Future Computing and Informatics Journal

Volume 7
Issue 2 (2022) *Volume 7 Issue 2*

Article 2

2022

Enhancing Query Processing on Stock Market Cloud-based Database

Hagger Essam
Helwan Univesity, hagger_bahgat@cic-cairo.com

Ahmed G. Elish
Helwan University, ahmed.g.elish@commerce.helwan.edu.eg

Essam M. shaban
Beni-Suef University, essam.shaban@fcis.bsu.edu.eg

Future Computing and Informatics Journal

Homepage: https://digitalcommons.aaru.edu.jo/fcij/

doi: http://Doi.org/10.54623/fue.fcij.7.2.2

ENHANCING QUERY PROCESSING ON STOCK MARKET CLOUD-BASED DATABASE

Hagger Essam Mohamed [1, a], **Ahmed Gamal Elish**[2, b], **Essam M. Shaaban** [3, c]

[1] Business Information Systems, Faculty of Commerce and Business Administration, Helwan University.
[2] Economics and foreign trade department, Faculty of commerce and business administration, Helwan University.
[3] Information system department, Faculty of Computers and artificial Intelligence, Beni-Suef University.

[a] hagger_essam@cic-cairo.com, [b] ahmed.g.elish@commerce.helwan.edu.eg ,
[3,c] essam.shaban@fcis.bsu.edu.eg.

ABSTRACT

Cloud computing is rapidly expanding because it allows users to save the development and implementation time on their work. It also reduces the maintenance and operational costs of the used systems. Furthermore, it enables the elastic use of any resource rather than estimating workload, which may be inaccurate, as database systems can benefit from such a trend. In this paper, we propose an algorithm that allocates the materialized view over cloud-based replica sets to enhance the database system's performance in stock market using a Peer-to-Peer architecture. The results show that the proposed model (MVCRS) improves the query processing time and network transfer cost by distributing the materialized views over cloud-based replica sets. Also, it has a significant effect on decision-making and achieving economic returns.

Keywords: Query Processing, Cloud Computing, Replica Sets, Materialized Views.

1. INTRODUCTION

In the stock market, stock value movements depend on several parameters and a lot of economic and non-economic variables, which affect the prices and volumes of trading, as a result, it is critical to optimize these parameters based on historical data results. In such a vast search space for parameters as the huge volume of historical data, it is an arduous task. Customer requests force databases to make quick decisions based on up-to-date information. Dealing with a large number of data can delay the performance, particularly in data retrieval, affecting the user experience. Also, the queries become very complex. The database is required to answer to these queries with high-quality data in a short period of time, but the key problem is the size of the data and the type of the queries. In a cloud-based database, users in Egypt can access and request data from several sources and distributed places, which can take loads of time to reach the data, that will delay their decisions, which may lead to a massive loss. The Cloud Computing model will play a huge role, which consists of processing data remotely in data centers and having end-users' devices act as terminals for inputting and displaying information. Data is retrieved on-demand and updated continuously. [1-5].

2. RELATED WORK

C. A. U. Hassan et al.,2022 proposed a cache mechanism that combines frequency, size, and aging policies. The cache-based technique supports Data warehouses in two ways: it reduces execution time through accessing records straight from the cache memory. Also, it conserves cache memory by eliminating infrequent data. Their purpose was to load the most frequently used data into the cache memory. To accomplish this purpose, the aging-based. The Least Frequently Used (LFU) algorithm was used, which considers data frequency and size. LFU assigns cache memory data a priority and analyzes its age. The cache block entry with the minimum age count and priority was erased first. Finally, the proposed cache mechanism made good use of cache memory and linked the performance gap between the primary data warehouse and the business user query. Their results indicate that

when the size of the requested data increases, the execution time begins to decrease by taking data size and frequency into consideration during performance assessments. They asserted that their technique outperforms the Size and LFU techniques separated [1].

Several more, H. Azgomi and M. K. Sohrabi, 2021 introduced the Map-Reduce-based MVPP (MR-MVPP) which executes a group of similarity join based on views and relations utilizing the hashing approach in conjunction with the map-reduce model. By avoiding redundant computations throughout the MVPP formation process, the proposed approach decreases MVPP development time. The MR-MVPP strategy, according to the study findings, has a quicker execution time than the other approaches. The average increase in time was 26.5 units approximately. Regarding the MVPP's efficacy, the proposed strategy works well and has a roughly 50% coverage percentage for view selection techniques. Predetermined approaches are more precise than hashing approaches and can be used for set similarity join in future studies to significantly reduce OLAP system query response time [6].

A framework is presented by Refed A. Jaleel, Talib M. J. Abbas, 2020 for choosing the optimum materialized view using the Quantum Particle Swarm Optimization (QPSO) technique. The findings show that this method outperforms others by calculating the ratio of query response time and comparing it to the response time of the identical queries on Materialized Views (MVs). The query execution on the base table takes five times more than the query execution on the MVs. Whereas the MVs access queries take 0.084

seconds to respond, direct access queries take 0.422 seconds, which shows that query performance using MVs access is 402.38 percent better than query performance via data warehouse-logical access. [8]

A. Sharma and P. Kaur, 2019 advocated utilizing a column-based NoSQL database to construct a multi-tenant data storage. Cassandra was used to implement the planned multi-tenant datastore. The built materialized views are then utilized to execute queries for distinct tenants in the Cassandra data storage engine, providing each tenant a feeling of isolation. In Cassandra the command TRACING ON/OFF is used for timestamp calculation. The findings show that the proposed architecture provides tenants with the necessary data isolation and performs well for large datasets, however query processing time increases for small datasets. [9]

While D.T. Wojtowicz et al.,2021 introduced Nebula, a non-profit middleware that offers multi-cloud querying capabilities through outsourcing its users' queries to the DBaaS providers. First, they offered a quotation mechanism for those questions whose necessity arises from the providers' pay-per-query policy. These quotations included estimated costs and response times and are created by provider-provided bids. They then offered an agent-based dynamic optimization engine that coordinates the outsourced execution of the queries. Agents inside this engine worked together to meet the specified values. Using the Join Order Benchmark, they compared Nebula to simulated providers (JOB). Their findings revealed that Nebula could calculate the cost of the multi-cloud searches it orchestrates. The dynamic optimization technique outperforms the replication and execution model from the multi-cloud DBMS literature concerning performance and cost, especially when the database is frequently updated.[10]

C. Wang et al., 2021 proposed a multi-query processing strategy for an end-to-end cloud database system based on reinforcement learning (RL). This algorithm has several stages. Initially, a query is provided to the optimizer, which at that time is compiled into a query execution plan (QEP). Then the first accessible operator is given to the RL model, which is converted into a vector representation. Then the RL model selected the best action, which is a mix of a physical operator and a container to execute this operator. After that, the reward function is updated by executing this operator and documenting the execution time and financial costs.

The weights of this RL model are modified to match the new reward function after the reward function is altered. Finally, the modified RL model is ready for further operator action. The proposed model, Re-optimization with reinforcement learning (ReOptRL), improved the time of query response from 12% to 39% and monetary cost from 17% to 56% compared to current algorithms that use no re-optimization, re-optimization after each stage of the QEP, supervised machine learning-based query re-optimization, or sample-based re-optimization. [11]

According to S. A. Chakraborty, 2021 when a query is issued, the MQDB is scanned to find any existed query in the database. IF the query existed and the saved results didn't require any updates, the results are obtained directly from the database. The mechanism of holding results of the requested query, then producing them once a matching query is performed reduced the time of query processing. The new assessment technique, which uses the data warehouse on the mid as well as a remote cloud server, demonstrated a reduction in the time required to get the results of the queries when compared to the conventional approaches. Queries, results, and meta-data are kept in MQDB. The study indicated that: When compared to alternative techniques such as data warehouses and data cubes, MQDB significantly reduces the time of similar queries. As the data warehouse is hosted on the central server, the time reduction of similar queries without updated results using MQDB is by nearly 95% compared to data warehouses usage and 84% compared to cubes usage. Once the data warehouse is hosted on a cloud server, it requires an additional 3.6272 seconds on average to connect to the distant cloud instance. The connection time varied based on the provider, bandwidth, time lag, and location of the cloud instance. Whereas processing identical queries without incremental updates, a substantial processing

time reduction is attained by nearly 99% and 98% compared to data warehouse and data cubes usage. [12]

U. Tos et al. 2021 introduced APER, a dynamic data replication approach to meet the objective of response time while also providing a financial benefit to the provider in the cloud. Their technique anticipated if the response time objective was met by estimating the response time of database queries prior to execution. If it was determined that a query would break the SLA, the recommended technique explored the deployment of a new replica to continue the execution. Also, a dynamic modification in the replicas number was performed in order to cloud resource consumption minimization. They assessed the cost of running each query based on predicted resource usage, whereas expected income is computed from rent received from tenants. They studied the effectiveness of APER in conjunction with other techniques. APER delivered on performance and profit by deploying replicas to improve data access time and minimize resource usage. [13]

3. THE PROPOSED MODEL

In this section, the proposed work is partitioned as follows. Initially, the pre-processing phase, where the data collection, and data sources are presented, also prepares the data for proper query processing format. Then, different cloud allocation techniques for enhancement and recommendations are introduced.

This section aims to explain the used framework, to improve stock markets' trade flow. And clarify the approach and steps taken to achieve the research results.

The following figure shows the model we used in query processing procedure, going through data collection to reaching the evaluation phase (results from different techniques: cloud database allocation, cloud materialized views allocation, cloud database allocation using replica sets, and cloud-based materialized views allocation using replica sets were illustrated in this phase) using seven stages.

Stage 1: Determine the needs of our clients. (Output: data types and attributes-depends on the client needs).

Stage 2: Data collection & preparation (Output: CSV files containing stock market's transactions).

Stage 3: Query request via Swagger (Open API documentation) (Output: query).

Stage 4: Materialized view existence check (Output: Yes-No).

Stage 5: In case of MV existence (Output: query results from the materialized view located in the replica sets), In case of MV non- existence (Output: query results from the cloud-based database).

Stage 6: Results (Output: Evaluation and analysis report)

Stage 7: Feedback and Enhancements.

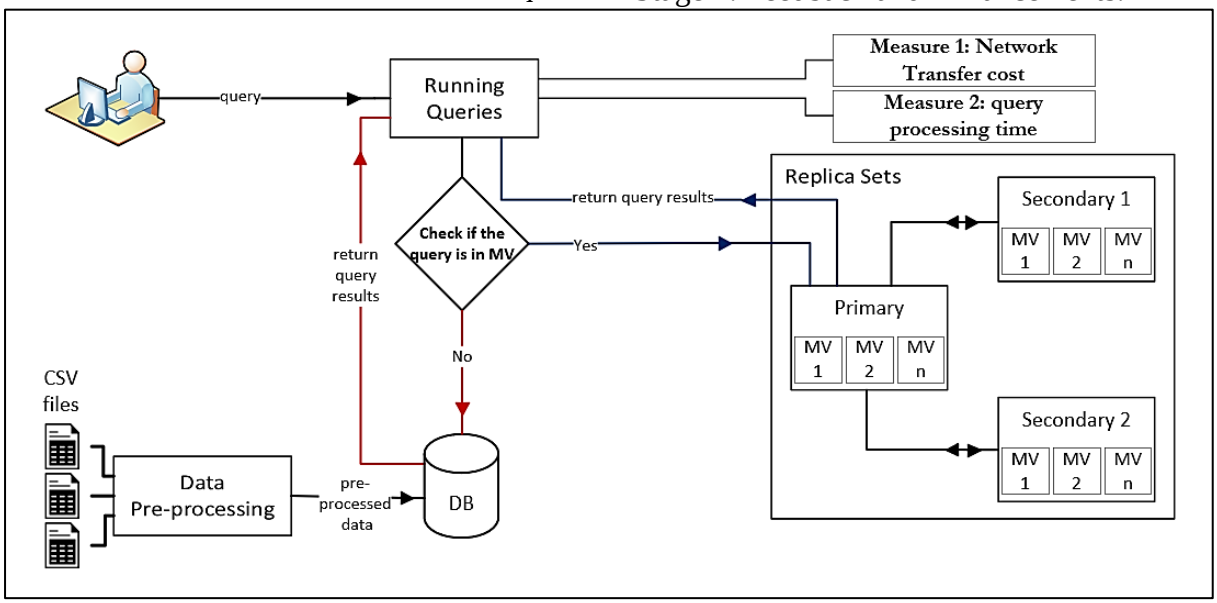

Figure 1: The proposed Model (MVCRS).

The model is composed of the following phases:

A. Data Pre-Processing

Data pre-processing is a vital phase in this model that consists of many steps. Extraction is the first step as it extracts the data from the stock market system. Once the data is collected from the several sources, the information is stored in a singular database. Cleaning data errors is required to use this data for a single integration point. Moreover, the data is directed to the transformation step for data cleansing, where data is converted into same format. There are various transformations, including data cleansing, handling missing elements, parsing into typical formats, and reduplicating data. The final step is storing data in the database. It is essential to ensure that the load is performed accurately.

- **Used Dataset**

In MVCRS an Egyptian stock market's system is used for collecting the data. The dataset holds 239 csv files, each filename indicates the company name, these files contain 220,191.00 instances and seven attributes of historical daily prices for some tickers from April 2017 to April 2022. Data cleansing is performed in this model using a PHP script that searches for the null/empty values and removes them.

The data attributes are described in table 1:

Table 1: Data Description:

Item	Description
Symbolid	unique ID for each company.
Ts_date	Specifies transaction's date.
Open	Opening price.
Close	Closing price.
High	Maximum price during the day.
Low	Minimum price during the day.
Volume	no. of changed shares in a day.

B. Cloud-based Servers

Google Cloud is used for building two Ubuntu servers, the first server "web" contains the PHP scripts that are used to connect to the second server "mongodb" which holds the data. Vesta control panel (VestaCP) is a free, open-source, with easy installation and configuration web-based control panel, which can be used to manage several websites, create and manage FTP accounts, email accounts, and databases.
Vesta Control panel is used in this model to ease the file uploading process. Multiple Services are installed on VestaCP such as PHP version.7.3, Apache Server, and NGINX along with deploying Laravel.

C. Database Cloud Allocation

Studio 3T supports MongoDB data management via storing new data, viewing, querying, and calculating current data. It also makes it simple to connect as many MongoDB servers as needed. In MVCRS, Studio 3T is used to create the database "market." Connect it to the "mongodb" server next.

D. Database Allocation using Replica sets

A replica set is the replication of a collection of MongoDB servers that hold copies of identical data; this is a crucial feature as it guarantees high redundancy and availability, which are vital characteristics to have in case of failovers and arranged maintenance periods. The main purpose of a sharded cluster, where data is distributed across many servers is to scale reads and writes along multiple shards. In this model as the standard replica set deployment for a production system which is a three-member replica set (P-S-S) one primary and two secondaries, to hold the same copies of data via MongoDB cluster.

E. Materialized Views Allocation

View materialization is the procedure of computing and storing the result of a query in the database in order to decrease processing cost. It is not conceivable to materialize all the views because it takes loads of storage. As a result, we must choose certain views to be materialized, those with the highest frequency of requests. In this model materialized views are allocated once

in cloud node using studio 3T like the database and allocated once more on the replica sets to measure the query processing enhancement, according to the following criteria:

i. Select the most frequent tables to be materialized
ii. Launch Studio 3T for MongoDB
iii. Choose the desired database
iv. Add view
v. Choose the desired collection (tables) to create a view on.

F. Query Processing

The following section illustrates query processing:

i. Query issuing is done via Swagger API.
ii. The materialized views are checked for query existence.
iii. If the query exists in the MVs the results are retrieved directly from them.
iv. In case it doesn't exist, the query is passed to the database itself to get the results.
v. Current time (microtime) is saved once before the query processing.
vi. Current time (microtime) is saved again after the query processing.
vii. Subtraction of the time before query processing from the time after query processing which will give us the actual query processing time.
viii. Regards the query answering the two variables that hold the query result and query processing time are returned to the API that prints them.
ix. The network transfer cost is calculated using the Swagger API.

4. Results and Discussions

MVCRS has many approaches the first applies the database over a cloud node and perform queries on it. The second allocated the materialized views over a cloud node and get the results directly from them. Third approach allocated the database over a cloud-based replica sets and perform queries on these copies. The last approach uses replica sets to store materialized views and get the results straight from them. The following experiments were operated on 50 different queries based on the stock market needs, shows the variance between the processing time and Network transfer cost for each applied algorithm. Each approach has a purpose (query processing time and network transfer costs enhancement)

Experiment 1 measures network transfer cost. Figure 2 shows a comparison between the cloud database allocation, cloud materialized views allocation, cloud database allocation using replica sets, and cloud materialized views allocation using replica sets against the transfer cost. The experiment result shows that cloud view allocation using the replica sets algorithm reduces the network transfer cost by 22.08%.

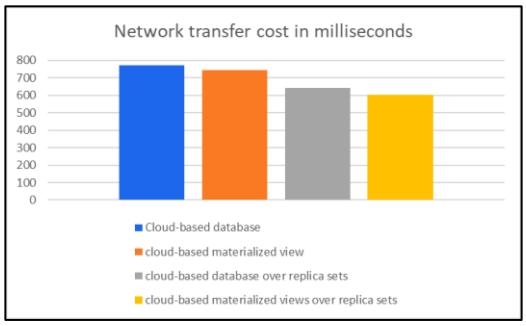

Figure 2: Comparison between the four approaches against the network transfer cost.

Figure 2 shows the time taken for the network transfer cost for Cloud-based database, cloud-based materialized view, cloud-based database over replica sets, cloud-based materialized views over replica was 771.78, 745.40, 643.25, and 601.36 respectively.

Experiment 2 measures query processing time. Figure 3 shows a comparison between the cloud database allocation, cloud materialized views allocation, cloud database allocation using replica sets, and cloud-based materialized views allocation using replica sets algorithms against the processing time. The result shows that cloud view allocation using the replica sets algorithm reduces the query processing time by 24.63%.

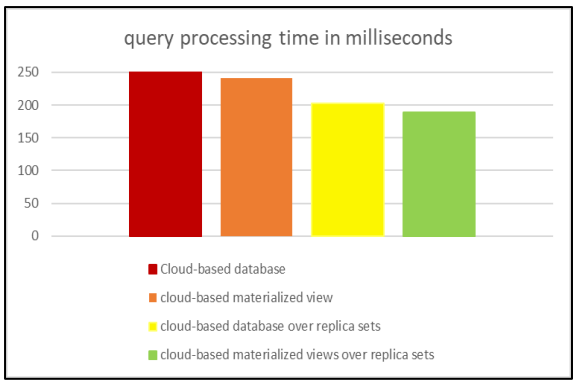

Figure 3: Comparison between the four approaches against the query processing time.

Figure 3 shows the time taken for the processing time for Cloud-based database, cloud-based materialized view, cloud-based database over replica sets, cloud-based materialized views over replica was 250.53, 241.24, 203.01, and 188.82 respectively.

The Yahoo Cloud Serving Benchmark (YCSB) is a set of database benchmarks. It enables the evaluation of the performance of a wide range of current NoSQL and SQL database management systems using basic database operations on synthetically created data. YCSB may be used to compare the performance of various database setups under different workloads. Using YCSB in MVCRS shows that the throughput for the runs of 200,000 records with varied number of threads such as 2, 4, 8, 16, and 32 was 968.8653, 3195.6028, 4220.4777, 5107.2522, and 36608581 (ops/sec) respectively.

According to stages and steps we followed above, the results showed that MVCRS enhanced query processing by allocating the materialized views over a cloud-based replica set. These enhancements will lead to avoid late decisions that can waste lots of profits, which can be presented in both macro and micro economic returns.

5. CONCLUSION AND FUTUREWORK

MVCRS is developed and tested by using peer-to-peer architecture in applying query processing using different techniques, including cloud database allocation, cloud materialized views allocation, cloud database allocation using replica sets, and cloud materialized views allocation using replica sets.

Our approach to database system performance is based on assessing query processing time as well as network transfer cost between peers. MVCRS improved them by almost 25% and 22%, respectively.

Future work aims to update the materialized view dynamically with the new queries issued into the database.

REFERENCES

[1] C. A. U. Hassan, M. Hammad, M. Uddin, J. Iqbal, J. Sahi, S. Hussain, and S. S. Ullah, “Optimizing the Performance of Data Warehouse by Query Cache Mechanism,” IEEE Access, vol. 10, pp. 13472–13480, 2022.

[2] Moktadir and N. M. Istiak Chowdhury, “Subject Oriented Data Partitioning – A Proposed Data Warehousing Schema,” 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), May 2019.

[3] Garani, A. Chernov, I. Savvas, and M. Butakova, “A Data Warehouse Approach for Business Intelligence,” 2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), Jun. 2019.

[4] O. Pivert, Ed., “NoSQL Data Models,” Aug. 2018.

[5] K. Sudhakar and S. Naganjaneyulu, “Optimizing parameters in algorithm trading using map reduce on Indian stock exchange (Sensex),” Journal of Statistics and Management Systems, vol. 23, no. 2, pp. 389–397, Feb. 2020.

[6] Azgomi and M. K. Sohrabi, “MR-MVPP: A map-reduce-based approach for creating MVPP in data warehouses for big data applications,” Information Sciences, vol. 570, pp. 200–224, Sep. 2021.

[7] Gupta and A. Sahayadhas, “Proposed Techniques to Optimize the DW and ETL Query for Enhancing data

warehouse efficiency," 2020 5th International Conference on Computing, Communication and Security (ICCCS), Oct. 2020.

[8] R. Adnan and T. M. J. Abbas, "MATERIALIZED VIEWS QUANTUM OPTIMIZED PICKING for INDEPENDENT DATAMARTS QUALITY," Iraqi Journal of Information & Communications Technology, vol. 3, no. 1, pp. 26–39, Apr. 2020.

[9] A. Sharma and P. Kaur, "A Multitenant Data Store Using a Column Based NoSQL Database," 2019 Twelfth International Conference on Contemporary Computing (IC3), Aug. 2019.

[10] D. T. Wojtowicz, S. Yin, F. Morvan, and A. Hameurlain, "Cost-Effective Dynamic Optimisation for Multi-Cloud Queries," 2021 IEEE 14th International Conference on Cloud Computing (CLOUD), Sep. 2021.

[11] C. Wang, L. Gruenwald, L. d' Orazio, and E. Leal, "Cloud Query Processing with Reinforcement Learning-Based Multi-objective Re-optimization," Lecture Notes in Computer Science, pp. 141–155, 2021.

[12] S. A. Chakraborty, "A Novel Approach Using Non-Synonymous Materialized Queries for Data Warehousing," International Journal of Data Warehousing and Mining, vol. 17, no. 3, pp. 22–43, Jul. 2021.

[13] U. Tos, R. Mokadem, A. Hameurlain, and T. Ayav, "Achieving query performance in the cloud via a cost-effective data replication strategy," Soft Computing, vol. 25, no. 7, pp. 5437–5454, Jan. 2021.