Bioinformatics, 35(12), 2019, 2156–2158
doi: 10.1093/bioinformatics/bty940
Advance Access Publication Date: 14 November 2018
Applications Note

Genome analysis

SeQuiLa: an elastic, fast and scalable SQL-oriented solution for processing and querying genomic intervals

Marek Wiewiórka[1,†], Anna Leśniewska[2,†], Agnieszka Szmurło[1], Kacper Stępień[2], Mateusz Borowiak[2], Michał Okoniewski[3] and Tomasz Gambin[1,*]

[1]Institute of Computer Science, Warsaw University of Technology, Warsaw 00-665, Poland, [2]Department of Computer Science, Poznan University of Technology, Poznań 60-965, Poland and [3]Scientific IT Services, ETH Zurich, Zürich 8092, Switzerland

*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 May 7, 2018; revised on July 23, 2018; editorial decision on November 11, 2018; accepted on November 13, 2018

Abstract

Summary: Efficient processing of large-scale genomic datasets has recently become possible due to the application of 'big data' technologies in bioinformatics pipelines. We present SeQuiLa—a distributed, ANSI SQL-compliant solution for speedy querying and processing of genomic intervals that is available as an Apache Spark package. Proposed range join strategy is significantly (∼22×) faster than the default Apache Spark implementation and outperforms other state-of-the-art tools for genomic intervals processing.
Availability and implementation: The project is available at http://biodatageeks.org/sequila/.
Contact: tgambin@ii.pw.edu.pl
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Analyses requiring intersection of genomic intervals as defined in Layer (2013) are supported by several reported software tools, including featureCounts (Liao, 2014), samtools (Li, 2009) and GenomicRanges (Lawrence, 2013). Despite their popularity and ease-of-use, they suffer from similar performance limitations, thereby making genome-wide analyses infeasible. On the other hand, recently, there has been an outburst of scalable solutions for genomics, including sparkhit (Huang, 2018), ADAM (Massie *et al.*, 2013) and the latest GATK version leveraging Apache Spark execution engine. Moreover, adapting relational algebra principles in a form of a declarative Structured Query Language (SQL) interface for querying genomic datasets is a novel approach (Kozanitis, 2014; Masseroli, 2015). A proof-of-concept solution that combines big data techniques and SQL interface for handling large-scale interval queries was proposed in GenAp (Kozanitis and Patterson, 2016). This approach, however, requires modifications in the Apache Spark source code, making this tool hard to maintain and extend; it also discards low-level optimizations resulting in a suboptimal performance. Furthermore, as a consequence of introducing the new keywords, it is effectively non-compliant with ANSI SQL standards, what may cause integration difficulties. To address the aforementioned issues, we have developed a SeQuiLa Apache Spark package which is a distributed, SQL-compliant solution, implementing fast range join computations between two tables, representing genomic intervals.

2 Materials and methods

2.1 Algorithm and implementation

Consider datasets $s1$ and $s2$, storing genomic intervals such as $|s1| < |s2|$. The main idea of the algorithm is to transform $s1$ into a

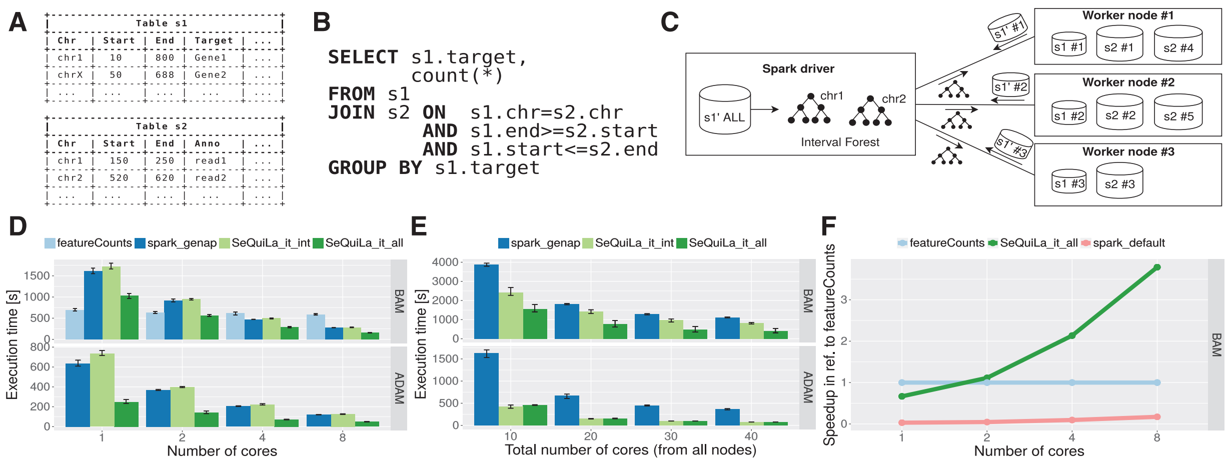

Table s1				
Chr	**Start**	**End**	**Target**	...
chr1	10	800	Gene1	...
chrX	50	688	Gene2	...
...	...	...	...	...

Table s2				
Chr	**Start**	**End**	**Anno**	...
chr1	150	250	read1	...
chr2	520	620	read2	...
...	...	...	...	...

```
SELECT s1.target,
       count(*)
FROM s1
JOIN s2 ON  s1.chr=s2.chr
        AND s1.end>=s2.start
        AND s1.start<=s2.end
GROUP BY s1.target
```

Fig. 1. Example of datasets' structure (*s*1, *s*2), including required columns, i.e. chr, start, end (**A**) and a sample SQL query (**B**). Broadcasting interval forest to worker nodes (**C**). Performance comparison of featureCounts (state-of-the-art single node solution) against GenAp (the only available distributed solution) and SeQuiLa on single node (**D**), and on a cluster (**E**). Speedup characteristics of SeQuiLa and default join implementation in Spark with featureCounts as a baseline (**F**)

broadcastable structure of an interval forest [a hash map of interval trees (Cormen *et al.*, 2009), each representing one chromosome]. The intervals from *s*2 can be efficiently intersected with the interval forest (Fig. 1A–C).

SeQuiLa package introduces a rule-based optimizer that chooses the most efficient join strategy based on input data statistics computed in runtime. First, the dataset with smaller row count (*s*1) is designated for constructing an interval forest. Then it estimates the size of dataset *s*1' defined as projection of *s*1 on the set of columns referenced by SQL query (Fig. 1B). If it fits into dedicated Spark Driver's memory (controlled by *maxBroadcastSize* parameter) the interval forest is augmented with all columns from *s*1' (*SeQuiLa_it_all* strategy) completing map-side join procedure in one stage. Otherwise an interval tree is used as an index for additional lookup step before the equi-shuffle-join operation between *s*1 and *s*2 (*SeQuiLa_it_int* strategy).

SeQuiLa has been developed in Scala using the Apache Spark 2.2 environment. In runtime it extends SparkSQL Catalyst optimizer with custom execution strategies. It implements distributed map joins using interval forest for inner range join operations. Useful genomic transformations have been added as User Defined Functions/Aggregates and exposed to the SQL interface. Furthermore, SeQuiLa data sources for both BAM and ADAM file formats have been implemented. It can also be integrated with third-party tools using SparkSQL JDBC driver and with R using sparklyr package. SeQuiLa is also available as a Docker container, and can be run locally or on a Hadoop cluster using Yet Another Resource Negotiator (see Supplementary Material for implementation details).

2.2 Performance evaluation

Testing infrastructure consisted of a six-node Hadoop cluster (Cloudera Hadoop distribution version 5.12 with Apache Spark upgraded to version 2.2.1), including four data nodes, a master node and an edge node with 24 CPUs and 64 GB RAM each. To prove the vertical and horizontal scalability of our solution and to compare its performance against existing tools, two tests scenarios have been executed, i.e. on a single node (edge node) and on a cluster, using a whole-exome and whole-genome alignment datasets from NA12878 sample, respectively. In each test, the number of sequencing reads overlapping each one of the pre-defined genomic regions (i.e. either list of exons or genes specified in BED files) has been computed. This type of data processing is widely used in both DNA and RNA sequencing pipelines (see use case examples in Supplementary Material). We have used featureCounts software as a baseline for performance and accuracy comparisons. Finally, we have converted original BAM files into columnar storage format (ADAM) and performed tests on both file formats to observe its impact on the performance (see Supplementary Material for details).

3 Results

SeQuiLa outperforms featureCounts, GenAp and default Spark join implementation in terms of speed on a single node (1.7–22.1×) and a cluster (3.2–4.7×) (Fig. 1D–F). *SeQuiLa_it_all* strategy has proven to perform best in most of the scenarios (no network shuffling required), whereas *SeQuiLa_it_int* performs comparable to, or better than, GenAp. All algorithms favor columnar to row oriented file format due to column pruning and disk reduction of I/O operations.

4 Conclusions

When run in parallel mode, SeQuiLa is the fastest tool in our benchmark, achieving significant performance gain in genomic interval queries. Further, SeQuiLa has a potential to unlock the doors to build additional scalable genomic data warehouse solutions, as well as to implement other higher-level applications for various types of bioinformatics analyses.

Funding

This work was supported by the National Science Center grants [OPUS 2014/13/B/NZ2/01248, PRELUDIUM 2014/13/N/ST6/01843, SONATA 2015/17/D/ST6/04063]; and by the Polish budget funds for science in years 2016–2019 [Iuventus Plus grant IP2015019874].

Conflict of Interest: none declared.

References

Cormen,T. H. *et al.* (2009) Data structures. In: *Introduction to Algorithms*. MIT Press, Cambridge, Massachusetts, pp. 348–354.

Huang,L. *et al.* (2018) Analyzing large scale genomic data on the cloud with Sparkhit. *Bioinformatics*, **34**, 1457–1465.

Kozanitis,C. and Patterson,D.A. (2016) GenAp: a distributed SQL interface for genomic data. *BMC Bioinformatics*, **17**, 63.

Kozanitis,C. *et al.* (2014) Using Genome Query Language to uncover genetic variation. *Bioinformatics*, **30**, 1–8.

Lawrence,M. *et al.* (2013) Software for computing and annotating genomic ranges. *PLoS Comput. Biol.*, **9**, e1003118.

Layer,R.M. *et al.* (2013) Binary Interval Search: a scalable algorithm for counting interval intersections. *Bioinformatics*, **29**, 1–7.

Li,H. *et al.* (2009) The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, **25**, 2078–2079.

Liao,Y. *et al.* (2014) featureCounts: an efficient general purpose program for assigning sequence reads to genomic features. *Bioinformatics*, **30**, 923–930.

Masseroli,M. *et al.* (2015) GenoMetric Query Language: a novel approach to large-scale genomic data management. *Bioinformatics*, **31**, 1881–1888.

Massie,M. *et al.* (2013) Adam: genomics formats and processing patterns for cloud scale computing. *Technical Report, No. UCB/EECS-2013-207*. University of California, Berkeley.