Revista de Informática Teórica e Aplicada - RITA - ISSN 2175-2745
Vol. 25, Num. 2 (2018) 93-100

RESEARCH ARTICLE

Relative Scalability of NoSQL Databases for Genotype Data Manipulation

Escalabilidade Relativa de Bancos de Dados NoSQL para Manipulação de Dados de Genótipo

Arthur Lorenzi Almeida[1], Vinícius Junqueira Schettino[2], Thiago Jesus Rodrigues Barbosa[2], Pedro Fernandes Freitas[1], Pedro Gabriel da Silva Guimarães[2], Wagner Arbex[3]*

Abstract: Genotype data manipulation is one of the greatest challenges in research fields such as population genetics, bioinformatics and genomics mainly because of high dimensionality and unbalancing characteristics. These peculiarities explain why relational database management systems (RDBMS), the ”de facto” standard storage solution, have not been presented as the best tools for this kind of data. However, the Big Data advent has been pushing the development of modern database systems that might be able to overcome RDBMS deficiencies. In this context, we extended our previous works on the evaluation of relative performance among NoSQLs engines from different families, adapting the schema design in order to achieve better performance based on its conclusions, thus being able to store more SNP markers for each individual. Using Yahoo! Cloud Serving Benchmark (YCSB) benchmark framework, we assessed each database system over hypothetical genotype data (SNP markers). Results indicate that Tarantool is approximately 21,8% more efficient than MongoDB when storing 770,000 SNP markers, but MongoDB is less impacted by the increase of SNP markers per individual.

Keywords: Database — NoSQL — Bionformatics — Data Science — SNP, Genotype

Resumo: A manipulação de dados de genótipo é um dos maiores desafios em áreas de pesquisa como genética de populações, bioinformática e genômica, principalmente devido às características de alta dimensionalidade e desbalanceamento. Essas peculiaridades explicam por que os sistemas de gerenciamento de bancos de dados relacionais (SGBDR), a solução padrão ”de fato” de armazenamento, não se apresentam como as melhores ferramentas para esse tipo de dados. Todavia, o advento do Big Data tem impulsionado o desenvolvimento de sistemas de bancos de dados modernos que podem superar deficiências de SGBDR. Neste contexto, estendemos nossos trabalhos anteriores sobre a avaliação do desempenho relativo entre os mecanismos NoSQLs de diferentes famílias, adaptando a proposta do esquema para obter melhor desempenho com base em suas conclusões, podendo armazenar mais marcadores SNP para cada indivíduo. Usando o Yahoo! Cloud Serving Benchmark (YCSB), avaliamos cada sistema de banco de dados sobre genótipos hipotéticos (marcadores SNP). Os resultados indicam que o Tarantool é aproximadamente 21,8% mais eficiente que o MongoDB ao armazenar 770.000 marcadores SNP, mas o MongoDB sofre menos impacto com o aumento de marcadores SNP por indivíduo.

Palavras-Chave: Banco de Dados — NoSQL — Bionformática — Ciência de Dados, SNP — Genótipo

[1] *Department of Computer Science, Federal University of Juiz de Fora, Juiz de Fora, MG, Brazil*
[2] *Postgraduate Program in Computer Science, Federal University of Juiz de Fora, Juiz de Fora, MG, Brazil*
[3] *Federal University of Juiz de Fora and Brazilian Agricultural Research Corporation, Juiz de Fora, MG, Brazil*
*__Corresponding author__: wagner.arbex@{ufjf.edu.br, embrapa.br}
DOI: http://dx.doi.org/10.22456/2175-2745.79334 • **Received:** 02/01/2018 • **Accepted:** 20/04/2018

1. Introduction

Conducting research on genotypes of entire populations presents the challenge of storing data in a structured manner with the guarantee of durability. This challenge extends to groups that design and develop computational tools, analysis pipelines or aggregated databases of this kind of data [1]. High dimensionality is the main reason why most database solutions do not handle genotype data efficiently. Big Data, however, pushed the development new database technologies designed under different premises; these systems are being called NoSQL databases.

The exact definition of a NoSQL database has not reached a consensus yet, despite the fact that the "NoSQL" buzzword stands for "Not only SQL". Nevertheless, A more appropriate description includes the traits shared among databases classified as NoSQL: (i) horizontal scalability over many servers; (ii) data replication and partitioning; (iii) simple call interfaces; (iv) weak concurrency model; (v) distributed indexes and usage of RAM and (vi) less rigid schemas [2].

For NoSQL databases, a weak concurrency model is generally considered advantageous because it facilitates the development of distributed systems (which increases scalability and availability) and processing of huge amounts of data. This kind of concurrency model is usually based on CAP theorem's assertion that it is impossible for a distributed data store to guarantee consistency, availability and partition tolerance at a given time (at best two of these can be provided simultaneously). This whole design premises are very different from the ACID (Atomicity, Consistency, Isolation, Durability) transactions observed in relational databases engines, that are suitable for most applications but a drawback when in cases that they are not needed, such as dealing with flexible schemas and unstructured data [3].

Some forms exist to represent an individual's genome, SNP genotyping is one of the most common and uses a subset of the genome, more specifically a set of known SNP markers, as representative. SNP stands for single-nucleotide polymorphism, a variation in a nucleotide in a specific position of the genome, and is the most common type of variation in humans. This variation must be present in at least a certain portion of the population, usually 1%. Since the vast majority of SNPs are bi-allelic, a common representation considers three possible pairs of alleles, one for homozygous dominant (AA), one for homozygous recessive (BB) and the last one for heterozygous (AB)[4].

The importance of SNP markers comes mainly from their application on association studies and haplotype mapping [5, 6], which can identify SNPs – or possibly alleles when SNP is in a coding region of the DNA – related to phenotypic traits, especially diseases. When the SNP is tightly related to a gene, i.e., the specific SNP nitrogenous base can be mapped to an allele, the genetic code of an individual can be represented by the pair of alleles existent for each SNP.

There are many techniques to identify SNPs, but SNP genotyping only became popular in the last 15 years when sequencing cost was greatly reduced by new methodologies [7]. Current technologies are usually divided into low, medium and high density based on the arrays used to detect SNPs. A higher density implies in more markers identified and consequently a more precise representation of genetic variation. This difference in granularity can range from a few thousands of markers (e.g., 3000 SNPs) to over 1.5 million SNPs for a single individual.

As genotyping services become more accessible, projects can opt to use denser genotypes and increase populations sizes on studies. Naturally, it is possible that the average density used in experiments will increase, which leads to the need for suitable infrastructures to manipulate and store SNP markers and related metadata, especially phenotype data.

Motivated by this scenario, this work aims to evaluate the relative scalability of different NoSQL databases when handling genotype data, extending our previous study regarding the performance of four database management systems (DBMS) – Tarantool, MongoDB, OrientDB and HBase [8]. These specific systems were chosen because of their differences between each other.

Tarantool is the representative of key-value stores and also an in-memory database, MongoDB is document based, HBase's data model supports column families and OrientDB is a multi-model DBMS who supports graph, document, key-value and object storage in one core, making it extremely versatile. Each type of storage is generally considered to be a NoSQL "family" and in fact, more of them exist [9]. We opted to consider only those four because their families are the most popular [10] and their data models seemed to be more appropriate to SNP handling.

2. Related Work

In a previous work, we have approached the relative performance of NoSQL engines for genotype data manipulation topic [8]. The main objective of that study was to experiment with NoSQL engines to determine whether they were suitable to genotype data manipulation and which data model would be more efficient in doing that. In the obtained results, Tarantool proved to be more efficient than MongoDB by a large margin and it was confirmed that non-tabular databases can be a very good option for this kind of data. However, results needed to be expanded as they lacked any scalability evaluation, an important aspect to be considered since genotype databases are constantly growing. Furthermore, each database was evaluated within two scenarios, one to measure insertion performance and the other to assess reads and updates. These scenarios do not adequately represent real use-cases, as we explain in subsection 3.3. Finally, previous results did not encompass OrientDB and HBase, decreasing its conclusion's scope.

Others recent works explored the benefits of NoSQL engines in genotype data handling. The work [11] explores features commonly found on NoSQL databases to propose a tool for managing genomic, transcriptomic and genotyping

Table 1. NoSQL engines characteristics summary

System	Data model	Main features	Written in	Licensing
HBase	Column family	• High fault tolerance • Built on a distributed FS	Java	Apache License 2.0
MongoDB	Document-based	• Ad hoc queries	C++	GNU Affero GPL
OrientDB	Multi-model	• Multiple schema modes • SQL interface • ACID transactions	Java	Apache License 2.0
Tarantool	Key-value store	• In-memory storage • Write-ahead logging • ACID transactions	C and Lua	BSD 2.0

data, although the work does not present a comparison between their tool and other nonrelational engines or families. Some of the challenges related to the storage of genotype and phenotype data in a real project are exposed by [12]. Although the study encompasses more structural components than only database systems, it highlights the presences of irregular data models, the heterogeneous nature of the data and the need for flexible schemas.

3. Materials and Methods

In order to evaluate the performance of NoSQL databases when handling SNPs, the experiment was conducted using a five-step process: (i) selection of a set of databases to represent the most common NoSQL families; (ii) schema modeling for each system; (iii) definition of workloads that are similar to real cases of SNP data manipulation; (iv) execution of the workloads in each database (v) analysis of the obtained results. This section will explore each of these steps in detail.

3.1 Database selection

The selection of NoSQL families representatives was done firstly considering which data models could fit SNP data and were, in general, more popular. Document, key-value and wide column stores have been receiving more attention from the database community than more specialized solutions such as RDF and time series databases [10].

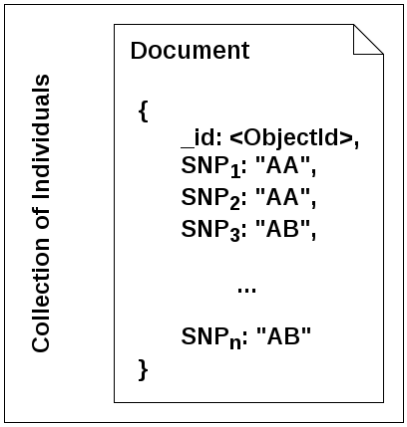
Apache HBase™ was selected as the representative of column families due to its popularity. This database is written in Java designed and runs over Hadoop, a distributed file system. The system aims to offer a highly scalable and fault tolerant store over a cluster of commodity hardware. HBase is distributed under offers features such as sharding, consistent reads and writes and failover support [13]. HBase was evaluated using its 0.98.24 version over Hadoop 2.5.2 and used pre-splitting of regions, as recommended by HBase developers.

MongoDB is a distributed database designed to store data in JSON-like documents and the example of a document store for this experiment. The presence of JSON data structures in languages such as JavaScript and Python, make this database a good option for many software projects. Given that fact and that it is published under GNU Affero General Public License [14], it is reasonable that MongoDB is ranked as the most popular nonrelational engine [10]. It also supports field and range queries, regular expressions searches and user-defined JavaScript functions. MongoDB is written in C++ and in this work, version 3.2.13 was used without additional configurations to the server or client.

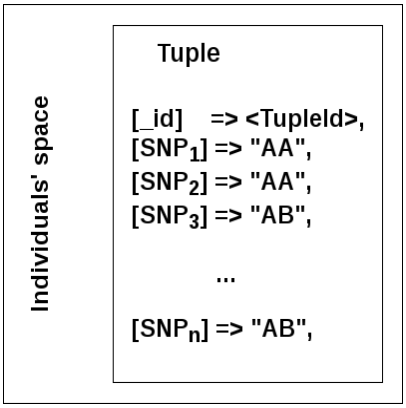
To increase the variety of systems used in the experiment and also evaluate how a multi-model database performs, OrientDB was included in the benchmarking process. It supports both graph and document-based database models, the latter supposedly been more attractive to store SNP sequences. OrientDB can also handle data in schema-less, schema-full and schema-mixed modes, has a SQL interface and supports ACID transactions. Written mainly in Java and under the Apache License 2.0, it is described "The first and best scalable, high-performance, operational NoSQL database" [15]. In this experiment, version 2.2.10 was used with default configurations.

Tarantool is an open-source (BSD 2.0) in-memory DBMS that uses write-ahead logging to achieve durability and atomicity and also supports SQL querying. Its main core languages are C and Lua. As others key-value engines, as Redis and Amazon ElastiCache, it presents features including sharding and load balancing [16]. Tarantool is expected to have the best results on this experiment, due to its in-memory implementation, although a larger consumption of RAM memory is also expected. By default, the system allocates up to 1MB for each tuple and 256MB for total storage, but due to the nature of SNP data and seeking to simulate a more realistic environment, these values were increased to 2MB and 12GB, respectively. We do acknowledge the constraints of using an in-memory database, especially its greatly reduced cost-effectiveness for larger sets of data due to size limitations. In the context of our experiment, specifically when taken into consideration the dimensions of the database and hardware used, these disadvantages are not very significant. Tarantool was chosen as the key-value database representative in this experiment, despite the fact that is Redis is a more popular in-memory key-value store. Tarantool was included mainly because it had a very good performance in our previous benchmarks.

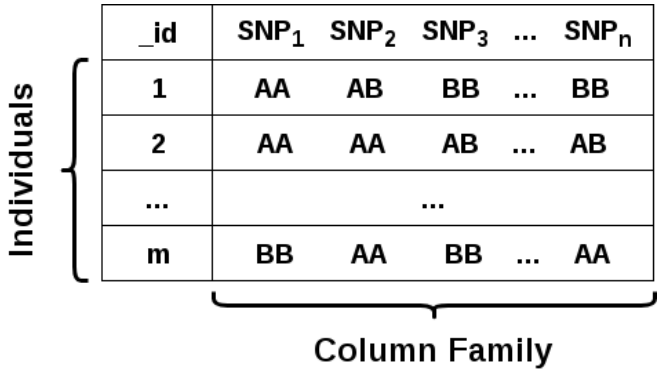
The unique main features of cited NoSQL engines are highlighted in table 1 along with some other characteristics. Features such as sharding and horizontal scaling are absent in



(a) JSON document model with a single level storing all SNP values



(b) Mapping of SNPs to values used in key-value stores



(c) Column family model where all columns belong to the same family

Figure 1. Data models used on experiments

the table because they are shared between all systems.

3.2 Schema modeling

A database schema design significantly affects the implemented system's performance; therefore, properly modeling is necessary to a valid database benchmark assessment. The chosen databases were evaluated over a hypothetical population where each individual was associated with a sequence of values that represented their genotype. Each position in these sequences informed the sample's pair of alleles for a specific SNP and since a pair of alleles can be homozygous dominant, homozygous recessive or heterozygous, the domain of these sequences consisted of only three values.

A collection of SNP sequences naturally fits a relational model where each column stores one or more SNPs and rows represent a single individual. Non-tabular models, however, can handle this kind of data in a simpler and more beneficial manner because they support flexible schemas. A research project could, for example, store individuals' genotypes obtained from distinct sources or generated by different technologies in the same space of a database. In this evaluation, three alternatives to classical relational models were considered: document based, key-value stores and column families.

In the database context, a document usually refers to JSON or XML files. These files can be represented as trees where a node's child can be a primitive value or another tree. Both MongoDB and OrientDB use a JSON-like document format that requires a key for each attribute (or simply child node). To store SNP sequences, the ability to increase a tree's depth is not needed since the individual's alleles are always represented by primitive data, thus making the document a structure that maps keys to values, similarly to a hash table. The document should hold all data related to an individual, like a row in RDBMS, and be part of a document collection to allow database administrators to cluster related documents as they see fit. Indexing is done in the scope of collections to allow fast retrieval of documents.

For key-value stores, such as Tarantool, modeling is very straightforward: each stored value must be related to a key. A set of these key-value pairs can be assembled into a tuple that should be stored in a space similar to a document collection that also holds indexes. In the context of this experiment, each tuple stores data from a single individual, each pair represents an SNP identification and its value for an individual.

Finally, in wide column stores, the concepts of tables are still present. The differences to relational databases are more

present in implementation than modeling. In this model, rows store data of a single individual while columns represent the values for each SNP and are clustered into groups called column families. Column families allow the creation of columns groups that should contain similar search and modifications patterns, they can also change the way a table is stored in the disk, affecting performance. In the context of this study, there were no benefits in dividing columns into different column families because reads encompassed all fields of a row, so a single family was defined.

A visual representation of the adopted models is presented in Figure 1.

3.3 Workload definition

The next steps after the selection of database engines and modeling of schemas were the definition of suitable workloads and their execution. Both of this steps were conducted using The Yahoo! Cloud Serving Benchmark (YCSB), a framework developed to facilitate performance comparisons of cloud data serving systems [17]. YCSB is heavily based on the modularity and extensibility design principles, making it possible to add support for new systems or benchmarking techniques without changes to its core module. Most of the supported data serving systems are classified as NoSQL, corroborating to our choice of using this software for the experiment. The benchmark tool also became vastly used in the industrial and academic fields for comparing NoSQL databases [18, 19, 20]. It is distributed under the Apache License 2.0[1] and its source code can be found on Github[2]

YCSB architecture is centered on measuring throughputs and latencies of user-defined workloads described on files that can be reused for multiple database systems. A workload definition consists of not only operations to be executed, but data characteristics such as the number of records, fields and its size. Using these pieces of information, YCSB is able to simulate real databases, avoiding the need to support different input formats of real data. Concerning the operations used in the benchmarking process, a workload must define the number of operations to be executed and the proportion of each type, which can be inserts, reads, updates and scans. When reading/scanning records, a distribution can be defined to select records. All results presented in this paper were obtained using a uniform distribution, meaning that all records had the same probability of being read during measurement.

The first step taken to create our workloads was choosing which operations were, in general, most relevant to population genetics, bioinformatics and genomics. Updates are not expected in this context because changes in genotype data are rare, inserts are also not frequent – especially when compared to more typical use cases of databases – since they require genotyping of genetic material. Therefore, those kinds of operations were ignored in all workloads. Besides that, these operations do not influence data analysis, commonly done in memory, hence all NoSQL systems were evaluated over read operations. This makes the testing scenarios significantly different from what was done in previous NoSQL evaluations for genotype data.

In order to evaluate scalability, performance measures must be done over various database dimensions in different executions. It is also important to point out that in previous studies, tests considered datasets with at most 56,000 SNP markers despite the fact that newer technologies allow genotyping of millions of markers [8]. This reduces the comprehensiveness of the obtained results. To encompass all these aspects, the amount of SNP markers per individual varied between 20,000, 56,000 and 770,000 in different workloads. Population sizes, in contrast, varied between 2000, 5000 and 10,000 individuals. These values were chosen based on population and sequence sizes that are commonly found in genetic improvement of dairy cattle research. To cover all possible database dimensions, nine workloads were defined.

3.4 Conducting The Experiment

Preliminary executions pointed out that the performance of executions over 770,000 SNP markers would not be sufficient for a real case scenario. In order to deal with this problem, data representation had to be slightly changed: every SNP was being stored using a single byte even though only two bits would be sufficient to store three possible values; since the number of fields was a determining factor in performance, the representation was adapted to store more than one SNP marker per field. The final form used eight-byte fields storing 32 markers because most modern architectures support 64-bit integers, allowing the extraction of a SNP using bitwise operations. The SNP extraction overhead was considered negligible because bitwise operations are implemented in hardware.

The changes in representation resulted in a dimensionality reduction that allowed some of the evaluated systems to efficiently handle a higher number of SNP markers. To store 20,000 SNP markers for an individual, 625 columns were needed, for 56,000 and 770,000 markers the required amount of columns was 1750 and 24063, respectively. Obviously, this solution requires a mapping of the columns to the SNPs that it stores. This part of the solution was not explored in this work because applications that consume genotype data will, generally, read the entire tuples, i.e., all markers.

Once the workloads were adapted to adequately store 32 SNP markers in each column, they were executed in all four databases described on Section 3.1. For consistency analysis, each workload was executed 10 times by MongoDB, OrientDB, HBase and Tarantool. The throughput (in operations per second) was measured during each execution, and between these executions, all databases were completely destroyed and DBMS cleaned to avoid any bias on consequent tests. The execution time of the reads was limited to 10 minutes, if a NoSQL system was not able to read the entire database in 10 minutes, execution would be interrupted. We found this threshold ac-

[1] https://www.apache.org/licenses/LICENSE-2.0
[2] https://github.com/brianfrankcooper/YCSB

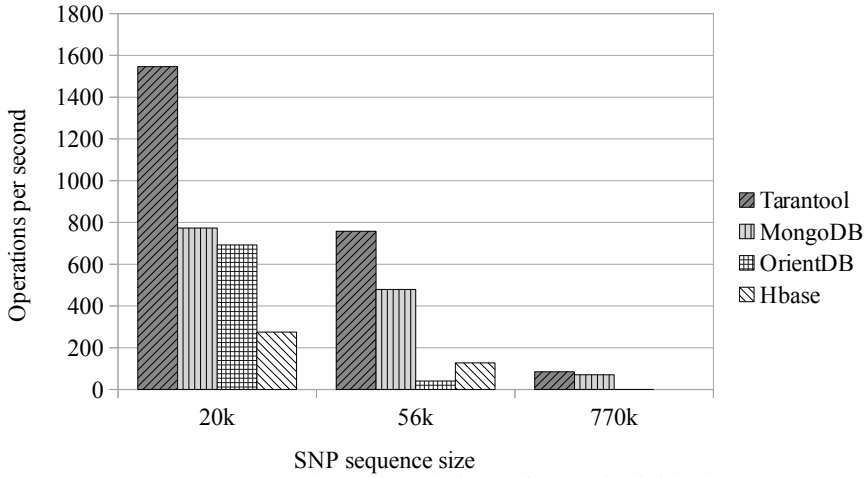


Figure 2. Results on populations of 2,000 individuals

ceptable since reading SNP markers with such performance would be unfeasible for most known applications. Therefore, for these cases, throughput values were normally collected but duly indicated on results section. The same policy was not applied on data loading phase (necessary to run read operations) because of the bias it would add to results when readings were to be made over smaller populations instead of the expected.

4. Results

The computational results exposed in this section were obtained executing all workloads in a dual-core machine, with 16GB of RAM in a Debian 8.8 environment. All databases engines and YCSB were installed accordingly to developers' instructions and all executions were automatized.

After the execution of the defined workloads, the results shown on table 2 were obtained. In more than half of the cases, the population size growth was accompanied by a throughput gain. This increase in throughput was relatively smaller in scenarios where more SNP markers were being stored. MongoDB executions' throughputs, for example, increased nearly 28% from 5,000 to 10,000 individuals when handling 20,000 markers sequences, but only 1.2% when manipulating 770,000 SNPs genotypes. A possible reason for this behavior is the network overhead; if it represents a significant fraction of the total time, an increase in the number of operations would be beneficial throughput.

Although HBase's performance was for most of the workloads better than OrientDB's, within the current context and configuration it was not possible to execute tests on it with 770,000 markers because the load phase was taking too long and using a lot of space (more than 60GB). OrientDB held the worst throughputs among all systems except on one case and was not able to finish all read operations in less than 10 minutes when handling 770,000 SNP markers, these results are indicated with a "*".

The behavior of each NoSQL engine when handling 2,000 individuals workloads is shown in Figure 2. Tarantool had the better throughput for every SNP sequence size followed by MongoDB. The patter is repeated for 5,000 and 10,000 individuals workloads (Figures 3 and 4).

Table 2. Throughput results

	Pop. Size	SNP markers		
		20k	56k	770k
HBase	2k	274.7258	127.2769	-
	5k	315.7282	143.1624	-
	10k	312.2846	80.2931	-
MongoDB	2k	773.5585	478.6498	70.6957
	5k	1076.9967	619.6147	74.3170
	10k	1384.0230	732.4774	75.2733
OrientDB	2k	692.3321	40.1699	0.2420*
	5k	223.0814	42.6078	0.2430*
	10k	234.8291	35.3310	0.2318*
Tarantool	2k	1546.7201	759.2681	85.4458
	5k	1846.6466	893.3301	83.7490
	10k	2052.3188	925.2444	88.0793

The behavior of each NoSQL engine when handling 2,000 individuals workloads is shown in Figure 2. Tarantool had the better throughput for every SNP sequence size followed by MongoDB. The patter is repeated for 5,000 and 10,000 individuals workloads (Figures 3 and 4).

Although Tarantool's advantage over MongoDB is visible on low-density sequences, their throughputs get closer on higher densities. For example, while MongoDB's performance is only 67% of Tarantool's on workloads with 10,000 individuals and 20,000 SNP markers, this percentage increases to about 85% on 770,000 markers. This is an evidence that Tarantool loses performance on a greater ratio than MongoDB. On workloads with 10,000 individuals, MongoDB preserver 10% of its throughput when comparing 20,000 and 770,000 sequence sizes, whereas on the same conditions Tarantool

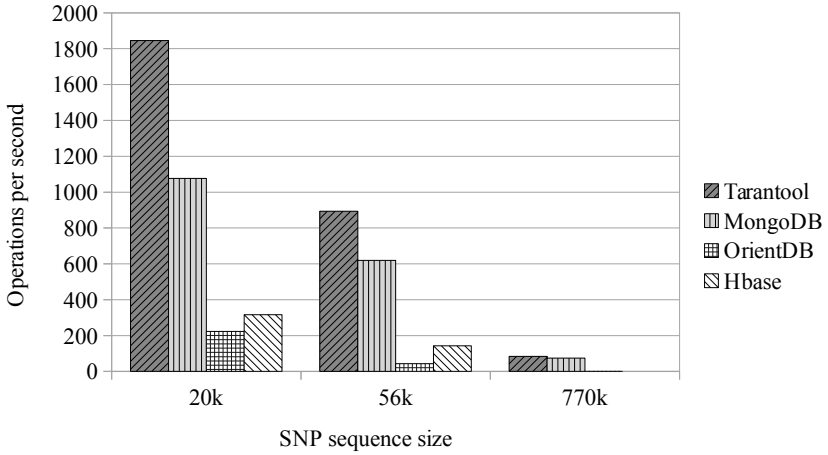


Figure 3. Results on populations of 5,000 individuals

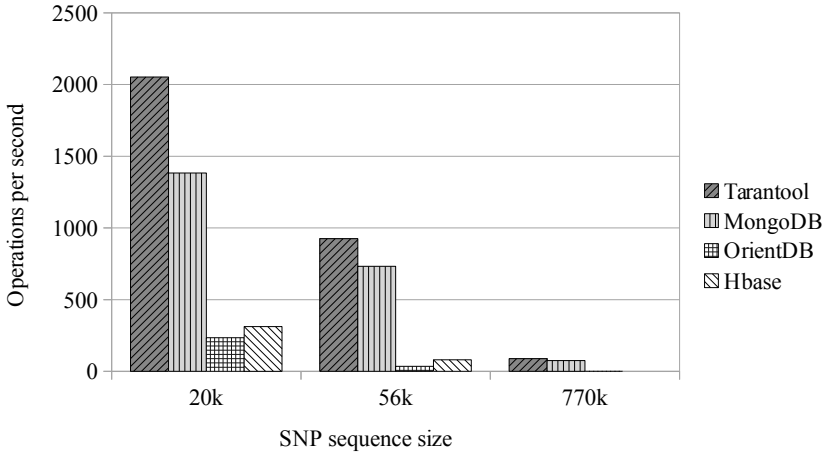


Figure 4. Results on populations of 10,000 individuals

kept only 4.29% of its performance. For the same comparison, OrientDB presented the worst ratio, keeping only 0.1% of its primary throughput. HBase's scalability was not asserted due to the lack of data for the more complex workloads.

5. Conclusion

Tarantool presented the best throughputs, even when dealing with a large number of SNP markers, such as in previous evaluations [8]. However, the performance gap between Tarantool and MongoDB was greatly reduced in this experiment, mainly because this time testing scenarios were focused on reads, which makes results more similar to real use-cases. Moreover, MongoDB's performance was less affected by the increase in SNP sequences sizes. Overall, both systems are capable of handling large genotype databases, but if data evolution is a concern, results point out MongoDB as a better choice. Also, it should be noted that Tarantool is restricted by memory size and consequently may require more expensive hardware.

Concerning scalability on population sizes, the results of this work contribute to the idea that they are not significant on performance, at least on sizes related to controlled populations used on studies such as in the genetic improvement of dairy cattle. These numbers are usually of that magnitude due to the costs of finding individuals to run analysis and the genotyping costs. We cannot draw conclusions over aggregate databases that might include millions of individuals, but they can be verified further in future works. With the current results, we can conclude that the size of SNP sequences has a major impact on choosing a NoSQL engine to handle genotype data.

Also, we were able to store much larger sequences than our previous work after deciding to store 32 markers on each field. This may be a complementary sign that the number of fields takes a major place on the performance of NoSQL databases, and finding new techniques to store and retrieve them taking this into account can lead to better results.

For upcoming works, the engines should be tested on a distributed environments, since this could reveal new potentials to handle genotype data. Also, testing these systems with real genotype data like real applications and might reveal new relevant aspects after analysis.

6. Threats to Validity

There is some possible bias in the results of this paper, and they must be taken into account while extending and using this work in further conclusions and research. First of all, the standard and flexible nature of YCSB does put away a set of singularities of each engine. That may lead to extra performance lost that could be avoided with further understanding and considerations about each engine details.

Similarly, the results could be significantly different after investigation and experimentation with different software configurations such as using OrientDB key-value model or sharding and load balancing configuration when executing the DBMS over a cluster. It is also important to emphasize that our workloads increased the size of the SNP sequences on a much bigger proportion than in population. Although this is a common ratio for genotype data, it narrows this work conclusions to this kind of data.

7. Acknowledgements

The authors thank the reviewers who gave purposive and helpful comments, as well as express thanks to the National Research Center of Dairy Cattle (Embrapa Dairy Cattle) of Brazilian Agricultural Research Corporation (Embrapa); the Federal University of Juiz de Fora (UFJF); the Coordination for the improvement of Higher Level Personnel (CAPES); State of Minas Gerais Research Support Agency (FAPEMIG); and to the National Council for Scientific and Technological Development (CNPq).

8. Author contributions

ALA and VJS carried out the experiments, analyzed the results and contributed to the methodology of this study. TJRB, PFF and PGSG structured and provided the infrastructure and computing resources needed to perform the experiments. PFF revised the text. WA is the project leader, proposed the methodology and general approach of this study. All authors read and approved the final manuscript.

References

[1] PARADIS, E. et al. Linking genomics and population genetics with r. *Mol Ecol Resour.*, Wiley Online Library, v. 17, n. 1, p. 54–66, 2017.

[2] CATTELL, R. Scalable SQL and NoSQL data stores. *Acm Sigmod Rec*, v. 39, n. 4, p. 12–27, 2011.

[3] STONEBRAKER, M. Sql databases v. NoSQL databases. *ACM Comm.*, v. 53, n. 4, p. 10–11, 2010.

[4] CONSORTIUM, . G. P. et al. A global reference for human genetic variation. *Nature*, v. 526, n. 7571, p. 68, 2015.

[5] SHI, W. et al. Informative snps selection based on fuzzy clustering and genetic algorithm. *J Comput Theor Nanosci.*, v. 14, n. 3, p. 1440–1445, 2017.

[6] ZHANG, K. et al. Haplotype block partitioning and tag SNP selection using genotype data and their applications to association studies. *Genome Res.*, v. 14, n. 5, p. 908–916, 2004.

[7] CAETANO, A. R. Marcadores SNP: conceitos básicos, aplicações no manejo e no melhoramento animal e perspectivas para o futuro. *Rev. Bras. Zootecnia*, v. 38, n. 8, p. 64–71, 2009.

[8] SCHETTINO, V. J. et al. Avaliação do desempenho relativo de bancos de dados NoSQL para arquivos de genótipos. *BRESCI 2016*, v. 1, n. 1, p. 306–309, 2016.

[9] EDLICH, S. *NoSQL*. 2016. Disponível em: ⟨http://www.nosql-database.org⟩.

[10] IT., S. *DB-Engines Ranking*. 2017. Disponível em: ⟨https://db-engines.com/en/ranking⟩.

[11] SEMPÉRÉ, G. et al. Gigwa—genotype investigator for genome-wide analyses. *GigaScience*, v. 5, n. 1, p. 1–9, 2016.

[12] CHLEBIEJ, M. et al. Architectural challenges of genotype-phenotype data management. In: BARBOSA, S. et al. (Ed.). *International Conference: Beyond Databases, Architectures and Structures*. Ustroń, Poland: Springer, 2015. (Communications in Computer and Information Science).

[13] FOUNDATION, A. S. *Apache HBase*. 2017. Disponível em: ⟨https://hbase.apache.org⟩.

[14] MONGODB, Inc. *MongoDB for GIANT Ideas*. 2017. Disponível em: ⟨https://www.mongodb.org⟩.

[15] ORIENTDB Ltd. *OrientDB - Distributed Multi-Model and Graph Database*. 2017. Disponível em: ⟨http://orientdb.com/orientdb⟩.

[16] GROUP, M. *Tarantool - Get your data in RAM. Get compute close to data. Enjoy the performance*. 2017. Disponível em: ⟨https://tarantool.org⟩.

[17] COOPER, B. F. et al. Benchmarking cloud serving systems with ycsb. In: HELLERSTEIN, J. M. (Ed.). *Proceedings of the 1st ACM Symposium on Cloud Computing*. New York, NY, USA: ACM, 2010. (SoCC '10, v. 1), p. 143–154.

[18] ABUBAKAR, Y.; ADEYI, T. S.; AUTA, I. G. Performance evaluation of NoSQL systems using YCSB in a resource austere environment. *Perf. Evaluation*, v. 7, n. 8, p. 23–27, 2014.

[19] MOREIRA, L. O.; SOUSA, F. R.; MACHADO, J. C. Analisando o desempenho de banco de dados multi-inquilino em nuvem. In: OLIVEIRA, J. P. M. de (Ed.). São paulo, SP: Brazilian Computer Society Special Interest Group on Databases, 2012.

[20] FRIEDRICH, S. et al. Nosql OLTP benchmarking: A survey. In: LEY, M. (Ed.). *GI-Jahrestagung*. Stuttgart, Germany: DBLP, 2014. (Data Management in the Cloud, v. 44).