Evaluation of Approximate Query Processing Systems

Solange Paz, MSc[1], Bruno Cabral, PhD[1,2], and Jorge Bernardino, PhD[3,2]
[1]University of Coimbra, Portugal, *spaz@student.dei.uc.pt*, *bcabral@dei.uc.pt*
[2]CISUC - Centre for Informatics and Systems of University Coimbra, Portugal
[3]Coimbra Polytechnic – ISEC, Coimbra, Portugal, *jorge@isec.pt*

***Abstract*– Today data-intensive systems, such as e-business. e-procurement e-government, e-commerce etc. systems present a huge amount of available data. One of the main issues of query processing is how to process queries efficiently. In many cases, it is impossible or too expensive for users to get exact answers in a short query response time. Approximate query processing (AQP) is an alternative way that returns approximate answer and is increasingly used, as millions of data are processed daily in a database. In this paper, we evaluate two of the latest AQP systems with the best results in the literature: VerdictDB and XDB. We test these systems according to the query response time and accuracy of results returned, using queries of the TPC-H benchmark with different sizes. The VerdictDB and XDB are good tools for large volumes of data. The experiments demonstrate that VerdictDB results can be 76x faster than MySQL. However, with the same query response time, XDB returns results with more accuracy.**

***Keywords*-- Approximate query processing, error rate, query response time.**

I. Introduction

In recent years Approximate Query Processing (AQP) is increasingly used, since the social media, mobile devices, and wireless sensors continue to create massive data volumes [1]. In a large company where this data increases daily is necessary to get answers quickly and accurately, in order to allow better decision making. The time to execute large data is too long. Currently users are waiting too long to get accurate results in large-scale data. Some users consider this time unacceptable, since the productivity of them is inhibited by slow and expensive data interaction. The goal of an AQP system is returning an answer in short response time. Thus, an approximate aggregate with a specified error warranty is an option to improve the productivity [2]. The AQP system allows users to change the accuracy of the query by the speed with which a response is returned in large data sets and using complex aggregation queries [3].

The applications of decision support and data mining often turn to aggregate functions, such as "SUM" and "AVG" to formulate a query. As these queries are performed in large databases, the time consumed is very high. In this type of systems, the accuracy of the results of the queries is not so relevant, as far as users prefer to know an approximate response instead of waiting too long to get an exact answer [4]. Approximate answers in these systems allow users to analyse data quickly and effectively.

To analyse data quickly, it is necessary to get an answer as accurately as possible in a short period of time. This is possible using approximate query processing techniques, once that reduce the response time of queries on online support systems, when it is not required nor important to obtain a very precise answer [5].

There are numerous techniques for processing approximate queries and can be categorized into two categories: online aggregation and offline synopses generation [6]. The AQP with random sampling as a basis is one of the most useful methods for the calculated of large quantities of data in databases efficiently [7]. The first generation of AQP was focused on online aggregation for simple OLAP queries. The second extended the scope to more complex workflows mainly by taking pre-calculated samples of costs, if most or all queries are beforehand known. The third generation cannot assume that most queries are known in advance, but instead can leverage that data exploration pipelines are incrementally created by the user with a visual interface [8].

For a large-scale data, a less accurate but instantaneous result is desirable. Query processing is the process that deduces information presented in the database. In many cases is impossible know how to process queries efficiently and obtain the exact answers as soon as possible [9]. The AQP consists in using a sample of the total data and processing queries with those sampled data.

The results of the approximate answer are better as more data is available and, if have time for the continue the processing the data converges for an exact answer [10]. The AQP is designed for aggregate queries such as using the AVERAGE, COUNT and SUM functions [11, 12].

We selected two of latest tools of AQP with better results in the literature review: VerdictDB and the XDB. Using TPC-H benchmark queries, we compare these tools using different database sizes of 1, 10, and 50 GB. We also compared the tools with the original DBMSs that support them, in order to understand the differences in speedup and data accuracy.

The main contributions of this paper are as follows: analyse the differences in terms of performance and error rate (accuracy) between the two tools and compare the performance of the tools in relation to MySQL and PostgreSQL.

The rest of this paper is structured as follows. Section 2 presents a literature review on approximate query processing and Section 3 describes the experimental setup, included the results of the experiment. Finally, Section 4 states the conclusions and proposes some future work.

II. Related Work

The sampling techniques from base relations in order to quickly estimate the answer to an aggregation query are the first that appear described in the research. However, these sampling techniques are not directly applicable to online aggregation. In online aggregation, an execution estimate is continuously updated based on the data known up to now. The error in this estimate is specified through a confidence interval [13]. One of the first proposals was ripple join, an online aggregation interface that permits users to observe in real time the progress of their aggregation queries and control the execution [14]. The wander join chooses the optimal plan for conducting the random walks without having to use and collect any statistics initially. Compared with ripple join, the wander join is more efficient for joins [15].

The XDB [16] integrates the wander join in PostgreSQL and comparing XDB with Turbo DBO. The comparing was made when there is enough memory and when there is no enough memory.

In [17] the authors presented a BlinkDB allows users to trade off query accuracy for response time, since it allows interactive queries with over massive data. BlinkDB presented results with error rating, being 200x faster than Hive within an error of 2-10%.

The VerdictDB [18] makes all communications with the backend database in SQL and return a very fast result. The authors show how this tool is being in 171x faster than other existing engines, for example, Spark SQL.

In [19] the authors proposed a framework for creating and running approximation-enable MapReduce programs. The ApproxHadoop reduces application runtime and energy consumption, returning answers with very small errors. This framework reduces runtimes by up to 32x with an error of 1% and 95% confidence.

In [20] the authors proposed the use of multi-dimensional wavelets as an effective tool for general-purpose approximate query processing. They compared the wavelets with sampling techniques and histograms. The performance of histograms is worse than that of wavelets. Comparing histograms and sampling, wavelets exhibit more than an improvement on the magnitude of the relative error.

Unlike previous papers, besides a comparison between two recent tools for AQP, in our work it is also a comparison with these tools and the original DBMS that support them.

III. Experimental Evaluation

In this section, we present the results of the experimental study conducted to evaluate the two selected tools: VerdictDB and XDB. Using different data from the TPC-H benchmark show what the best tool. We start this section with a description of the tools evaluated. Then we describe the experimental scenario used, being then presented the results and conclusions of the respective evaluation.

A. Analyzed Tools

We selected two recent AQP tools with better results in the literature review: VerdictDB [21] and XDB [22]. Although both tools provide approximate answers, they have differences and constraints. The XDB only support online aggregation and not support the ORDER BY or LIMIT clause. In this tool is mandatory to define a confidence interval and the time we want to wait until we get a query result. But also, is possible visualizing various results and stop the query when a response very close to the original is shown. In contrast, the VerdictDB support the normal syntax of query, including also the ORDER BY or LIMIT.

VerdictDB [23] is an AQP system that uses a middleware architecture that does not require changes to the database and can work with all the data management systems available in the market. The user sends the queries to the VerdictDB and gets the result from them without interacting with the database. VerdictDB then communicates with the database to obtain metadata and to access and process data. If there is a set of sample tables that can be used in place of each of the base tables, the sample tables in the query are used. Each query is converted into logical operators and then converted to another logical expression capable of executing in the AQP, in the end it is rewritten in an SQL statement, to be executed. VerdictDB uses sub-sampling as a technique to estimate error through variational subsampling. In variational subsampling the same row of the table may belong to several subsamples and all subsamples must be the same size. For each row of the sub-sample, only a random number is generated to determine which sub-sample belongs to, and then the aggregation is performed only once per row. You define which tables you want to build samples of, which are usually those that contain a larger set of data. When a query is performed, the Query Parser converts it to logical expressions such as joins, which are then converted by the AQP Rewriter into another logical expression that can be executed with AQP. The Syntax Changer converts this rewritten logical expression into an SQL statement that can be executed in the underlying database. At the end, after this rewritten query is executed, Answer Rewriter returns an approximate response to the original query.

XDB is an engine developed into PostgreSQL 9.4.2 to support the wander join algorithm in online aggregation queries. The idea for wander join is to model a join over k tables as a join graph, and then perform random walks in this graph. In this graph, each tuple is modelled as a vertex and there is an edge between two tuples if they can join, are have the same value on their common attribute. How can they exist various random walks for the same query, XDB have a mechanism who chooses the best walking plan. This mechanism consists of the execution of a series of trial runs, using multiple paths in order to find the best. After selecting the best walk path, are extracted samples of B-tree indexes, one by one. The counts of the subtrees of the internal nodes of indices are used to extract samples. The selection predicates

are applied when the related tuples are sampled. When the walk is complete, the executor returns the current estimators and confidence intervals. It returns an empty tuple when the time set in the query is reached, stating to PostgreSQL that no longer exist tuples available. XDB stores relationships between tables and indexes in main memory, which will affect their performance if there are too many tables of data.

B. *Experimental Setup*

To enable the public sharing of the results, we use the modified queries in [24] on the data set of the TPC-H benchmark [25]. These queries are presented in Table I.

TABLE I
TPC-H QUERIES MODIFIED.

Query number	Query
Q3	SELECT SUM(l_extendedprice * (1 - l_discount)) FROM customer, orders, lineitem WHERE c_mktsegment = 'BUILDING' AND c_custkey = o_custkey AND l_orderkey = o_orderkey;
Q7	SELECT SUM(l_extendedprice * (1 - l_discount)) FROM supplier, lineitem, orders, customer, nation n1, nation n2 WHERE s_suppkey = l_suppkey AND o_orderkey = l_orderkey AND c_custkey = o_custkey AND s_nationkey = n1.n_nationkey AND c_nationkey = n2.n_nationkey AND n1.n_name = 'CHINA';
Q10	SELECT SUM(l_extendedprice * (1 - l_discount)) FROM customer, lineitem, orders, nation WHERE c_custkey = o_custkey AND l_orderkey = o_orderkey AND l_returnflag = 'R' AND c_nationkey = n_nationkey;

There experiments were running on a virtual machine with Ubuntu 16.04. This machine has 4GB of memory and 125GB of disk.

We used a three scale factors of the TPCH- benchmark: 1, 10 and 50 for generating our test data. This results in a database that is approximately 1, 10 and 50GB respectively. The queries in the XDB were defined with the same execution time as returned in the VerdictDB, in order to compare the accuracy of the results in the same amount of time. Each query was executed five times in PostgreSQL, MySQL, VerdictDB and XDB. For each one of them have been the result of each query and the time each took to return a result. Then, with these values were calculated the speedup and the error rate.

C. *Discussion of results*

MySQL was the database engine chosen to assist the VerdictDB in AQP, because the data were stored in MySQL. The speedup ratio is a number that measures the relative performance gain of two systems processing the same problem. In our case it is the improvement in speed of query response time on VerdictDB compared to MySQL and is calculated as follows:

$$\text{Speedup} = \frac{\text{MySQL query response time}}{\text{VerdictDB query response time}} \quad (1)$$

Table II presents the speedup of the VerdictDB compared to MySQL. The VerdictDB was nearly 76x faster than MySQL in executing the query Q7 with 50 GB of data. This result is related to the sampling of the VerdictDB. The query Q7 is the query that joins more tables, and all are sampled. So, after the query rewriting with these tables sampled, the response is returned in a short period of time.

TABLE II
VERDICTDB SPEEDUP COMPARED TO MYSQL.

Query number	Size of TPC-H data		
	1GB	10GB	50GB
Q3	1.44	16.42	17.87
Q7	2.84	43.29	76.13
Q10	2.60	22.85	29.08

According to the results presented in Figure 1, you can see that the speedup increases with the increase of the quantity of data. The speedup also increases when the query has several larger joins, as we can see through the queries Q7 and Q10. This is because MySQL uses a simple multi-join method that looks for rows that match a given column. As such, your disk I/O cost is higher than in VerdictDB, which only uses a sample of the data.

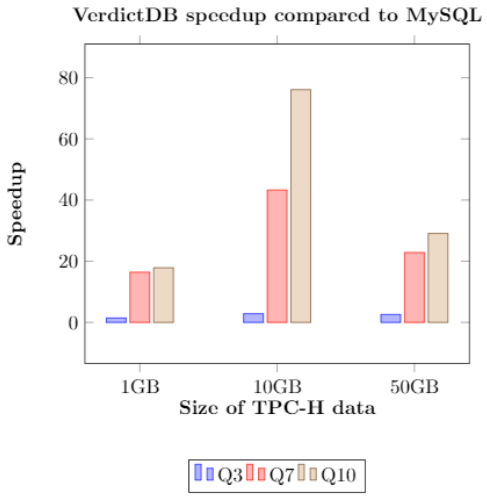

Fig. 1 Speedup in VerdictDB compared to MySQL.

Digital Object Identifier: (to be inserted by LACCEI).
ISSN, ISBN: (to be inserted by LACCEI).

The response returned by the VerdictDB was used in the parameter "WITHTIME" of XDB, being also used a confidence interval of 95%. Figure 2 shows the error rate of both tools for Q3, with the different sizes of TPC-H data. According to this figure, the VerdictDB presents higher error rates than the XDB for large volumes of data. The query Q3 is very simple and only involves the join between three tables, but the difference in results for 1 GB of data can be related to memory issues at the time of execution of the query.

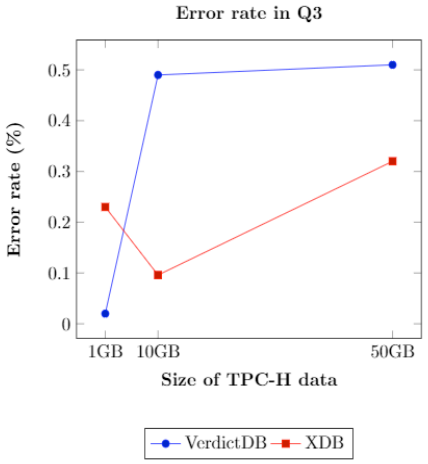

Fig. 2 Error rate in Q3.

Figure 3 shows the error rate of both tools for Q7, with the different sizes of TPC-H data. This is the query more complex, involving six joins. The error rate greatly increases compared to the query Q3, because information is required a larger number of tables. The high error rate of VerdictDB in relation to the XDB can be may be related to the fact that VerdictDB has a middleware architecture, all of which calculations about the sampling data are based on SQL.

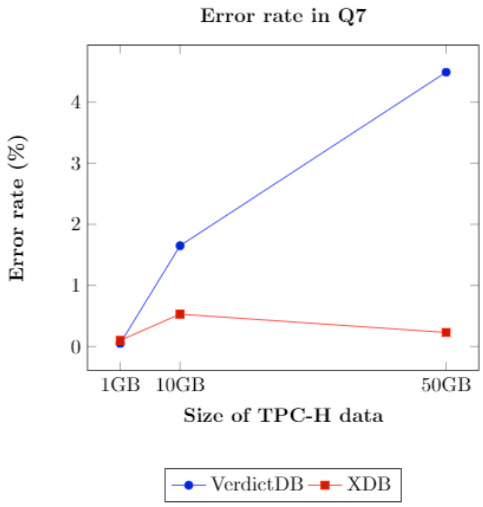

Fig. 3 Error rate in Q7.

Figure 4 shows the error rate of both tools for Q10, with the different sizes of TPC-H data. The error rate is still very low in both tools, because there is a very small amount of data to be sampled and are used a few tables. With 50 GB of data error rate is very low in XDB, since this only makes the sampling of a table with selection predicates. VerdictDB uses variational subsampling, in which case only one table is sampled, where the rows of the table belong to several subsamples. This causes a repetition of the data in the samples, increasing the error of the response to the query. Because XDB has a walking plan optimizer, it is possible that the best path has always been selected, since error rates are greatly reduced.

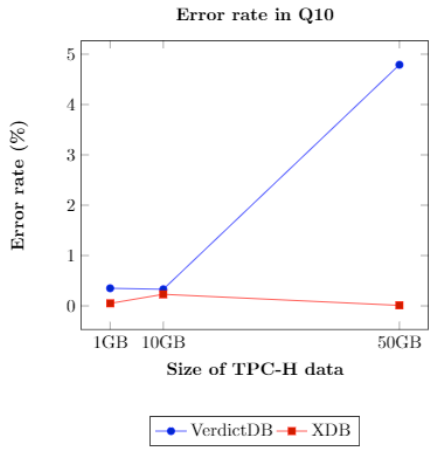

Fig. 4 Error rate in Q10.

The query Q3 is the one that presents a lower rate of error, because it uses fewer sample tables. On the other hand, the queries Q7 and Q10 use more sample tables, returning results less accurate.

The XDB give more accurate results than VerdictDB, especially when using many tables in the same query. The variational sampling of VerdictDB may be at the origin of these results, since this is based on randomness. Thus, it is not possible to be sure that a row in a table is not sampled again. If repeated lines are sampled, the sample does not become representative of the population and causes high error rates.

IV. CONCLUSIONS AND FUTURE WORK

In this paper, we compare the VerdictDB and the XDB with different scale factors of the TPC-H benchmark in relation to speedup and error rate. Also compared the speedup of VerdictDB compared to MySQL.

We conclude that the VerdictDB can be 76x even faster than MySQL. The VerdictDB and the XDB are good tools for large volumes of data, but the XDB stood out. With the same response time, the XDB returns results with double accuracy.

For small amounts of data, such as 1 GB, it is not justified to use an AQP system. Although we have a speedup of around 20 x greater in AQP, we also have an associated error rate. For this case, if we want a result less fast, but more precise, we use the queries in a traditional DBMS.

As future work, we propose to evaluate more AQP systems in Big Data and cloud environments. We also intend to suggest new algorithms to improve performance and accuracy.

REFERENCES

[1] Mozafari, B. 2017. Approximate Query Engines pproximate Query Engines: Commercial Challenges and Research Opportunities. In Proceedings of the ACM International Conference on Management of Data (SIGMOD '17). ACM, New York, NY, USA, 521-524.

[2] Han, X., Wang, B., Li, J. and Gao, H. 2017. Efficiently processing deterministic approximate aggregation query on massive data. Knowledge and Information Systems, 57(2), pp.437-473.

[3] Moritz, D. and Fisher, D. 2017. What Users Don't Expect about Exploratory Data Analysis on Approximate Query Processing Systems. Proceedings of the 2nd Workshop on Human-In-the-Loop Data Analytics - HILDA'17.

[4] Sassi, M., Tlili, O. and Ounelli, H. 2012. Approximate Query Processing for Database Flexible Querying with Aggregates. Transactions on Large-Scale Data- and Knowledge-Centered Systems V, pp.1-27.

[5] Moritz, D.. and Fisher, D. 2017. What Users Don't Expect about Exploratory Data Analysis on Approximate Query Processing Systems. Proceedings of the 2nd Workshop on Human-In-the-Loop Data Analytics - HILDA'17.

[6] Li, K. and Li, G. 2018. Approximate Query Processing: What is New and Where to Go?. Data Science and Engineering, (2018) 3: 379. https://doi.org/10.1007/s41019-018-0074-4.

[7] Inoue, T., Krishna, A. and P. Gopalan, R. 2016. Approximate Query Processing on High Dimensionality Database Tables Using Multidimensional Cluster Sampling View. Journal of Software, 11(1), pp. 80-93.

[8] Kraska, T. 2017. Approximate Query Processing for Interactive Data Science. Proceedings of the 2017 ACM International Conference on Management of Data - SIGMOD '17.

[9] Özsu, M. and Liu, L. Editors. 2009. Encyclopedia of Database Systems. New York: Springer.

[10] Vrbsky, S. 1996. A data model for approximate query processing of real-time databases. Data & Knowledge Engineering, 21(1), pp.79-102.

[11] R. Almeida, J. Vieira, M. Vieira, H. Madeira, and J. Bernardino. 2008. Efficient Data Distribution for DWS. In International Conference on Data Warehousing and Knowledge Discovery, DaWaK 2008, pages 75–86.

[12] J. Bernardino, P. Furtado, and H. Madeira. 2002. DWS-AQA: A Cost Effective Approach for Very Large Data Warehouses. In Proceedings International Database Engineering and Applications Symposium, IDEAS 2002, M. A. Nascimento, M. T. Ozsu, and O. R. Zaiane, editors, pp. 233–242. IEEE Computer Society.

[13] Potti, N. and Patel, J. 2015. DAQ: A New Paradigm for Approximate Query Processing. Proceedings of the VLDB Endowment, 8(9), pp.898-909.

[14] Haas, P. and Hellerstein, J. 1999. Ripple joins for online aggregation. Proceedings of the 1999 ACM SIGMOD international conference on Management of data - SIGMOD '99.

[15] Li, F., Wu, B., Yi, K. and Zhao, Z. 2016. Wander Join. Proceedings of the 2016 International Conference on Management of Data - SIGMOD ’16.

[16] Li, F., Wu, B., Yi, K. and Zhao, Z. 2017. Wander Join and XDB: Online Aggregation via Random Walks. ACM SIGMOD Record, May 2017, 46(1), pp.33-40.

[17] Agarwal, S., Mozafari, B., Panda, A., Milner, H., Madden, S. and Stoica, I. 2013. BlinkDB. Proceedings of the 8th ACM European Conference on Computer Systems - EuroSys '13.

[18] Park, Y., Mozafari, B., Sorenson, J. and Wang, J. 2018. VerdictDB. Proceedings of the 2018 International Conference on Management of Data - SIGMOD '18.

[19] Goiri, I., Bianchini, R., Nagarakatte, S. and Nguyen, T. 2015. ApproxHadoop. Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS '15.

[20] Chakrabarti, K., Garofalakis, M., Rastogi, R. et al.: Approximate query processing using wavelets. The VLDB Journal 10(2-3), 199-223 (2001)

[21] github.com/mozafari/verdictdb/

[22] github.com/InitialDLab/XDB

[23] docs.verdictdb.org/

[24] github.com/InitialDLab/XDB/tree/master/queries

[25] www.tpc.org/tpch/