Bioinformatics Advance Access published August 13, 2016

Subject Section

CGDM: Collaborative Genomic Data Model for Molecular Profiling Data Using NoSQL

Shicai Wang[1], **Mihaela A. Mares**[1] **and Yi-ke Guo**[1, 2,*]

[1]Data Science Institute, Imperial College London, London, UK and
[2]School of Computer Science, Shanghai University, Shanghai, China.

*To whom correspondence should be addressed.

Associate Editor: Dr. Jonathan Wren

Abstract

Motivation: High-throughput molecular profiling has greatly improved patient stratification and mechanistic understanding of diseases. With the increasing amount of data used in translational medicine studies in recent years, there is a need to improve the performance of data warehouses in terms of data retrieval and statistical processing. Both relational and Key Value models have been used for managing molecular profiling data. Key Value models such as SeqWare have been shown to be particularly advantageous in terms of query processing speed for large datasets. However, more improvement can be achieved, particularly through better indexing techniques of the Key Value models, taking advantage of the types of queries which are specific for the high-throughput molecular profiling data.
Results: In this paper, we introduce a Collaborative Genomic Data Model (CGDM), aimed at significantly increasing the query processing speed for the main classes of queries on genomic databases. CGDM creates three Collaborative Global Clustering Index Tables (CGCITs) to solve the velocity and variety issues at the cost of limited extra volume. Several benchmarking experiments were carried out, comparing CGDM implemented on HBase to the traditional SQL data model (TDM) implemented on both HBase and MySQL Cluster, using large publicly available molecular profiling datasets taken from NCBI and HapMap. In the microarray case, CGDM on HBase performed up to 246 times faster than TDM on HBase and 7 times faster than TDM on MySQL Cluster. In single nucleotide polymorphism (SNP) case, CGDM on HBase outperformed TDM on HBase by up to 351 times and TDM on MySQL Cluster by up to 9 times.
Contact: y.guo@imperial.ac.uk

1 Introduction

Molecular profiling refers to the study of specific patterns or signatures, such as DNA polymorphism, mRNA gene expression profiling, RNA profiling, proteomics and metabolic polymorphism. Biomedical research is moving towards using more high-throughput molecular profiling data to improve disease understanding.

A typical molecular profiling database contains values of features from individuals of interest or samples, to be used for several medical studies. A subject of study is usually a particular sample which we want to analyze for the specific study. At high level, queries on bioinformatics databases are meant to retrieve subjects based on their features as searching conditions. We investigated massive molecular profiling data based analysis applications, such as Van't Veer *et al.* (2002), Cross and Burmester (2004), Sotiriou *et al.* (2006) and Bissonnette *et al.* (2016), and participated in popular European medical projects, such as U-BIOPRED (Unbiased BIOmarkers in PREDiction of respiratory disease outcomes) (Wheelock *et al.*, 2013) and eTRIKS (European Translational Information and Knowledge Management Services) (Pandis *et al.*, 2015). We found three most frequently used steps for disease understanding. The first step is the marker selection. The next step is to discover the relationship between the selected markers and a particular disease. If the relationship is validated, the last step is to detect potential patients with the disease based on their markers. According to the three steps, we consider the three classes of queries for disease understanding:

In the first step, multiple cohorts with all markers in the database are collected for a particular study, for example "asthma".

```
select * from table
where subjects in {the cohorts}
and study = 'asthma';
--Query 1
```

In the second step, the selected markers of all the samples in related studies are retrieved to discover the relationship between the markers and the disease. For example, we may want to retrieve the samples genotyped at a particular DNA position in a "asthma" study.

```
select * from table
where markers in {the DNA positions}
and  study = 'asthma';
--Query 2
```

If a relationship between features and diseases is validated, all the samples with these markers in the database are selected to detect potential patients for a particular disease. For example, we may want to retrieve all the samples genotyped at a particular DNA position.

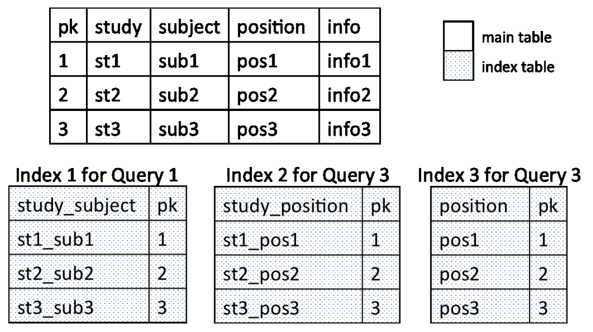
```
select * from table
where markers in {the DNA position};
--Query 3
```

Many management systems can enable scientists to do that, such as tranSMART (Athey *et al.*, 2013), NCBI (Barrett *et al.*, 2013) and SeqWare (O'Connor *et al.*, 2010). The storage of the first two platforms uses an SQL model and the last one uses a Key Value model. TranSMART is a biomedical data warehouse and analytics software platform that integrates clinical and genomics data using SQL models. The NCBI dbSNP (Sherry *et al.*, 2001) individual genotype data schema is a widely used relational data model for SNP data. The SeqWare implements a Key Value model based on HBase with secondary indices.

In the big data era, molecular profiling data size increases sharply due to new biological techniques, such as next generation sequencing. None of the existing databases work well whilst considering the three "V" features of big data (Volume, Variety, and Velocity). An alternative solution, and what forms the contribution described in this paper, is to take advantage of emerging NoSQL techniques with high speed indices to speed up queries over molecular profiling data.

Google Bigtable (Chang *et al.*, 2008) is a NoSQL database to store large volumes of structured data in the range of petabytes across thousands of machines. The database model of Bigtable is a set of processors known as clusters. Each cluster controls a set of table partitions. A table in Bigtable is a sparse, distributed, persistent and dynamic sorted tree, known as Log-structured merge-tree (LSM-tree) (OÕNeil *et al.*, 1996), and the data is organized into three dimensions: rows, columns, and timestamps. Many open source projects are implemented based on the Bigtable design, such as Accumulo (Sen *et al.*, 2013), Cassandra (Lakshman and Malik, 2010), HBase (George, 2011), Hypertable (Khetrapal and Ganesh, 2006), Druid (Yang *et al.*, 2014) and Open Neptune (Atzori and Dessì, 2011).

Other NoSQL databases, such as MongoDB (Chodorow, 2013), Couchbase (Brown, 2012), Redis (Carlson, 2013) and Memcached (Petrovic, 2008), MemcacheDB (Tudorica and Bucur, 2011) and ClusterPoint (Rats and Ernestsons, 2013) have also become a popular solution for managing large volumes of data. For example, MongoDB is a cross-platform document-oriented database, which avoids the traditional relational database structure in favor of JSON-like documents with dynamic schemas (BSON), making the integration of data in certain types of applications easier and faster.



pk	study	subject	position	info
1	st1	sub1	pos1	info1
2	st2	sub2	pos2	info2
3	st3	sub3	pos3	info3

☐ main table
▨ index table

Index 1 for Query 1

study_subject	pk
st1_sub1	1
st2_sub2	2
st3_sub3	3

Index 2 for Query 3

study_position	pk
st1_pos1	1
st2_pos2	2
st3_pos3	3

Index 3 for Query 3

position	pk
pos1	1
pos2	2
pos3	3

Fig. 1. Traditional molecular profiling data model

Moreover, range scan query in Bigtable can perform even faster than indices of other NoSQL databases for molecular profiling data. For example, Wang *et al.* (2014a) indicates HBase data query performs up to 7x faster than MongoDB for microarray data. Yet queries by non-row-key columns in Bigtable can be extremely slow due to random read operations usually being slower than range scan operations. Thus, in Bigtable the secondary index (Liu and Yoneda, 2001) has a smaller space requirement but performs slow; the clustering index (Zou *et al.*, 2010) has larger storage overhead but performs fast. The global index has high network traffic but is easy to implement; the local index has low network traffic but is difficult to implement. Feng *et al.* (2015) did a performance test on all those indices. The global clustering index performs best with a reasonable high storage overhead.

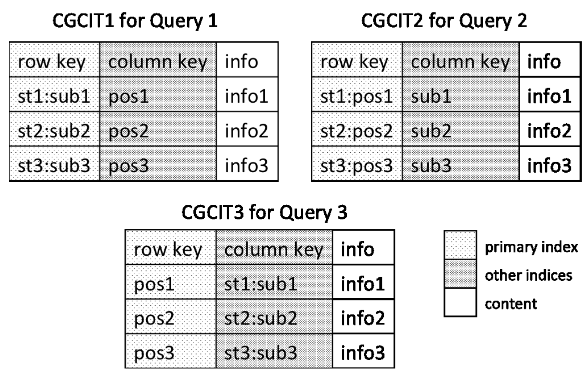
However, none of existing molecular profiling data models work well for Bigtable. The distinctive characteristic of our approach is to create a collaborative genomic data model (CGDM), which uses three complementary global clustering index tables (CGCITs) for each of the classes of queries described above. Each CGCIT uses vertical partitions to further speed up the queries and increase the data availability. Thus, CGDM can solve the velocity and variety issues with limited extra volume cost. We index and order each table based on the specific selection criteria of the query (i.e. study, subject, DNA position) and then distribute the queries to the right table based on the specific class or selection criteria it belongs to. Two benchmarks for gene expression and SNP are used to test. The experiments show CGDM implemented in HBase greatly outperforms the traditional model implemented in both HBase and MySQL Cluster.

2 Methods

2.1 Traditional secondary index for the queries

A typical traditional molecular profiling data model (TDM) used in tranSMART includes a main table and three secondary index tables, as shown in Figure 1. The pk is a unique id. Main table stores data ordered by the id. Three secondary indices are created to speed up the three typical queries above. The subject id is usually associated with its study, so study and subject together can be a unique id for Query 1. The combination of study and DNA position is the fastest index to locate a DNA postion in a specific study for Query 2. The position alone can be used to search information on a specific DNA position in the database for Query 3.

There are two types of secondary indices that can be used for this purpose. One stores pk values in the main table, while the pk column in the other index stores pointers linking to the physical location in the main table. In a distributed database, it is very difficult to maintain the second type of index. For example, if the distributed table blocks are merged



CGCIT1 for Query 1

row key	column key	info
st1:sub1	pos1	info1
st2:sub2	pos2	info2
st3:sub3	pos3	info3

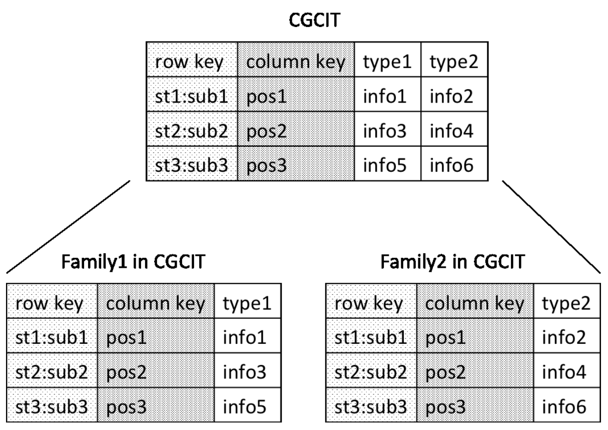
CGCIT2 for Query 2

row key	column key	info
st1:pos1	sub1	info1
st2:pos2	sub2	info2
st3:pos3	sub3	info3

CGCIT3 for Query 3

row key	column key	info
pos1	st1:sub1	info1
pos2	st2:sub2	info2
pos3	st3:sub3	info3

Fig. 2. CGDM structure.



CGCIT

row key	column key	type1	type2
st1:sub1	pos1	info1	info2
st2:sub2	pos2	info3	info4
st3:sub3	pos3	info5	info6

Family1 in CGCIT

row key	column key	type1
st1:sub1	pos1	info1
st2:sub2	pos2	info3
st3:sub3	pos3	info5

Family2 in CGCIT

row key	column key	type2
st1:sub1	pos1	info2
st2:sub2	pos2	info4
st3:sub3	pos3	info6

Fig. 3. Structure of multiple types of data.

or split, all the related physical locations in the pk column will be re-calculated. Thus, the pk in the index table usually contains the pk value in the main table. That means that if we sequentially search the index table to find the pk, we need to randomly search the pk in the main table to find the data. The random search leads to a big challenge to design a multi-dimension index for a large range query in Bigtable.

2.2 An optimized cluster index for Bigtable

A clustering index may be introduced to solve the random search problem from the secondary index. But the clustering index leads to a big storage overhead, as each index table is a full copy of the main table. In order to reduce the storage overhead, the backup copies in the database can be re-used. Fortunately, in Bigtable there are usually three copies of each table to make sure the data consistence and fault toleration. A main table with two index tables does not increase any overhead. Further more, if the main table can be even re-used, three collaborating clustering index tables can be created for the 3 typical queries.

2.3 Collaborative global clustering index table

The remaining work consists in presenting the design of the index for each query. From the analysis point of view, the study, subject and the DNA position are frequently used as search conditions; From the database point of view, Bigtable offers a hierarchical composite primary key, including row key (primary index) and column key (assistant index). Bigtable horizontally partitions a table by row key and and vertically partitions the table by column key. The row key is the primary index of the key, so the main index columns in this index table are stored here. The other index columns can be added in the column key. Based on the principles above, a collaborative data model for molecular profiling data is created, as shown in Figure 2.

2.4 Column Family separating different types of data

Except the usage of horizontal index, CGDM also considers utilizing the Bigtable vertical partition feature, i.e. Family, to further speed up data query, as shown in Figure 3. Family is a special concept, related to a group of columns that contain the same type of data. The same type of data is usually stored together and retrieved together. For example, three different types of gene expression data may be stored, such as raw value, logarithm value and zscore value, and a normal query only focuses on one type of data. If three Families are used for the three types of data, the query could be speeded up by searching about 1/3 of the data only.

2.5 The space overhead estimation

The disadvantage of Clustering index is the space overhead. However, the CGDM design uses an even smaller space than the secondary index and that is due to the optimization.

The secondary index space includes the main table and three indices. Each table has three copies of data. The space of each record S_i is the sum of Main table space S_m and three index table space S_{i1}, S_{i2} and S_{i3}.

$$S_m = (L_{pk} + L_{study} + L_{subject} + L_{position} + L_{info}) * 3 \quad (1)$$

$$S_i1 = (L_{study} + L_{subject} + L_{pk}) * 3 \quad (2)$$

$$S_i2 = (L_{study} + L_{position} + L_{pk}) * 3 \quad (3)$$

$$S_i3 = (L_{position} + L_{pk}) * 3 \quad (4)$$

$$\begin{aligned} S_i &= S_m + S_{i1} + S_{i2} + S_{i3} \\ &= 12 * L_{pk} + 9 * L_{study} + 6 * L_{subject} + 9 * L_{position} \quad (5) \\ &\quad + 3L_{info} \end{aligned}$$

Where L_{pk} is the length of primary key, L_{study} is the length of study column, $L_{position}$ is the length of DNA position column, L_{info} is the length of non-indexed columns and data replica factor is 3.

The CGDM space S_c consists of the space of three CGCITs S_{c1}, S_{c2}, S_{c3}.

$$S_{c1} = (L_{study} + L_{subject} + L_{position}) * N + L_{info} \quad (6)$$

$$S_{c2} = (L_{study} + L_{position} + L_{subject}) * N + L_{info} \quad (7)$$

$$S_{c3} = (L_{position} + L_{subject} + L_{study})N + L_{info} \quad (8)$$

$$\begin{aligned} S_c &= S_{c1} + S_{c2} + S_{c3} \\ &= 3N * L_{study} + 3N * L_{subject} + 3N * L_{position} + 3 * L_{info} \end{aligned} \quad (9)$$

Where N is the number of Family in each CGCIT.

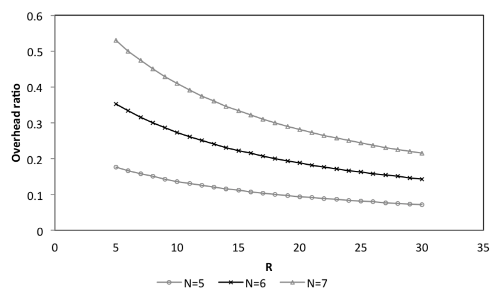


Fig. 4. The overhead ratio of CGDM to TDM.

The overhead ratio of CGDM to TDM is:

$$\begin{aligned}(S_c - S_i)/S_i = {} & ((N-3) * L_{study} + (N-2) * L_{subject} \\ & + (N-3) * L_{position} - 4 * L_{pk})/ \\ & (4 * L_{pk} + 3 * L_{study} + \\ & 2 * L_{subject} + 3 * L_{position} + L_{info})\end{aligned} \quad (10)$$

Assume pk, study, subject and position have the same length L then, the ratio of the info length to any other index column length R is:

$$R = L_{info}/L \quad (11)$$

Then, the overhead ratio becomes:

$$(S_c - S_i)/S_i = (3N - 12)/(12 + R) \quad (12)$$

If N<5, there will not be any overhead. For N from 5 to 7, we plotted the Figure 4. The overhead ratio drops significantly as the R increases and the N decreases, which indicates that CGDM should have less columns to index and all index columns should have small length to avoid big space overhead. If N changes from 5 to 7 and the R changes from 10 to 30, then the overhead changes from 7.14% to 52.94%.

2.6 Fault Tolerance

In CGDM, CGCITs will have no replica to avoid huge storage overhead, and cause the problem of fault tolerance. The basic idea is that CGCITs will replicate and recover each other at record level. As both CGCIT1 and CGCIT2 row keys begin with study column, rows in CGCIT1 and CGCIT2 are both ordered by study. When a region of a CGCIT1 is damaged, we can quarantine the regions of the studies involved and use regions containing the same studies in CGCIT2 to reconstruct the regions. For CGCIT2, the situation is similar. For CGCIT3, we can only isolate DNA positions involved and fetch corresponding information by scanning the whole CGCIT2. During the scan, we can use DNA position range to skip unnecessary regions.

3 Results

3.1 Experiment Environment

There are many implementations of Bigtable, such as Hypertable (Khetrapal and Ganesh, 2006), Apache HBase (George, 2011) and Apache Cassandra Lakshman and Malik (2010). We chose the most popular one - HBase. A monolithic relational database is difficult to be compared to HBase clusters, but many relational database clusters can be used for this purpose, such as MySQL Cluster (Ronstrom and Thalmann, 2004), PostgreSQL XE (Momjian, 2001) and SQL Server Cluster (Campbell *et al.*, 2010). Compared to other database clusters, MySQL Cluster, supported by Oracle, is one of the most powerful and easy to deploy clusters.

Both HBase and MySQL Cluster were running on a OpenStack platform Sefraoui *et al.* (2012). HBase was used to implement both the TDM and CGDM, while MySQL Cluster implemented TDM. Each database was configured as follows:

HBase (Version 0.96.0 on Hadoop 1.0.3): One master server node and three slave nodes with HBase configured in fully distributed mode. The master server was configured as a virtual machine (VM) with 4 CPU cores and 8 GB memory, while each slave node was configured as VMs with 4 CPU cores and 8 GB memory. Each VM used a 100 GB disk. Three copies is the minimum number for the Hadoop Distributed File System to guarantee data consistency.

MySQL Cluster (Version 5.6.11-ndb-7.3.2): Four VMs, with one as a manager node and three data nodes. The manager node consists of a MGM, a MySQL and a NDB using a VM with 4 CPU cores and 8 GB memory. Each VM used a 100 GB disk. Each data node consisted of a NDB. Two data copies were used to guarantee its data consistency.

3.2 Microarray benchmark

Our experiment was performed using a public Multiple Myeloma (MULTMYEL) (Raab *et al.*, 2009; Hanamura *et al.*, 2006) dataset [GEO:GSE24080] (Shi *et al.*, 2010). The reason we chose MULTMYEL as our test dataset was that it is one of the largest datasets, consisting of 559 samples and 54,675 probesets for each sample, totalling approximately 30.5 million records. In each record there are three columns in the info part, the raw value and two normalized values (logarithm and zscore). These values are usually used separately for different purposes, so three data types (Family) are set for the three values.

3.2.1 Query 1 Evaluation

We refer to the use cases and part of the results from the paper (Wang *et al.*, 2014b) and add an new baseline test to see how TDM works for HBase. The searching conditions in these use cases are mostly related to the subject, so CGCIT1 is selected to perform the tests. In Figure 5, CGDM in HBase demonstrates an average 5.24 and 73.89 times of increase compared to TDM implemented on MySQL Cluster and HBase.

Compared to TDM on HBase, in 5/11 of cases, CGDM performs more than 80x faster than TDM on HBase. In 3/11 of cases, CGDM is more than 90x faster. With the subject number increasing, the CGDM's advantage is more obvious. In the last and biggest case, CGDM outperforms the traditional one by 101x.

Compared to TDM on HBase, in 6/11 of cases, CGDM is more than 5x faster than traditional model on MySQL Cluster. In 10/11 of cases, CGDM is more than 4x faster than TDM on MySQL Cluster. In the worst case, A2, CGDM is more than 3.61 times faster than traditional model on MySQL Cluster.

3.2.2 Query 2 Evaluation

A normal marker selection returns about 100 potential probe sets, which need to be verified by other datasets of similar studies. Based on the design CGCIT1 will perform sub-standard for this purpose, but CGCIT2 can perform much faster. We randomly generated 100 probes and test CGDM against the one. The query of Key Value model is based on the Random Read operation.

CGDM demonstrates an average 174.52 times of increase compared to TDM implemented on HBase. In 4/10 of cases, CGDM on HBase is more than 170 times faster than TDM on HBase. In 9/10 of cases, CGDM on HBase performs 140 times faster than TDM on HBase.

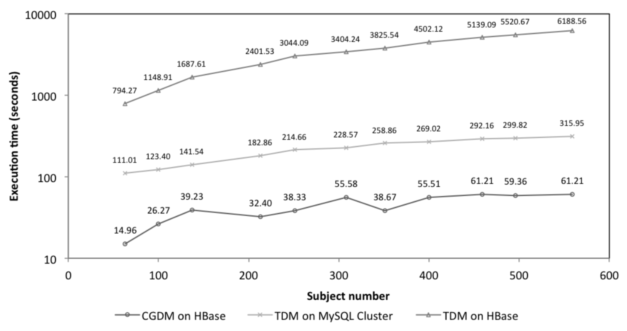


Fig. 5. Result of Query 1.

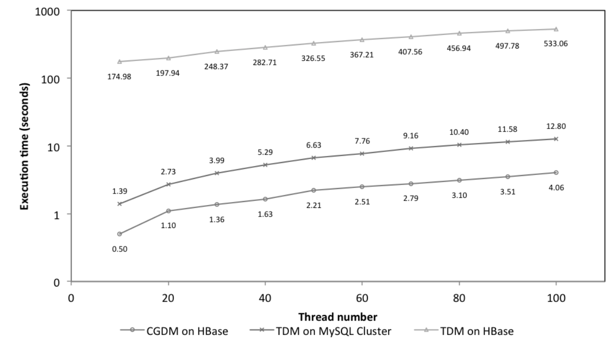


Fig. 6. Result of Query 2.

CGDM demonstrates an average 3.06 times of increase compared to TDM implemented on MySQL Cluster. In 6/10 of cases, CGDM is more than 3 times faster than traditional model on MySQL Cluster. In 9/10 of cases, CGDM is more than 2.5 times faster than TDM on MySQL Cluster. In the worst case, 20-thread, CGDM is more than 2.47 times faster than traditional model on MySQL Cluster. HBase performs as stable as MySQL Cluster in most of the cases, as show by the greater average result deviations in Figure 6.

3.2.3 Query 3 Evaluation

A further validation is performed using three GSE24080 studies. We copied GSE24080 dataset twice and arbitrarily named them GSE24081 and GSE24082 and re-used the 100 probe sets in the query by position experiment. MySQL Cluster failed to load these three datasets due to a memory limitation. We compared the query using both CGDM and TDM on HBase.

CGDM demonstrates an average 246.48 times of increase compared to TDM, as shown in Figure 7. In 6/10 of cases, CGDM is more than 220 times faster than TDM. In 3/10 of cases, CGDM is more than 250 times faster than TDM. In largest case, CGDM is more than 210 times faster than TDM.

3.3 SNP benchmark

The large Microarray benchmark shows the performance of CGDM throughput, without considering the response time for small queries. If only one subject or a few DNA position are required, a large overhead may damage the users' experience. The SNP benchmark not only illustrates the throughput of CGDM but also its response time of small queries. Also, the TDM on HBase performs much worse than the other two implementations

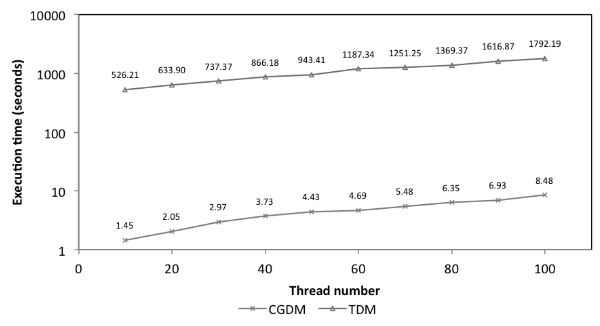


Fig. 7. Result of Query 3.

in all the tests above. So, the TDM on HBase will not be performed any more.

The experiments are performed using three SNP datasets (ASW, 87 samples and 119,487 SNPs; CEU, 165 samples and 119,487 SNPs; CHB, 137 samples and 317,642 SNPs) from the 2010-05 phase III consensus datasets of HapMap project (Consortium, 2007). This SNP dataset has only one type of data.

3.3.1 Query 1 Evaluation

Usually, the first step of SNP based medical research is to find significant differential SNPs with clinical information in a study. All available SNPs of the specified subjects with typical features are loaded for further tests to find the expected SNPs. Thus, the first query is usually applied on the main cohorts (almost all subject) at the beginning of a study to find the most relevant SNPs. For this type of case, we searched all subjects in each dataset and as shown in Figure 8, CGDM on HBase is on average 269.39 times faster than TDM on HBase. CGDM on HBase outperforms TDM on HBase by 351.33 times when tested with CHB dataset, while when using datasets ASW and CEU, TDM it performs 215.15 and 241.68 times faster than TDM on HBase. CGDM on HBase performed about 4 or more times faster for the data retrieval time than TDM on MySQL. Particularly in ASW, CGDM on HBase outperforms TDM on MySQL Cluster by 9.12 times. While in datasets CEU and CHB, CGDM respectively performed 5.12 and 4.51 times faster than TDM on MySQL Cluster.

In the response time test of one subject queries over 119,487 SNPs, CGDM on HBase is 148.25 times faster than TDM on HBase on average, as shown in Figure 9, while the queries over 317,462 SNPs using CGDM outperformed TDM on MySQL Cluster by 99.20 times. The response time of 119,487 SNPs queries with CGDM are 2.61 times faster than TDM on MySQL Cluster on average, while the queries over 317,462 SNPs using CGDM outperformed TDM on MySQL Cluster by 1.57 times.

3.3.2 Query 2 and 3 Evaluation

After significant differential SNPs are calculated, the top hundred differential SNPs are usually tested using other datasets to confirm or further reduce the number of these SNPs. If SNPs associated with certain diseases are found, queries over several SNPs across datasets are generated to detect whether a patient is vulnerable to certain diseases based on his genotype information from the SNPs.

While querying 100 discrete SNP by position, CGDM on HBase performs on average 15.40 times faster than TDM on HBase. But TDM on MySQL Cluster is 1.84 times faster than CGDM on HBase on average, as shown in Figure 10, due to the slow HBase Random Read.

However, with concurrent query thread number increasing, MySQL Cluster does not scale well, as shown in Figure 11. CGDM on HBase demonstrates an average 4.66 times of increase in query processing speed

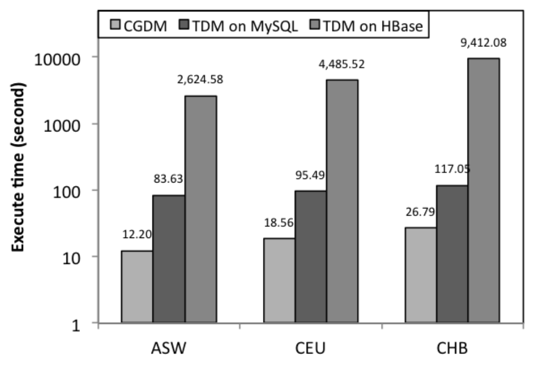


Fig. 8. Result of Query 1 for throughput test.

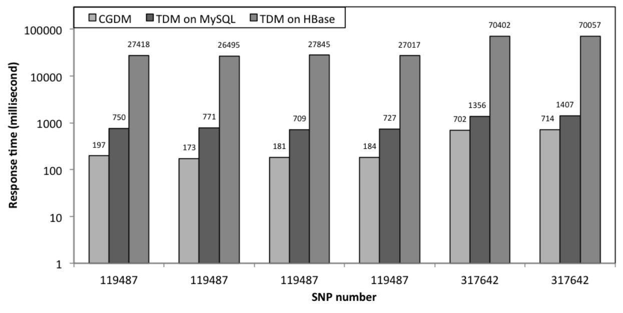


Fig. 9. Result of Query 1 for response time test.

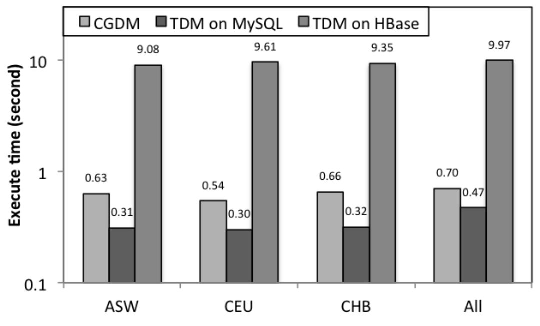


Fig. 10. Result of Query 2 and 3 for response time test.

compared to TDM implemented on MySQL Cluster. In 6/10 of cases, CGDM is more than 5 times faster than TDM on MySQL Cluster. In the worst case, when using only 10 threads, CGDM is more than 3.07 times faster than TDM on MySQL Cluster. Compared to TDM on HBase, CGDM performs 105.40 times faster on average. Especially, in 6/10 of cases, CGDM is more than 100 times faster than TDM on HBase.

4 Discussion

The most common queries for disease understanding do not cover all the commonly used queries, for example wildcard query. Generally, there are two main types of wildcard queries. One begins with a wildcard (%search-string...). This query will perform a scan operation on a full or large range of the table. The other one starts with a search string (search-string%...). This query will perform a relatively small seek operation based on the

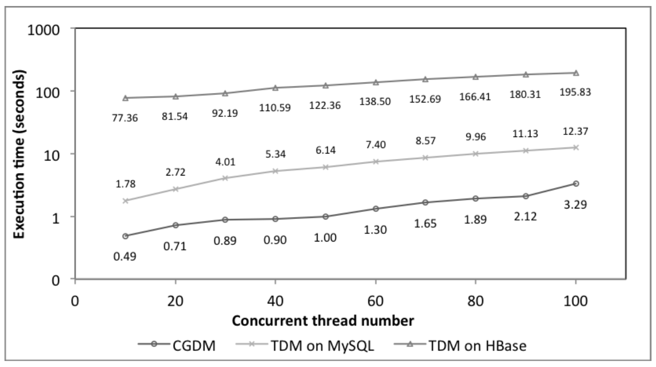


Fig. 11. Result of Query 3 for throughput test.

prefix 'search-string'. Though tests for wildcard query are not specified, some of the results show some hints about how CGDM may perform. The last query in section 3.2.1, which performed a scan on the whole table, indicates the performance of a test similar to (%search-string...). The query tests in section 3.2.3 performed queries similar to queries with wildcard (GSE2408%) on HBase. Due to the hierarchical design of CGCIT, CGCIT has three types of wildcard queries, the wildcard query on primary index, the wildcard query on other indices and the wildcard query on content. The last query in section 3.2.1 shows a wildcard query on primary key and the queries in section 3.2.3 indicates wildcard queries on other indices. Any one of these three wildcard queries may not be the best choice for certain cases, while a mixed dynamically wildcard query plan may perform better than statically choosing one.

5 Conclusion

In this paper, CGDM has been created for high-throughput molecular profiling data to improve the query processing speed for the three main classes of queries on genomic databases. Multiple benchmarking experiments were carried out, comparing CGDM implemented on HBase to TDM implemented on both HBase and MySQL Cluster, using large publicly available molecular profiling datasets. In the microarray case, CGDM on HBase performed up to 246 times faster than TDM on HBase and 7 times faster than TDM on MySQL Cluster. In the SNP case, CGDM on HBase outperformed TDM on HBase by up to 351 times and TDM on MySQL Cluster by up to 9 times.

Funding

This research is partially supported by the Innovative R&D Team Support Program of Guangdong Province (NO. 201001D0104726115), China, Johnson & Johnson Pharmaceutical and Research Comp, the European Commission Innovative Medicines Initiative (IMI) under the European Translational Information & Knowledge Management Services project (eTRIKS: Grant Agreement NO. 115446).

References

Athey, B. D., Braxenthaler, M., Haas, M., and Guo, Y. (2013). tranSMART: An Open Source and Community-Driven Informatics and Data Sharing Platform for Clinical and Translational Research. *AMIA Summits on Translational Science proceedings AMIA Summit on Translational Science*, **2013**, 6–8.

Atzori, M. and Dessì, N. (2011). Dataspaces: Where structure and schema meet. In *Learning Structure and Schemas from Documents*, pages 97–119. Springer.

Barrett, T., Wilhite, S. E., Ledoux, P., Evangelista, C., Kim, I. F., Tomashevsky, M., Marshall, K. a., Phillippy, K. H., Sherman, P. M., Holko, M., Yefanov, A., Lee, H., Zhang, N., Robertson, C. L., Serova, N., Davis, S., and Soboleva, A. (2013). NCBI GEO: archive for functional genomics data sets–update. *Nucleic acids research*, **41**(Database issue), D991–5.

Bissonnette, R., Suárez-Fariñas, M., Li, X., Bonifacio, K. M., Brodmerkel, C., Fuentes-Duculan, J., and Krueger, J. G. (2016). Based on molecular profiling of gene expression, palmoplantar pustulosis and palmoplantar pustular psoriasis are highly related diseases that appear to be distinct from psoriasis vulgaris. *PloS one*, **11**(5), e0155215.

Brown, M. C. (2012). *Getting Started with Couchbase Server*. " O'Reilly Media, Inc.".

Campbell, D. G., Kakivaya, G., and Ellis, N. (2010). Extreme scale with full sql language support in microsoft sql azure. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 1021–1024. ACM.

Carlson, J. L. (2013). *Redis in Action*. Manning Publications Co.

Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A., and Gruber, R. E. (2008). Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, **26**(2), 4.

Chodorow, K. (2013). *MongoDB: the definitive guide*. " O'Reilly Media, Inc.".

Consortium, T. I. H. (2007). A second generation human haplotype map of over 3.1 million SNPs. *Nature*, **449**, 851–61.

Cross, D. and Burmester, J. K. (2004). The promise of molecular profiling for cancer identification and treatment. *Clinical medicine & research*, **2**(3), 147–150.

Feng, C., Yang, X., Liang, F., Sun, X.-H., and Xu, Z. (2015). Lcindex: A local and clustering index on distributed ordered tables for flexible multi-dimensional range queries. In *Parallel Processing (ICPP), 2015 44th International Conference on*, pages 719–728. IEEE.

George, L. (2011). *HBase: the definitive guide*. " O'Reilly Media, Inc.".

Hanamura, I., Huang, Y., Zhan, F., Barlogie, B., and Shaughnessy, J. (2006). Prognostic value of cyclin d2 mrna expression in newly diagnosed multiple myeloma treated with high-dose chemotherapy and tandem autologous stem cell transplantations. *Leukemia*, **20**(7), 1288–1290.

Khetrapal, A. and Ganesh, V. (2006). HBase and Hypertable for large scale distributed storage systems A Performance evaluation for Open Source BigTable Implementations. *Evaluation*, page 8.

Lakshman, A. and Malik, P. (2010). Cassandra: a decentralized structured storage system. *ACM SIGOPS Operating Systems Review*, **44**, 35.

Liu, L.-C. H. and Yoneda, K. (2001). Secondary index search. US Patent 6,266,660.

Momjian, B. (2001). *PostgreSQL: introduction and concepts*, volume 192. Addison-Wesley New York.

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

OÕNeil, P., Cheng, E., Gawlick, D., and OÕNeil, E. (1996). The log-structured merge-tree (LSM-tree).

Pandis, I., Guo, Y., Guitton, F., Yang, X., Sun, K., Wang, S., Jullian, N., Sousa, A., Bansal, A., Corfield, J., *et al.* (2015). etriks it platfroms for large-scale biomedical research. *European Respiratory Journal*, **46**(suppl 59), PA3976.

Petrovic, J. (2008). Using memcached for data distribution in industrial environment. In *ICONS*, pages 368–372.

Raab, M. S., Podar, K., Breitkreutz, I., Richardson, P. G., and Anderson, K. C. (2009). Multiple myeloma. *Lancet*, **374**, 324–339.

Rats, J. and Ernestsons, G. (2013). Clustering and ranked search for enterprise content management. *International Journal of E-Entrepreneurship and Innovation (IJEEI)*, **4**(4), 20–31.

Ronstrom, M. and Thalmann, L. (2004). Mysql cluster architecture overview. *MySQL Technical White Paper*.

Sefraoui, O., Aissaoui, M., and Eleuldj, M. (2012). Openstack: toward an open-source solution for cloud computing. *International Journal of Computer Applications*, **55**(3), 38–42.

Sen, R., Farris, A., and Guerra, P. (2013). Benchmarking apache accumulo bigdata distributed table store using its continuous test suite. In *Big Data (BigData Congress), 2013 IEEE International Congress on*, pages 334–341. IEEE.

Sherry, S. T., Ward, M. H., Kholodov, M., Baker, J., Phan, L., Smigielski, E. M., and Sirotkin, K. (2001). dbSNP: the NCBI database of genetic variation. *Nucleic acids research*, **29**, 308–311.

Shi, L., Campbell, G., Jones, W. D., Campagne, F., Wen, Z., Walker, S. J., Su, Z., Chu, T.-M., Goodsaid, F. M., Pusztai, L., Shaughnessy, J. D., Oberthuer, A., Thomas, R. S., Paules, R. S., Fielden, M., Barlogie, B., Chen, W., Du, P., Fischer, M., Furlanello, C., Gallas, B. D., Ge, X., Megherbi, D. B., Symmans, W. F., Wang, M. D., Zhang, J., Bitter, H., Brors, B., Bushel, P. R., Bylesjo, M., Chen, M., Cheng, J., Cheng, J., Chou, J., Davison, T. S., Delorenzi, M., Deng, Y., Devanarayan, V., Dix, D. J., Dopazo, J., Dorff, K. C., Elloumi, F., Fan, J., Fan, S., Fan, X., Fang, H., Gonzaludo, N., Hess, K. R., Hong, H., Huan, J., Irizarry, R. A., Judson, R., Juraeva, D., Lababidi, S., Lambert, C. G., Li, L., Li, Y., Li, Z., Lin, S. M., Liu, G., Lobenhofer, E. K., Luo, J., Luo, W., McCall, M. N., Nikolsky, Y., Pennello, G. A., Perkins, R. G., Philip, R., Popovici, V., Price, N. D., Qian, F., Scherer, A., Shi, T., Shi, W., Sung, J., Thierry-Mieg, D., Thierry-Mieg, J., Thodima, V., Trygg, J., Vishnuvajjala, L., Wang, S. J., Wu, J., Wu, Y., Xie, Q., Yousef, W. A., Zhang, L., Zhang, X., Zhong, S., Zhou, Y., Zhu, S., Arasappan, D., Bao, W., Lucas, A. B., Berthold, F., Brennan, R. J., Buness, A., Catalano, J. G., Chang, C., Chen, R., Cheng, Y., Cui, J., Czika, W., Demichelis, F., Deng, X., Dosymbekov, D., Eils, R., Feng, Y., Fostel, J., Fulmer-Smentek, S., Fuscoe, J. C., Gatto, L., Ge, W., Goldstein, D. R., Guo, L., Halbert, D. N., Han, J., Harris, S. C., Hatzis, C., Herman, D., Huang, J., Jensen, R. V., Jiang, R., Johnson, C. D., Jurman, G., Kahlert, Y., Khuder, S. A., Kohl, M., Li, J., Li, M., Li, Q.-Z., Li, S., Li, Z., Liu, J., Liu, Y., Liu, Z., Meng, L., Madera, M., Martinez-Murillo, F., Medina, I., Meehan, J., Miclaus, K., Moffitt, R. A., Montaner, D., Mukherjee, P., Mulligan, G. J., Neville, P., Nikolskaya, T., Ning, B., Page, G. P., Parker, J., Parry, R. M., Peng, X., Peterson, R. L., Phan, J. H., Quanz, B., Ren, Y., Riccadonna, S., Roter, A. H., Samuelson, F. W., Schumacher, M. M., Shambaugh, J. D., Shi, Q., Shippy, R., Si, S., Smalter, A., Sotiriou, C., Soukup, M., Staedtler, F., Steiner, G., Stokes, T. H., Sun, Q., Tan, P.-Y., Tang, R., Tezak, Z., Thorn, B., Tsyganova, M., Turpaz, Y., Vega, S. C., Visintainer, R., von Frese, J., Wang, C., Wang, E., Wang, J., Wang, W., Westermann, F., Willey, J. C., Woods, M., Wu, S., Xiao, N., Xu, J., Xu, L., Yang, L., Zeng, X., Zhang, J., Zhang, L., Zhang, M., Zhao, C., Puri, R. K., Scherf, U., Tong, W., and Wolfinger, R. D. (2010). The MicroArray Quality Control (MAQC)-II study of common practices for the development and validation of microarray-based predictive models. *Nature biotechnology*, **28**(8), 827–38.

Sotiriou, C., Wirapati, P., Loi, S., Harris, A., Fox, S., Smeds, J., Nordgren, H., Farmer, P., Praz, V., Haibe-Kains, B., *et al.* (2006). Gene expression profiling in breast cancer: understanding the molecular basis of histologic grade to improve prognosis. *Journal of the National Cancer Institute*, **98**(4), 262–272.

Tudorica, B. G. and Bucur, C. (2011). A comparison between several nosql databases with comments and notes. In *Roedunet International Conference (RoEduNet), 2011 10th*, pages 1–5. IEEE.

Van't Veer, L. J., Dai, H., Van De Vijver, M. J., He, Y. D., Hart, A. A., Mao, M., Peterse, H. L., van der Kooy, K., Marton, M. J., Witteveen, A. T., *et al.* (2002). Gene expression profiling predicts clinical outcome of breast cancer. *nature*, **415**(6871), 530–536.

Wang, S., Pandis, I., Wu, C., He, S., Johnson, D., Emam, I., Guitton, F., and Guo, Y. (2014a). High dimensional biological data retrieval optimization with NoSQL technology. *BMC genomics*, **15 Suppl 8**(Suppl 8), S3.

Wang, S., Pandis, I., Wu, C., He, S., Johnson, D., Emam, I., Guitton, F., and Guo, Y. (2014b). High dimensional biological data retrieval optimization with nosql technology. *BMC genomics*, **15**(Suppl 8), S3.

Wheelock, C. E., Goss, V. M., Balgoma, D., Nicholas, B., Brandsma, J., Skipp, P. J., Snowden, S., Burg, D., D'Amico, A., Horvath, I., *et al.* (2013). Application of'omics technologies to biomarker discovery in inflammatory lung diseases. *European Respiratory Journal*, **42**(3), 802–825.

Yang, F., Tschetter, E., Léauté, X., Ray, N., Merlino, G., and Ganguli, D. (2014). Druid: a real-time analytical data store. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 157–168. ACM.

Zou, Y., Liu, J., Wang, S., Zha, L., and Xu, Z. (2010). CCIndex : A Complemental Clustering Index on Distributed Ordered Tables for Multi-dimensional Range. In *the 9th IFIP International Conference on Network and Parallel Computing*, pages 247–261.