Genome sequence analysis with MonetDB - A case study on Ebola virus diversity

Robin Cijvat, Stefan Manegold, Martin Kersten, Gunnar W. Klau, Alexander Schönhuth, Tobias Marschall, Ying Zhang

Genome sequence analysis with MonetDB: a case study on Ebola virus diversity

Robin Cijvat[1] Stefan Manegold[2] Martin Kersten[1,2] Gunnar W. Klau[2]
Alexander Schönhuth[2] Tobias Marschall[3*] Ying Zhang[1,2]
[1]MonetDB Solutions, Amsterdam, The Netherlands
[2]Centrum Wiskunde & Informatica, Amsterdam, The Netherlands
[3]Saarland University & Max Planck Institute for Informatics, Saarbrücken, Germany
[1]*first.last*@monetdbsolutions.com [2]*first.last*@cwi.nl [3]marschal@mpi-inf.mpg.de

Abstract: Next-generation sequencing (NGS) technology has led the life sciences into the big data era. Today, sequencing genomes takes little time and cost, but results in terabytes of data to be stored and analysed. Biologists are often exposed to excessively time consuming and error-prone data management and analysis hurdles. In this paper, we propose a database management system (DBMS) based approach to accelerate and substantially simplify genome sequence analysis. We have extended MonetDB, an open-source column-based DBMS, with a BAM module, which enables *easy*, *flexible*, and *rapid* management and analysis of sequence alignment data stored as Sequence Alignment/Map (SAM/BAM) files. We describe the main features of MonetDB/BAM using a case study on Ebola virus genomes.

1 Introduction

Next-generation sequencing (NGS) technology has confronted the life sciences with a 'DNA data deluge' [SL13]. Thanks to its massively parallel approach, NGS allows for generating vast volumes of sequencing data, which, in comparison to conventional 'first-generation' sequencing methods, happens at drastically reduced costs and processing times. Consequently, biologists now need to invest in the design of data storage, management, and analysis solutions. Ever more often, improvements in this area are no longer an option, but a pressing issue.

As per common NGS-based "re-sequencing" workflows, short DNA fragments are sequenced and subsequently aligned to a reference genome, which aims at determining the differences between the sequenced genome and the reference genome. Thereby, the resulting alignment data files, most often stored in Sequence Alignment/Map (SAM) format or its binary counterpart BAM [L+09], quickly reach the terabyte mark. Complementary software libraries, e.g., SAMtools [L+09], provide basic functionality, such as predicates-based data extraction. For more complex data exploration, scientists usually resort to writing customised software programs.

*This work was done when the author worked at the Life Sciences group of Centrum Wiskunde & Informatica.

However, the traditional file based approach has several drawbacks. First, it does not make use of technology that specifically addresses large data file handling. Even with compression, a BAM file containing the alignment data of a single human genome amounts to hundreds of gigabytes, which, when processing collections of genomes, quickly reaches the terabyte scale [F+14]. Existing file-based tools usually only work properly with data that fits in main memory. Thus, researchers are often left with the non-trivial tasks of partitioning data into optimally fitting pieces and constructing final results from partial results. Moreover, having to repeatedly reload such large files is undesirable. Second, software development and maintenance are extremely time-consuming and error-prone tasks. They require highly specific knowledge of programming languages and applications. Finally, performance is paramount for big data analysis. Therefore, scientists have been increasingly enforced to become "hard-core" programmers, to exploit the full computational power of modern hardware, such as multicore CPUs, GPUs, and FPGAs.

Although Hadoop systems have recently gained much interest in big data processing, they are no ideal candidates for data analysis in life sciences. Hadoop systems are primarily designed for document-based data processing [DG04]. They can be extremely fast in executing simple queries on large number of documents, e.g., distributed grep. But Hadoop systems quickly suffer from serious performance degradation, when they are used to process more complex analytical queries that often involve aggregations and joins [P+09].

The problems mentioned above have been extensively tackled by database management systems (DBMS), which are designed and built to store, manage, and analyse large-scale data. By using a DBMS for genome data analysis, one can significantly reduce the data-to-knowledge time. A major advantage of using a declarative language such as SQL is that the users only need to state *what* they want to analyse, but not *how* exactly to analyse. The DBMS should take care of efficient execution of the queries, e.g. choose the best algorithms, optimise memory usage, automate parallel execution where possible, and make use of the aforementioned modern hardware, while hiding the heterogeneity of underlying hardware and software systems. In this way, scientists can reap the fruits of 50+ years work of the database community on optimising query processing, so as to spend more time on their primary research topics.

However, so far DBMSs have not been widely adopted in the life sciences beyond metadata management. This is mainly due to the mismatch between what existing DBMSs support and what data analysis in life sciences needs. There is a general lack of DBMS functionality and operations to directly query genomic data already stored in files. The current common practice is to first convert the data into CSV files, then load them into a DBMS. This conversion step not only incurs a high data-to-query time, but also substantially increases storage requirements, especially if the original data are compressed. Moreover, it is extremely difficult to keep duplicate data consistent, when there are updates. Finally, although genomic data are encoded in strings, a standard DBMS data type, they have particular semantics. Without dedicated functions, it is not trivial to express even the basic operations on genomic data using SQL, e.g., compute the length of an alignment.

MonetDB/BAM. Our first step towards a solution for big data analysis in life sciences is to tackle the aforementioned functional mismatches. Therefore, we have extended the open-

source column-based DBMS MonetDB[1] with a BAM module[2], which allows *in-database* processing of SAM/BAM files. The software is available as of the Oct2014 release of MonetDB. MonetDB is primarily designed for data warehouse applications, such as data mining and Business Intelligence [M+09]. These applications are identified by their use of large data sets, which are mostly queried to provide business intelligence or decision support. Similar applications also appear frequently in the big data area of e-science, where observations are collected into a warehouse for subsequent scientific analysis. This makes MonetDB a good candidate to provide a data management solution for such applications.

The main features of MonetDB/BAM include: a) SQL loading functions to load a single, a list or a repository of SAM or BAM files; b) SQL export functions allow writing query results to SAM formatted files; c) SQL functions for elementary operations on sequence alignment data, e.g., computing reverse complements of DNA strings and the actual lengths of mapped sequences; and d) automatically construct primary or secondary read pairs, which accelerates analyses on paired alignments. In this paper, we demonstrate how MonetDB/BAM can be used to facilitate genome sequence alignment data analysis, by conducting a case study on a current and highly important topic: studying the diversity of the Ebola virus.

Related work. There are several prototypes that also use DBMSs for genome data analysis. For instance, Röhm et al. [RB09] propose a hybrid data management approach, which relies on file streaming features of SQL Server 2008 to process gene sequence data stored as binary large objects (BLOBs). When faced with large data file, loading data on-the-fly will suffer from the same performance problems as the file-based approaches. Moreover, this work does not consider additional DBMS functionality to facilitate genomic data analysis. Schapranow et al. [SP13] describe an in-memory DBMS platform, HIG, in which existing applications are incorporated for genome analysis. But HIG does not integrate the analysis functionality *into* the DBMS. The work Dorok et al. [D+14] is most closely related to our work, in the sense that it proposes both a DBMS schema to store genome alignment data and an integrated user-defined function `genotype` (in MonetDB) to enable variant calling using simple SQL. However, this work mainly focuses at supporting variant calling. With MonetDB/BAM, we try to target at genome data analysis in general.

2 Ebola virus diversity: a case study

Viruses populate their hosts as swarms of related, but genetically different mutant strains, each defined by its own, characteristic genomic sequence. Analysing such "mutant clouds", often called *viral quasispecies*, is of clinical importance, as it explains virulence, pathogenesis, and resistance to treatment. Exploring the composition of sequences and their relative frequencies, the *genetic diversity* of a quasispecies, based on NGS is a current, central issue in virology [B+12, T+14].

We demonstrate how to make easy use of MonetDB/BAM for some helpful steps towards an easy exploration of the genetic diversity of Ebola infections. Although it has recently

[1] `https://www.monetdb.org/`

[2] `https://www.monetdb.org/bam/`

```
1 CALL bam.bam_loader_repos('/path/to/ebola-bam-repo', 0)                    (Q1)
```

```
1 SELECT s.value AS refpos,
2        bam.seq_char(s.value, al.seq, al.pos, al.cigar) AS seq_char,
3        COUNT(*) AS cnt
4 FROM generate_series(0, 18960) as s
5      JOIN (SELECT pos, seq, cigar FROM bam.alignments_all WHERE pos > 0) AS al
6      ON s.value BETWEEN al.pos AND al.pos + bam.seq_length(al.cigar)
7 GROUP BY refpos, seq_char ORDER BY refpos, seq_char                        (Q2)
```

```
1 SELECT refpos, SUM(cnt) AS cnt FROM positional WHERE seq_char IS NOT NULL
2 GROUP BY refpos ORDER BY cnt LIMIT k                                       (Q3)
```

```
1 SELECT refpos - refpos % 1000 AS grp_start,
2        refpos - refpos % 1000 + 1000 AS grp_end, AVG(cnt) AS average
3 FROM coverage GROUP BY grp_start, grp_end ORDER BY average DESC LIMIT k    (Q4)
```

```
1 SELECT refpos, coverage.cnt AS coverage, diversity.cnt AS diversity,
2        CAST(diversity.cnt AS FLOAT) / coverage.cnt * 100 AS diversity_perc
3 FROM coverage JOIN (
4          SELECT refpos, SUM(cnt) AS cnt FROM base
5          WHERE seq_char IS NOT NULL AND seq_char <> SUBSTRING(ref, refpos, 1)
6          GROUP BY refpos
7      ) diversity USING (refpos)
8 ORDER BY diversity_perc DESC, coverage DESC, diversity DESC                (Q5)
```

Figure 1: Use case queries

been established that Ebola is a highly diverse and rapidly mutating virus [G+14], conclusive insights are yet to be made. In the project, we use BAM files containing sequence fragments from viral quasispecies of the actual (2014) Ebola outbreak in Sierra Leone [G+14].

Preparing and loading data. First, we retrieved 32 files containing Ebola virus genome sequences (SRP045416) from [G+14]. Together they contain 6,786,308 reads and take 390 MB on hard disk. Then, we used the Burrows-Wheeler Aligner [LD09] to align the reads with the Zaire reference string (NC_002549.1) [V+99], resulting in an average mapping rate of 15.6%. The results are stored in 32 BAM files containing an alignment for every read, with a total size of 500 MB.

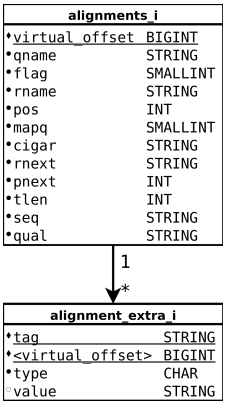

Figure 2: Sequential storage schema

All BAM files are loaded into a MonetDB database with the SQL query Q1 in Figure 1. The first argument is the path to the repository of BAM files. The second argument chooses the storage schema: 0 for sequential, 1 for pairwise[2].In this paper we only use the sequential schema (Figure 2), a straightforward mapping of alignment records in BAM files.

All alignments of one BAM file are stored in two SQL tables `bam.alignments_`i and `bam.alignment_extra_`i, where i is the internal ID of the BAM file. Each tuple in `bam.alignments_`i contains all main fields of one alignment, e.g., `qname`, `flag`, `rname`, `pos`, `cigar`, `seq` and `qual`. The `EXTRA` field of the alignments are parsed and stored in `bam.alignment_extra_`i as `<tag,type,value>` tuples. The `virtual_offset` is used to link the tuples in these tables. For all queries in this work, we have defined a view `bam.alignment_all`, containing `bam.alignment_`i tables of all loaded BAM files.

refpos	seq_char	cnt
...	...	...
46	A	1
46	C	1
47	A	8
...	...	...

Table 1: Result Q2

refpos	cnt
6239	9340
6240	9337
6245	9196
1571	9191
...	...

Table 2: Result Q3

grp_start	grp_end	average
1000	2000	7053.6699999999992
3000	4000	6694.4919999999984
6000	7000	6681.6100000000024
4000	5000	6150.8489999999983
...	...	...

Table 3: Result Q4

Use case 1: computing positional data. Query Q2 in Figure 1 shows how to compute the count of all characters that occur on all positions in MonetDB/BAM. The MonetDB-specific function `generate_series` generates a one-column table with a row for every integer number in the range. We use this in Line 4 to create a table with an entry for every position in the reference genome (NC_002549.1), with a length of 18960 [V$^+$99]. Line 5 selects the alignment position, the sequence, and the CIGAR string for all mapped reads. We join the series table with the mapped reads (lines 4–6). A result tuple is produced if the sequence of the read overlaps with a position in the series table (line 6). The join results are grouped and ordered on the reference positions of the reads and the characters that are found on these positions (line 7). Values of these characters are extracted in the `SELECT` clause (line 2). Finally, from the grouped result, we select the reference positions, the characters on these positions, and their occurrence counters (lines 1–3). Applying Q2 on the Ebola alignment data produces results as shown in Table 1, which reveals that, e.g., on position 46, there is one aligned read with an A and one with a C.

Use case 2: computing coverage and diversity. Assume that the results of Q2 are stored in a table `positional`, query Q3 in Figure 1 shows how to create a top-k of positions that have the highest coverage, i.e., the highest number of aligned reads that overlap with these positions. The results of Q3 in Table 2 show that the reference position 6239 has the highest number of overlapping aligned reads, i.e., 9340.

Assuming the result of Q3 is stored in a view `coverage`, a next step is to calculate a top-k of regions with the highest average coverage, as Q4 in Figure 1. The results of Q4 are in Table 3, which shows that the region 1000–2000 has the highest average coverage.

Diversity is another interesting analysis we can do with the results of Q2 and Q3, i.e., compute the percentage of alignments that differ from the reference genome on each position. The query Q5 in Figure 1 produces a list of positions with their coverage and diversity, with decreasing diversity percentages. In Q5, we have loaded the reference genome string (NC_002549.1) [V$^+$99] in the SQL variable `ref`. The function `SUBSTRING` returns a single character at the given `refpos`. Q5 first computes similar intermediate data as in Q3 (lines 4–6), except filtering out the positions with matching characters with the reference genome (line 5). Then, we join the `coverage` table with the just computed diversity information on the reference position (lines 3–7). This gives us for every position: i) the total number of overlapping read alignments, and ii) the number of overlapping read alignments that differ from the reference genome. Finally, we select the reference position, the count of both the coverage and the diversity subresults, and calculate the diversity percentage for all reference positions as

refpos	coverage	diversity	diversity_perc
721	1471	1471	100
7029	1469	1469	100
5639	1131	1127	99.6463307
...	...	...	...

Table 4: Result Q5

the number of differing read alignments divided by the total number of read alignments for these positions (lines 1,2). The results of Q5 are in Table 4, which e.g. shows that all aligned reads at reference positions 721 and 7029 differ from the reference genome.

Query performance. We run all five queries on a moderate laptop (i7 Quad Core CPU, 4GB RAM, SSD disk). Figure 3 shows the query execution times on different data sizes. The x-axis denotes both the number of files and the file size for each data set. All results are averages of 10 runs.The execution times show a linear behaviour with growing data size. Loading (Q1) and computing positional data (Q2) are the most time consuming tasks. Q1 spends most time on decompressing the BAM files and passing values. The execution times of Q2 include the time to store its results in the `positional` table, which serves as a preprocessing step for the remaining queries. Once data are loaded and preprocessed, the further analysis queries (Q3 – Q5) are done in milliseconds, and the execution times are hardly affected by growing number of files and data sizes.

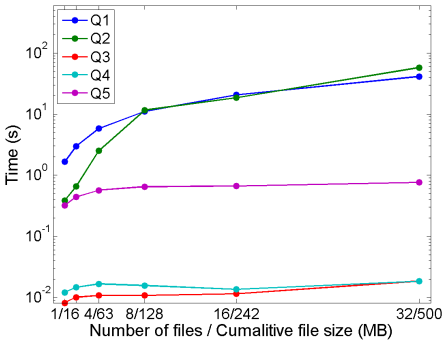

Figure 3: Execution times of all queries

Conclusion and Outlook. In this paper, we showed how to use MonetDB/BAM to facilitate exploration of the genetic diversity of the Ebola virus. Our study indicates that many conceivable analyses on genome sequence alignment data can be easily expressed as SQL queries, provided the DBMS offers the proper functionality. However, before we come up with a comprehensive solution for big data analysis in life sciences, plenty open issues call for consideration. For instance, the performance and scalability of MonetDB/BAM must be extensively evaluated and improved. We should stress the system with both BAM files of larger genomes, such as human or plant genomes, and terabytes scale file repositories. Also, we should compare the performance of our approach with existing analysis tools, such as BEDTools [AI10]. Moreover, the use cases study should be extended with more important analysis, such as variant calling [N+11], so as to determine more functional requirements MonetDB/BAM should satisfy. Finally, workflow support is a must for scientific exploration. MonetDB already provides various client connections (e.g., JDBC, ODBC), a REST interface, and seamless integration with the R project for statistical computing[1]. Therefore, MonetDB can be easily integrated into existing workflow systems, such as Taverna [W+13].

References

[AI10] Quinlan AR and Hall IM. BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics*, 26(6):841–842, 2010.

[B+12] N. Beerenwinkel et al. Challenges and opportunities in estimating viral genetic diversity from next-generation sequencing data. *Frontiers in Microbiology*, 2012.

[D+14] S. Dorok et al. Toward Efficient Variant Calling Inside Main-Memory Database Systems. In *DEXA Workshops*, pages 41–45, 2014.

[DG04] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *OSDI*, 2004.

[F+14] L. C. Francioli et al. Whole-genome Sequence Variation, Population Structure and Demographic History of the Netherlands. *Nature Genetics*, 46:818–825, 2014.

[G+14] S. K. Gire et al. Genomic surveillance elucidates Ebola virus origin and transmission during the 2014 outbreak. *Science*, 345(6202):1369–1372, 2014.

[L+09] H. Li et al. The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, 25, 2009.

[LD09] H. Li and R. Durbin. Fast and accurate short read alignment with Burrows-Wheeler Transform. *Bioinformatics*, 25:1754–60, 2009.

[M+09] S. Manegold et al. Database Architecture Evolution: Mammals Flourished long before Dinosaurs became Extinct. *PVLDB*, 2(2):1648–1653, 2009.

[N+11] R Nielsen et al. Genotype and SNP calling from next-generation sequencing data. *Nat. Rev. Genet.*, 12(6):443–451, 2011.

[P+09] A Pavlo et al. A Comparison of Approaches to Large-Scale Data Analysis. In *SIGMOD*, 2009.

[RB09] Uwe Röhm and Jose A. Blakeley. Data management for high-throughput genomics. In *CIDR*, 2009.

[SL13] M. C. Schatz and B. Langmead. The DNA data deluge. *IEEE Spectrum*, 50(7):28–33, 2013.

[SP13] Matthieu-P. Schapranow and Hasso Plattner. HIG - An in-memory database platform enabling real-time analyses of genome data. In *BigData*, pages 691–696, 2013.

[T+14] A. Toepfer et al. Viral quasispecies assembly via maximal clique enumeration. *PLoS Computational Biology*, 10(3):e1003515, 2014.

[V+99] V. E. Volchkov et al. Characterization of the L gene and 5' trailer region of Ebola virus. *The Journal of general virology*, 80(Pt2):355–62, 1999.

[W+13] K. Wolstencroft et al. The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. *Nucleic Acids Research*, 41(Web Server issue):W557–W561, 2013.