with large BAM files is achieved by precomputing summary files, which are used to create zoomed out views (see Figure 1).

Implementing summarizing is simple, because Hadoop-BAM allows developers to treat BAM files as Hadoop input/output formats, which includes the provision of a custom partitioner for the input data. The library further extends Hadoop to offer SAMRecord from the Picard toolbox as a map-reduce value type. In essence, the task is to extract the genomic coordinates from the given BAM file, sort the resulting records first by their center point, and for each consecutive group of records of size at most N, output a summarized record containing mean position and group size.

The tool was implemented on top of Hadoop version 0.20.2, which was the latest stable version as of writing. Intermediately data was compressed via hadoop-lzo. For benchmarking, we relied on a test cluster with 112 nodes, each of which has two six-core AMD Opteron 2435 CPUs with a clock speed of 2.6 GHz and 250 GB of local disk space, and InfiniBand interconnect. A 50 GB BAM file containing whole genome sequencing data from the 1000 Genomes Project was summarized into groups of size 2^k for $k \in \{1, 2, \ldots, 16\}$ during a single map-reduce run.

Total execution time is already well under an hour with eight worker nodes. This is very reasonable for a 50 GB data set. As shown in Figure 2, the map-reduce job scales well up to about eight worker nodes, after which scaling worsens. This also has a significant effect on the total time: starting at the four worker mark, the job actually takes less time than the file transfers. As the import and export of data requires much time, we conclude that when designing Hadoop based pipelines, one should avoid moving data in and out of Hadoop between analysis steps. Performance is also bound by the interconnect network. This result indicates that BAM, as a binary and compressed format, is suitable for large scale NGS data analysis in the cloud. Using SAM or another text format would greatly reduce performance, as there would be far more data to transfer. All in all, compact formats are good not only for storage, but also for distributed processing with map-reduce.

3 DISCUSSION

To conclude, we presented how the combination of a compact data format such as BAM and a powerful distributed framework Hadoop can be used to efficiently process large NGS data sets. The Hadoop-BAM library provides an easy-to-use interface for their integration by resolving the incompatibilities these two technologies have. We predict that similar integration efforts will become common when cloud computing is taken into wider use in NGS data analysis. While our use case consisted of coverage calculations, it is important to note that Hadoop-BAM can be used for virtually any analysis task based on BAM files, ranging from variant detection to peak calling.

In order to make Hadoop-BAM more accessible, we are currently evaluating simpler and higher-level Hadoop-based query languages for working with BAM files. Examples of such include Apache Pig (Olston *et al.*, 2008) and Hive (Thusoo *et al.*, 2010). We have also developed a command line interface and are extending it to provide Samtools-like functionality.

Funding: This work was supported by the Cloud Software Program of Finnish Strategic Centre for Science, Technology and Innovation TIVIT, and Academy of Finland (#139402).

REFERENCES

Dean, J. and Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Communications of the ACM (CACM)*, **51**(1), 107–113.

Kallio, A., Tuimala, J., Hupponen, T., Klemelä, P., Gentile, M., Scheinin, I., Koski, M., Kaki, J., and Korpelainen, E. (2011). Chipster: user-friendly analysis software for microarray and other high-throughput data. *BMC Genomics*, **12**(1), 507.

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., Durbin, R., and 1000 Genome Project Data Processing Subgroup (2009). The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, **25**(16), 2078–2079.

McKenna, A., Hanna, M., Banks, E., Sivachenko, A., Cibulskis, K., Kernytsky, A., Garimella, K., Altshuler, D., Gabriel, S., Daly, M., and DePristo, M. A. (2010). The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Research*, **20**(9), 1297–1303.

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

Olston, C., Reed, B., Srivastava, U., Kumar, R., and Tomkins, A. (2008). Pig latin: a not-so-foreign language for data processing. In J. T.-L. Wang, editor, *SIGMOD Conference*, pages 1099–1110. ACM.

Pireddu, L., Leo, S., and Zanetti, G. (2011). SEAL: A distributed short read mapping and duplicate removal tool. *Bioinformatics*, pages 1–2. Advance Access published June 22, 2011.

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

Thusoo, A., Sarma, J. S., Jain, N., Shao, Z., Chakka, P., 0002, N. Z., Anthony, S., Liu, H., and Murthy, R. (2010). Hive – a petabyte scale data warehouse using Hadoop. In F. Li, M. M. Moro, S. Ghandeharizadeh, J. R. Haritsa, G. Weikum, M. J. Carey, F. Casati, E. Y. Chang, I. Manolescu, S. Mehrotra, U. Dayal, and V. J. Tsotras, editors, *ICDE*, pages 996–1005. IEEE.

White, T. (2009). *Hadoop - The Definitive Guide: MapReduce for the Cloud*. O'Reilly.