Tampere University

Aki Kankaanpää

QUERY PERFORMANCE AND OPTIMIZATION IN CLOUD DATABASES

Informaatioteknologian ja viestinnän tiedekunta
Kandidaatintutkielma
Lokakuu 2022

TIIVISTELMÄ

Aki Kankaanpää: Tietokantakyselyiden suorituskyky ja optimointi pilvitietokannoissa
Kandidaattitutkielma
Tampereen yliopisto
Tietotekniikka
Elokuu 2022

Tietokannat ovat olleet tarpeellinen osa lähes jokaista toimialaa jo vuosikymmeniä. Ne helpottavat monenkaltaisen tiedon ylläpitämistä, muokkaamista sekä käsittelyä yksityisellä, kaupallisella sekä julkisella sektorilla. Maailman digitalisoituessa vuosi vuodelta yhä useampi yritys tarvitsee tietokantapalveluita, muttei välttämättä omista tarvittavia resursseja ja pääomaa tarvittavan elektronisen infrastruktuurin rakennuttamiseen. Tämän tarpeen täyttääkseen jotkin yritykset ovat aloittaneet tarjoamaan tietokantapalveluita pilvipalveluiden kautta ja välttäneet näin tarpeen paikallisesti asennetuille palvelimille.

Tietokannan tarjoama tallennustila ei itsessään riitä kaikkiin tietojenkäsittelyn sovelluksiin, vaan tietokantaan suoritettujen kyselyiden täytyy myös toimia tarpeen mukaan tehokkaasti ja nopeasti. Edistysaskeleet datan analysointiin ja keräämiseen ovat myös mahdollistaneet massadatan hyödyntämisen esimerkiksi kaupallisella sektorilla personalisoidun mainonnan kautta. Jotta näitä prosessointialgoritmejä voitaisiin käyttää suurelle määrälle dataa, ovat tietokannan vaatimukset sekä suorituskyvyn että optimoinnin suhteen kasvaneet.

Yksi tärkeimmistä tavoista parantaa kyselyiden tehokkuutta on käyttää optimointialgoritmeja. Tietokantojen toteutuksien eroista pilviympäristön ja paikallisesti asennetun kannan välillä, optimointialgoritmeja on muutettava toteutustavan mukaan, joka voi johtaa suorituskykyeroihin.

Tämä tutkielma on muodoltaan kirjallisuuskatsaus, joka sisältää selityksen tietokantojen ja niihin toteutettavien kyselyiden perustoiminnasta, sekä vertailun tietokantahakujen, niiden optimoinnin ja suorituskyvyn eroista perinteisien ja pilvitietokantojen välillä.

Tutkielmassa saatiin selville pilvitietokantojen olevan nykyisin varteenotettava vaihtoehto perinteisien tietokantojen rinnalla. Samansuuruisella sijoituksella kumpaan tahansa vaihtoehtoista saadaan toisiaan lähellä olevia tuloksia. Pilvitietokantojen suorituskyky on parantunut selvästi vuosien varressa, ja aiheen ollessa vieläkin aktiivisen tutkimuksen alla, on täysin mahdollista, että ero perinteisien ja pilvitietokantojen välillä vähenee tai katoaa täysin tulevina vuosina.

Avainsanat: tietokanta, tietokantakysely, optimointi, pilviympäristö

ABSTRACT

Aki Kankaanpää: Query performance and optimization in cloud databases
Bachelor's Thesis
Tampere University
Computing Sciences
August 2022

Databases have been a necessary part of nearly every field for decades, helping the modification, upkeep, and processing of data in both the private, commercial, and public sectors. As the world become more digitalized by the year, the number of companies that require database services that do not have the requisite resources and capital to build the necessary electronic infrastructure to support their needs. To fill this demand, some companies have taken to offering database services through cloud computing, avoiding the need for on-premises architecture for the clients.

For companies in fields or industries that heavily utilize data processing, not only does the database have to support large volume, but queries to the database also must be performed quickly and efficiently. Advancements in data analysis and collection have also enabled the usage of big data in ways that wouldn't have been possible, such as through personalized advertising through consumer patterns. For a database to be able to support processes at that scale, an even greater degree of performance and optimization is required from the database.

One of the major ways to improve query performance is to use optimization algorithms and methods. Due to the differences in the way a cloud database is structured when compared to a traditional, on-premises database, the optimization algorithms must be adapted to work in the cloud environment, which presents new problems and possible differences in performance.

This thesis contains a leading explanation into databases, and an evaluation on the differences of database queries between on-premises and cloud databases, as well as differences to their query optimization and performance.

In the thesis we discover cloud databases to be a formidable alternative to their on-premises counterparts. A similar investment into either option will currently give you similar performance. In addition, the performance of cloud databases is seen to have clearly improved over the years, and as the subject is still under research, it is completely feasible to see cloud databases to lessen or remove the gap between their on-premises counterpart in the coming years.

Keywords: database, database query, optimization, cloud environment

CONTENTS

1. INTRODUCTION

“From tiny devices to megastructures, every entity in this world is generating data, whether it is living or nonliving.” [7] Data is everywhere. Advances in data processing have created more and more methods of processing that data, finding patterns that may be transformed into profit.

Some companies and firms have a need for database services but are not able to or would greatly suffer financially from enacting the infrastructure required to create a database on their own premises [15]. Whether that need is simply to upkeep a small collection of past transactions, or to contain and process large volumes of data, an alternative to on-premises is in demand.

To supply this demand, cloud service providers have taken to offering database functionality through a cloud platform. These services, while not free, present an opportunity for companies looking for a solution without having to have the capital for the initial investment, often providing plenty of performance for a lower price point.

As the competition within the industry has grown, one of the main metrics for companies aiming to provide services for larger clients has become the performance of the database. An increasing number of companies use big data computations to determine their business decisions [4], which require vast resources to compute.

While performance may be acquired using higher-grade hardware, optimization is usually focused on improving the method a query is executed. Cloud computing has complicated the process of query optimization [3], as the way in which queries are conducted differs from traditional, on-premises databases. In cloud environments, query optimization is considered as one of the main challenges in the context of query processing [2].

The second chapter details the research method for the literature used within this thesis. The third chapter explains and differentiates on-premises and cloud databases. The fourth chapter details query optimization. The fifth chapter describes the results of empirical tests done on database performance, both comparing on-premises to cloud databases and optimization results on cloud databases. The sixth chapter contains an assessment of the results of the tests. The seventh chapter concludes the thesis with a summation of the findings as well as a prospective view into the future.

2. RESEARCH METHOD

The main portion of the literature selected for sources within this thesis were queried from the Andor search service. Other services, such as Google Scholar and IEEE Xplore, were queried to find a broader range of already existing research on the subject. As the technology is still under research and is comparatively newer, all of the sources for this thesis were published in 2010 or later, with a large quantity released within the last 5 years. This was done to get a recent view at the development of cloud databases, as well as to create a comparison between test results taken in the early and late 2010s.

The searches into the databases were performed using keywords such as query optimization, query processing, query performance, cloud database and cloud environment. For sources on the differences between cloud and on-premises databases, keywords such as traditional database or on-premises coupled with comparison, comparative or evaluation. Queries that had the form of “cloud database” AND X returned around 100-300 results, which was narrowed down by the already mentioned release date.

The selection process for which articles were chosen was done manually by reading the title and continuing onto the abstract if the title seemed promising, as the amount of literature was small enough to work through. Some of the sources were also found by browsing the sources listed on already chosen pieces of literature.

3. CLOUD DATABASES

A database is a structure collection of files and information that may be stored, updates and retrieved by a user or by a program [8]. Although a database can mean a physical collection of information or items, within the confines of this thesis it refers to an electronic database storing data within a computer's memory.

"A Cloud can be defined as a parallel and distributed system which has a number of virtualized and interconnected computers." [9] Simply combining the two definitions, a cloud database may be defined as a structure of files and information, stored across multiple connected computers as a parallel and distributed system, where a user or a program may store, update, and retrieve currently stored data. While this description does not fully grasp the complexity and potential differences from case to case, it as a generalization is not a bad definition.

3.1 Main features of cloud databases

In on-premises databases, the capacity of the database is limited by the hardware present and can often not be increased to fit the growing demands of the client. In cloud computing however, the data is stored in nodes, the amount of which may be scaled to fit the demands of the client while the service provider has nodes to allocate [3]. These nodes may therefore be dispersed to multiple storage sites, technically increasing the maximum capacity to the total capacity of all the sites combined.

Additionally, it is possible to create different levels of performance to fit different clients' needs, as not only is the storage space allocated in this manner, but the performance may also be divided into categories depending on how much of the possible processing power is given to the processes of a certain client, as the nodes contain not only the data, but also contain the computing power of the database. This gives great flexibility to the service provider to grant multiple different plans for the clients to choose from. With this elasticity, providers have the capability to also add and remove resources according to demand in real-time [3].

3.2 Query differences between on-premises and cloud databases

A query is a question for another entity, with the purpose to request information. In databases, this definition is still partially valid, as a database query can be used to retrieve information on the data contained within it, but it also has other possibilities, such as changing the data located inside the database. Queries are created using a query language, the most common of which is SQL (Structured Query Language), which allows the database to understand and transform a request into a set of instructions for itself with which it may perform the requested operations.

In an on-premises database, the query is received by the database, and the database itself both processes and performs the query, using models and functions it has been provided. Whenever a query is requested of a cloud database, it is sent to the master nodes, which decompose the query into multiple subqueries which may be performed concurrently and create a plan for how the query will be executed [5]. After the query has been decomposed, the master nodes dispatch the instructions for the subqueries to slave nodes for processing. When the slaves return the results, the master nodes merge the returned result, which is then returned to the requestee. As the subqueries are not guaranteed to require the same amount of time and resources, the division of work between the subqueries is a key part in the optimization of the query to balance the load each individual node receives [5].

4. THE OPTIMIZATION OF DATABASE QUERIES

The total cost of a standard query execution includes the total cost of the necessary central processing unit instructions and disk I/O operations. In cloud databases, one must also consider the communication and transference of data between the various nodes as additional cost on top of the already existing costs [2]. Therefore, to minimize the cost of a query, one should aim to reduce as many of these variables by as large of a portion as possible.

The two main areas where the cost of queries may be reduced are query processing and data partitioning. Query processing attempts to optimize the way the single query is performed by estimating which method of performing the query will have the best performance and / or the lowest cost. Data partitioning attempts to partition the data added onto the database to minimize unnecessary operations within the database cluster in future queries.

Additionally, in cloud databases, as resources and data are dynamically stored, optimization methods cannot be directly migrated onto cloud services, as they cannot predict the future availability of resources [2]. Some methods work well when made to fit into their new environment, while others are not efficient enough to be applied even after reconfiguration.

Queries range from simple to complex, both ends affected by the volume and the type of data which is used. While demanding and complex queries may have more room to optimize, the most common use case, which often are simpler, but mandatory operations, should most often be the focus on how optimization is put into place.

4.1 Query Processing

Query processing refers to the entire operation from when the database receives a query request from a user, all the way to when the database sends the results of the query back to the user.

The query parser is the first tool used in query processing. It is used to check the validity of the query in question to determine whether the query should be performed at all, and how the user should be informed if the query is not valid. The syntax of the query is verified to determine if the query is valid within the used database language. The semantics of the query are verified to determine if the tables, objects, or other entities the query attempts to affect or observe both exist and that the user has rights to perform the requested operations to them.

Additionally, both the main query itself as well as in more high-performing databases the subqueries may be given identifying values, often in the form of hash codes. As creating an optimization plan for a query can be an expensive operation depending on its complexity and the way the data is stored, the plans themselves are saved for future usage. If the same query is performed again, the plan may be re-enacted by the query parser in the case that the main storage structure has not changed [14], saving the time and resources from having to create the plan each time after the first. As data allocation is very dynamic in cloud databases, it is yet another challenge to design the allocation of data in such a way that optimization plans may be reused even after changes have been made.

The second tool used in query processing is the query optimizer. The job of the query optimizer is complex, as there are multiple different variables to consider. A cost model is a mathematic algorithm or equation which may be applied to a query to determine the cost it would accrue if the model was used. It uses functions which calculate the total cost of the various operations a specific method of performing a query would cause, which may and should into account the total volume of data, the algorithmic complexity of the query in question and the physical costs, such as I/O and CPU operations, the latter of which relies more heavily on the hardware in question [12]. The optimizer chooses one of the cost models and sends the instructions to the master node, which disperses it to the slave nodes.

Within a cloud environment, the cost model should also consider the possibility of parallelism across multiple different nodes, while still maintaining minimal communication between them by minimizing the number of distributed transactions [3].

Query optimization approaches may be divided into static and dynamic methods. Static methods rely on the values used for optimization staying similar during the execution of the query. Static methods are not always applicable to cloud systems, as one of their main features is to be changed dynamically [1]. However, while dynamic optimization techniques are more well suited for cloud systems, they are often more expensive than the static alternatives, therefore they must increase performance in general, as well as outperform the less expensive static methods.

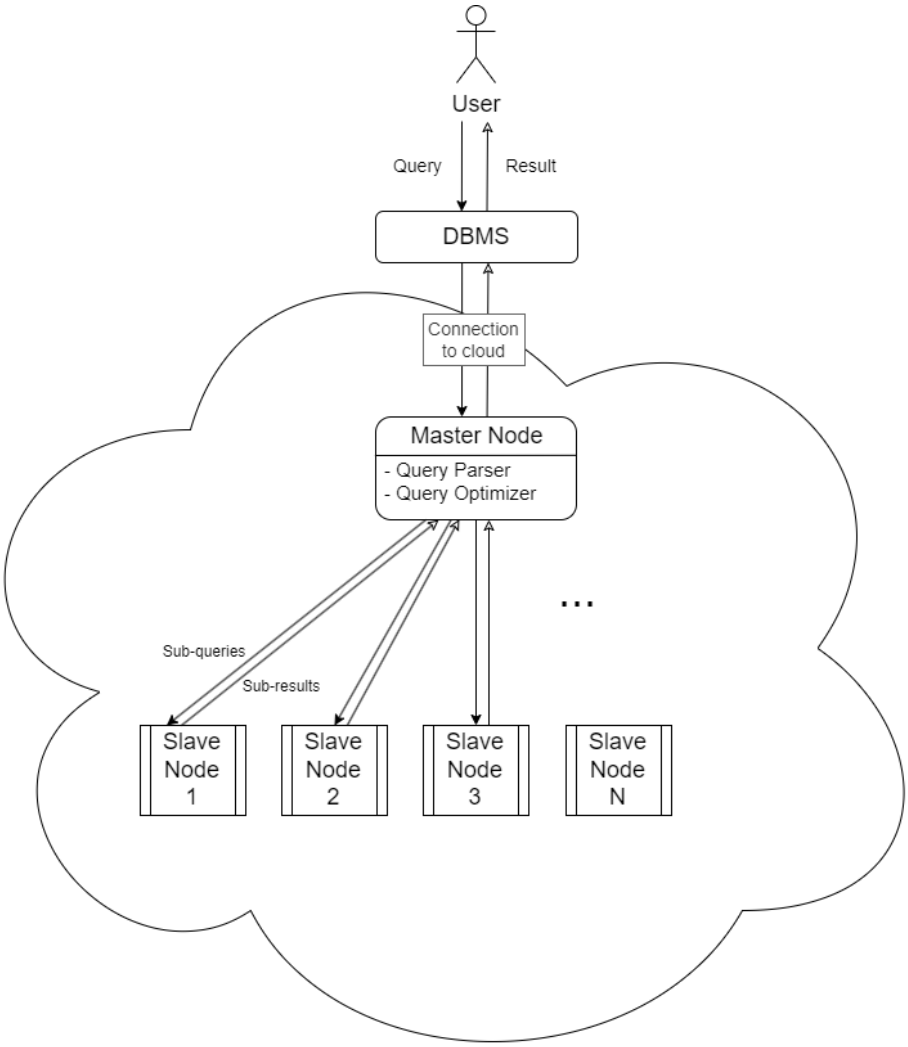

Image 1. *Query processing in the cloud, adapted from [3].*

As can be seen in image 1, the user is never in contact with any of the slave nodes but contacts the cloud through an interface such as a DBMS, directing their requests to master nodes. However, in larger clusters, the cloud may have more than one master node which either oversee different areas of the database, or act together to increase the total computing power the master nodes have.

4.2 Data Partitioning

Data partitioning is a method of distributing data across multiple storage sites and nodes [12], either done as data is being input into the database, or afterwards through a partition process. As nodes can and need only to participate in database operations if they contain the data they need to process, the partitioning of data requires observation, and from an optimization side more importantly, prediction on how to allocate resources and already existing data [3].

While the data may be partitioned in a multitude of different methods and configurations, the methods usually work towards a few different outcomes. Some partitioning methods attempt to split the data into chunks that may be identifiable as unnecessary for specific queries before the query is run, saving the time and cost of having to run the query over a portion of data with no gain [12]. Other partitioning methods aim to instead maximize the parallelism potential of the database, allowing queries and their subqueries to be performed simultaneously, therefore improving performance [12].

Additionally, in cloud databases, one option is to partition the data in a way that minimizes the number of multi-node transactions (i.e., most transactions should complete by touching data on only one node). The goal is to minimize the number of cross-node distributed transactions, which incur overhead both because of the extra work done on each node [9], as well as having nodes waiting on other nodes to finish processing before being able to continue. This also reduces the need for communication between the nodes.

Depending on the way the partition is performed, the partitioning system may be described as either static, dynamic, or scalable. In static partitioning systems, when a partition is formed, it cannot be changed later even if a more suitable partition would be found. In dynamic partitioning systems, the partitions are formed dynamically as changes occur to the database, often very frequently. Scalable partitions work by first creating an initial partition and then depending on the usage of the data, viewed through for instance the transaction logs and data access patterns, the data may be repartitioned periodically, usually less often than in dynamic partitioning systems. [13]

5. EXPERIMENTAL RESULTS

This section will contain performance tests performed by K. Donkena & S. Gannamani [6], R. Győrödi et al. [10], C. Wang et al [11] and S. Ahirrao & R. Ingle [13].

The tests performed by Donkena and Gannamani, and Győrödi et al. were run using the cloud platform Windows Azure to compare the performance of the cloud database when compared to an on-premises alternative. Their results should help create a picture of the performance differences between cloud and on-premises at these times, as well as the evolution of cloud performance when compared to more recent research.

The tests performed by C. Wang et al., and S. Ahirrao & R. Ingle are used as examples of the effect of applying more advanced optimization onto a database to see it's effect. The first focuses on the effect of re-optimization of a query during run time on both time and resource expenditure. The latter focuses on smarter, scalable partitioning to increase the throughput of the cloud system.

5.1 Tests by K. Donkena & S. Gannamani

The tests performed by K. Donkena and S. Gannamani were done by comparing the query response time (ms), with each different test being iterated on“at least 30 times, with an average being calculated for the iterations. The tests were performed on-premises with Microsoft SQL 2008 R2, and with cloud using Windows Azure through a web browser.

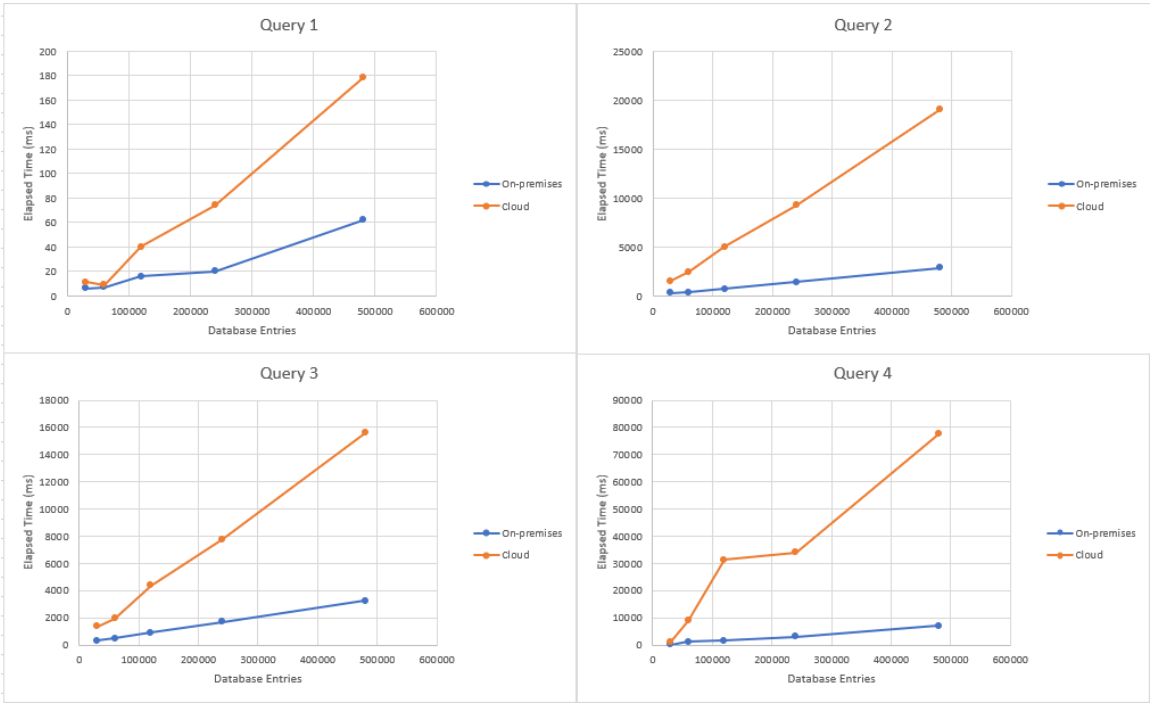

***Figure 1**. Response time values of different entries for on-premises and cloud database. [6]*

As can be seen in Figure 1, which was drawn using the test results [6], in all the tests, the on-premises database performed better. In query 1, where a small number of rows were retrieved, the cloud database performed best when compared to the rest of the tests. As the number of retrieved results in Query 2 become greater, the difference in performance between on-premises and cloud increased. Queries 3 and 4 tested their performance on a simple and a more complex join operation. The difference in performance was greater when the operation was made more complex.

From these results, the conclusion was that the cloud database performance is poor compared to that of the traditional database, which matched the hypothesis laid out by the sources used within the document. However, it should be noted that as this test is from the year 2012, it does not mirror the current performance of cloud databases and was chosen mostly as a historical example of the advancement in the optimization of cloud databases.

5.2 Tests by R. Győrödi et al.

The tests performed by R. Győrödi et al. were done using the average execution time of each of the following basic SQL operations: INSERT, SELECT, UPDATE and DELETE. Execution time for each type of operation was averaged by running a specific query for each procedure. As parallel usage is a key point in cloud services, parallel usage was simulated using a for-loop that calls the database through an API, calling to perform a procedure of the desired operation on each iteration of the loop.

Six different test cases were used consistently through all the tests. Two of them were on-premises, with one being tested after a recovery procedure to test fault tolerance, and the remaining four were different tiers of performance offered by Azure, titled Free, P1, S4 and S9. These tiers may be described by the amount of DTUs (database transaction units) the package supported, which consider computing power, storage as well as the IO resources for the cloud databases. The four tiers respectively provided 5, 125, 200 and 1200 DTUs.

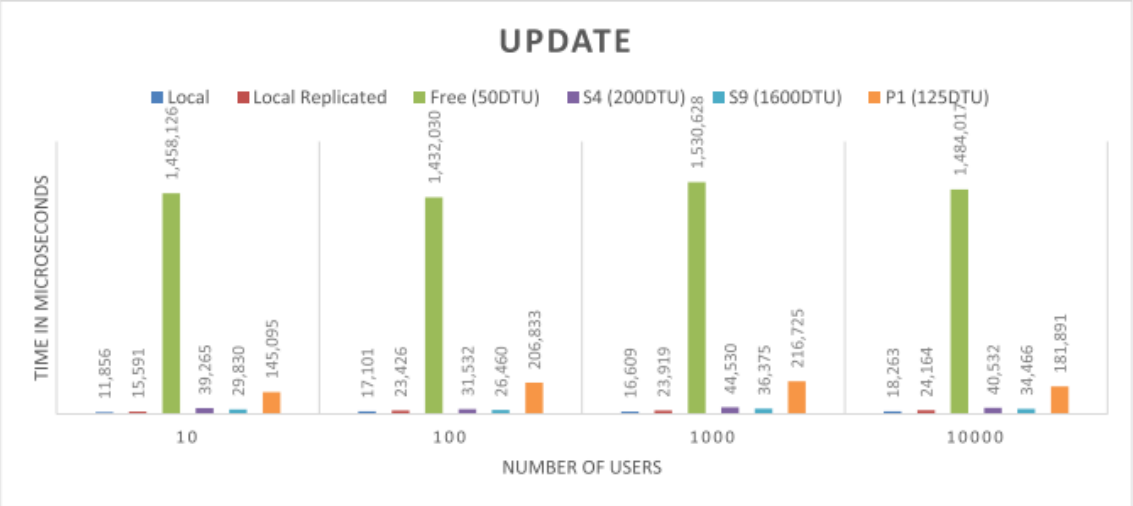

Figure 2. *Update statement experimental results. [10]*

As shown in Figure 2, the on-premises databases had the best performance when using the UPDATE operation, being around twice or thrice as fast when compared to the S9 or S4 package respectively.

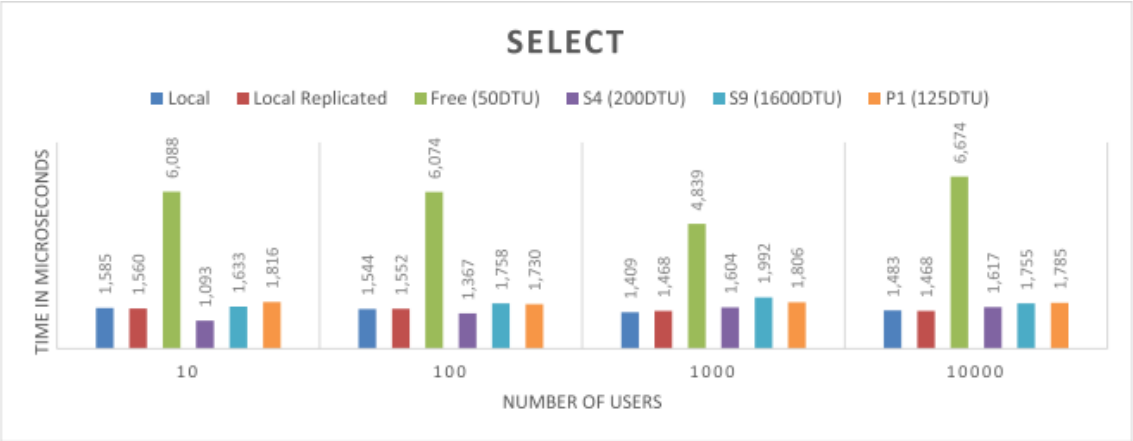

Figure 3. *Select statement experimental results. [10]*

As shown in Figure 3, the performance of both on-premises and cloud databases were somewhat similar, with on-premises being marginally faster than cloud in SELECT. However, the S4 option had better performance than any other alternative with low user count, including the more powerful S9. No explanation for this behavior was presented, and as the more powerful option from the same provider had worse performance, the results was likely the cause of random chance, such as high server load during the S9 tests.

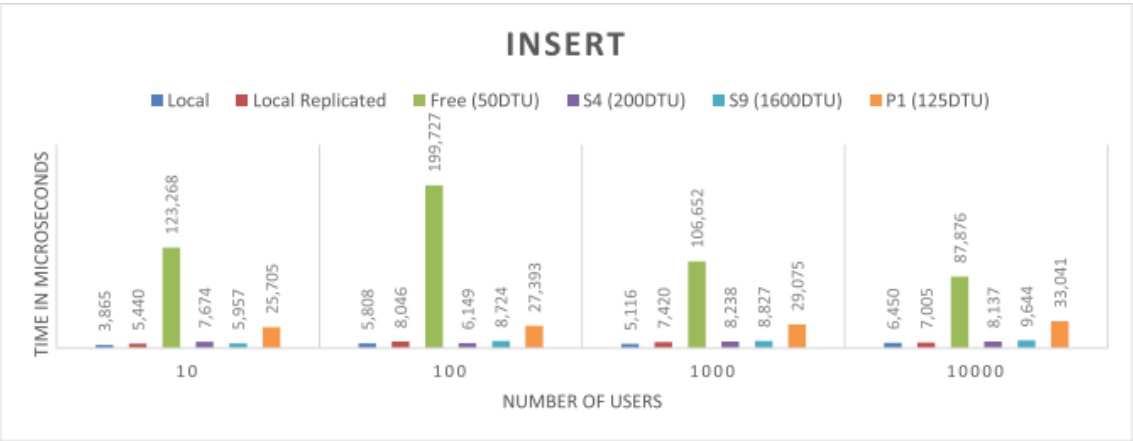

Figure 4. *Insert statement experimental results. [10]*

As shown in Figure 4, the performance of both on-premises and cloud databases were somewhat similar, with on-premises performing between 15 – 50% faster than the higher-end options.

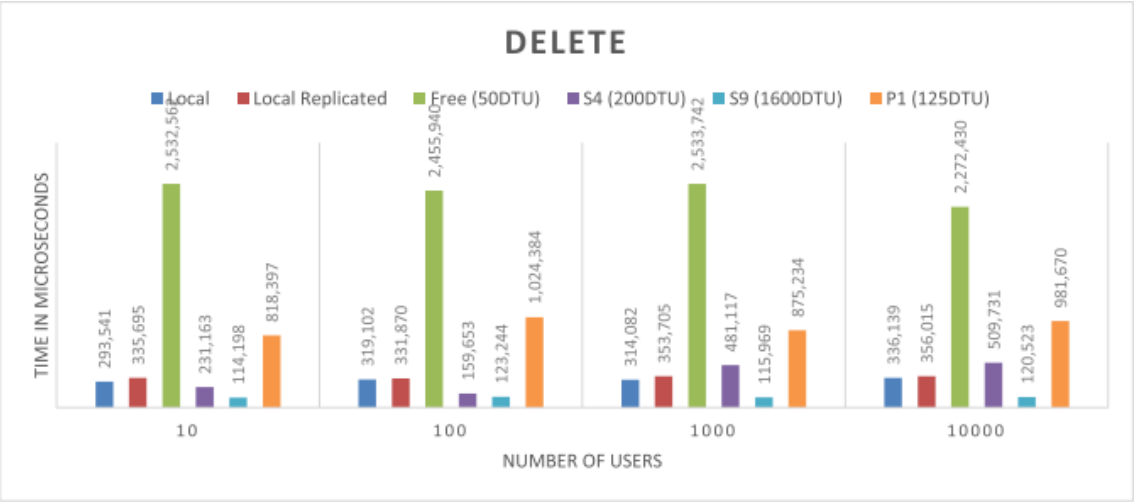

Figure 5. *Delete statement experimental results. [10]*

As shown in Figure 5, the test on the DELETE operation had the cloud outperform the on-premises databases with the higher-end options. Both S4 and S9 were faster when the user count was set to 10 or 100, and S9 was clearly faster than the on-premises databases with the rest as well, being around three times as fast as either of the on-premises alternatives with all the performed tests. While the paper did not speculate on the reason, a plausible explanation was that a partition that aided the operation call was already present within the database.

5.3 Tests by C. Wang et al.

The tests performed by C. Wang et al. were performed on a system of 10 virtual machines to simulate a cloud environment. Each of the virtual machines was given the same specifications. Two different tests were performed with the intention on seeing if the re-optimization would cause noticeable improvement in resource allocation, monetary cost, and performance. For the purposes of tests, monetary cost was calculated as:

$$Monetary\ cost = container\ amount * container\ cost / second * elapsed\ time$$

Both tests were run on 30 separate instances of the same type on each of the tested data sizes, with the average being taken for both response time and the determined monetary cost. It should be noted that the goal of the optimization algorithm is to adapt to both the time and monetary constrains. This test does not therefore clearly indicate best possible performance but acts more as a showing of how it is possible to balance and aim for agreed upon performance a service provider would grant their client, as it is in the interest of the service provider to grant the agreed upon amount of performance while minimizing their own monetary losses.

5.3.1 Parallelism Test

The first test was performed to see if the re-optimization would affect the performance or monetary cost of the query, as the query in question is able to be performed with a high degree of parallelism due to the way the sub-queries select data from multiple different partitions that may be performed in parallel.

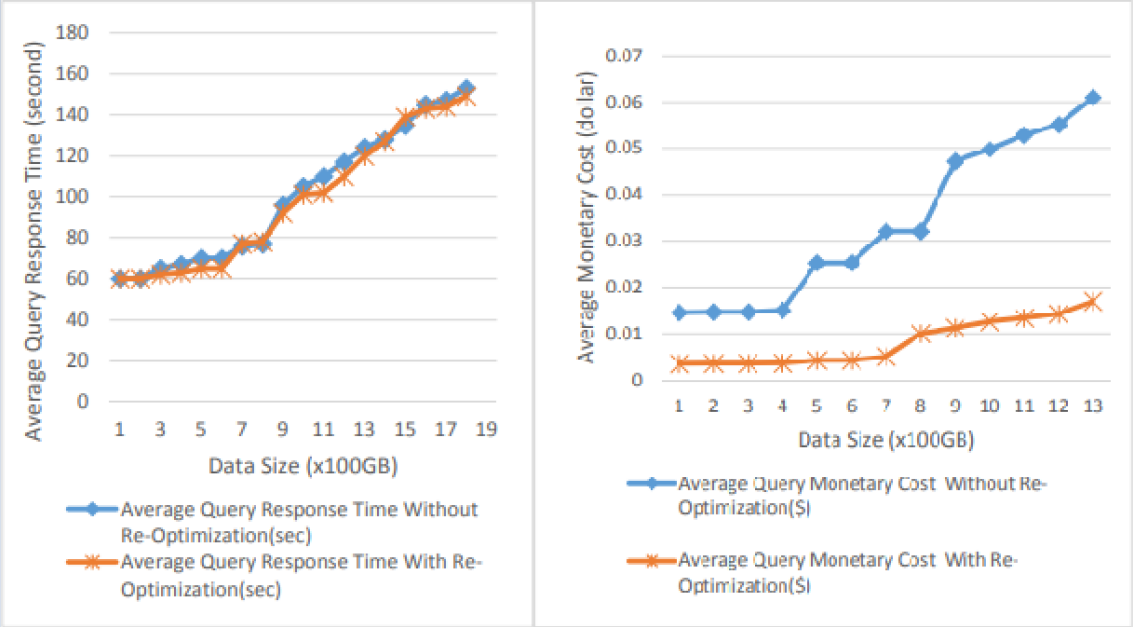

Figure 6. *Impact of data size on query response time and monetary cost for query 1. [11]*

The results of the test are shown in Figure 6. The response time of the query reduced quite little, at best within an 8% difference to the un-optimized counterpart. However, the cost of the query was massively decreased. The optimization therefore leads to a minimal increase in performance, as well as a very noticeable decrease in the estimated costs of the query.

5.3.2 Operator Change Test

The second test was performed to see how big the impact of the re-optimization has in a scenario where the data size for both sides of a join-operator are unknown, as they are results of yet unperformed subqueries. The right side of the join is purposefully designed to be of a size where it can fit into the data cache, which should in turn cause the specific type of join operator to change from a shuffle join into a broadcast join. While it was already known that a broadcast join is faster than a shuffle join in this specific environment. The test was performed to see how the adaptive re-optimization of the query would improve the performance of the query.

The results of the test are shown in Figure 7. The response time of the query was reduced by a noticeable amount, with around a 20% improvement. This is not equal to the

40% described to be the difference between shuffle and broadcast join in the created environment as the query contains many parts aside from the join, including the optimization process, which either add to or were not part of the optimization process in this example, reducing the total net time cost improvement. The monetary cost remained rather close, with a 4% difference, with the re-optimization becoming more expensive than without. This was credited to some parts of the query being executed on containers with a higher unit price after re-optimization, increasing monetary cost by improving performance.

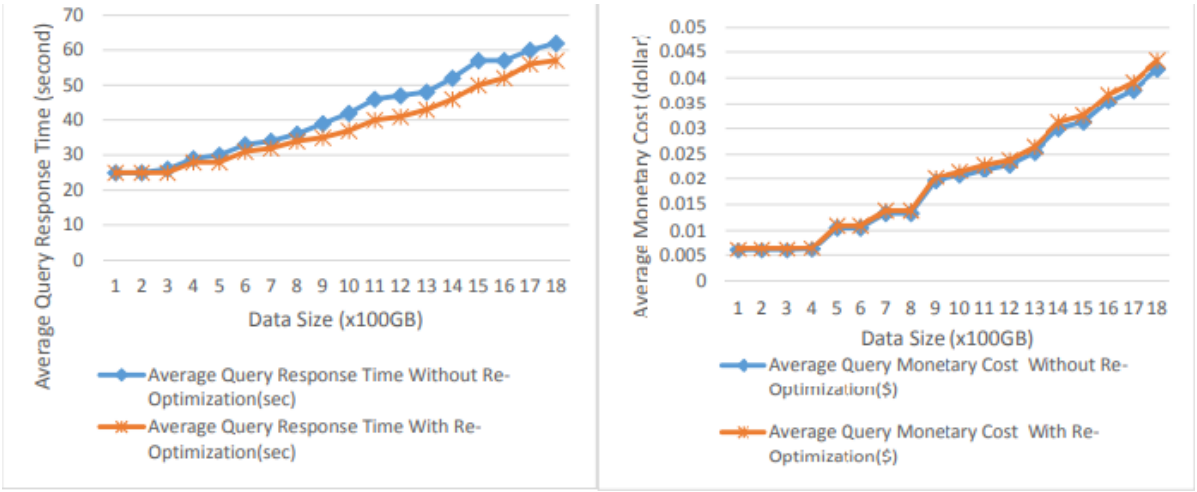

Figure 7. *Impact of data size on time and monetary cost for executing query 2. [11]*

5.4 Tests by S. Ahirrao & R. Ingle

The tests performed by S. Ahirrao & R. Ingle focused on determining the performance improvement from adapting a different, scalable partition method. They ran two tests, one on the Amazon SimpleDB cloud data store infrastructure, and another on the Hadoop HBase database management system.

Their partition method uses data structured around it's access patterns into entities called warehouses. A single partition may include multiple of these warehouses. The main purpose of these warehouses is to be able to track which warehouses supply data to one another, identifying warehouses which "are more probable to supply" data to one another. Then, using scalable partitions, the warehouses / data is moved around on the partitions to reduce the amount of multi-partition transactions and queries according to the recorded patterns.

Both tests were run using the TPC-C (Transaction Processing Performance Council Benchmark C), which is a standard benchmark using to simulate the workload of an e-

commerce application, simulating transaction types fitting of such an environment. The benchmark uses real-life data to divide the number of each transaction accordingly.

To use the developed partition method presented in the paper, both infrastructures had to first be converted to be usable with it.

Their partition method was compared to schema and graph partitioning in each of the tests. In schema partitioning, the database is partitioned according to the relations of the data. In graph partitioning, the database is partitioned according to frequently used attributes of the data.

5.4.1 Tests using Amazon SimpleDB

The tests run on the Amazon SimpleDB were run on a cluster of 5 machines, with standardized specifications for each of the machines. The database is split into 15 warehouses, and tested with benches of 250 to 5000 concurrent users.

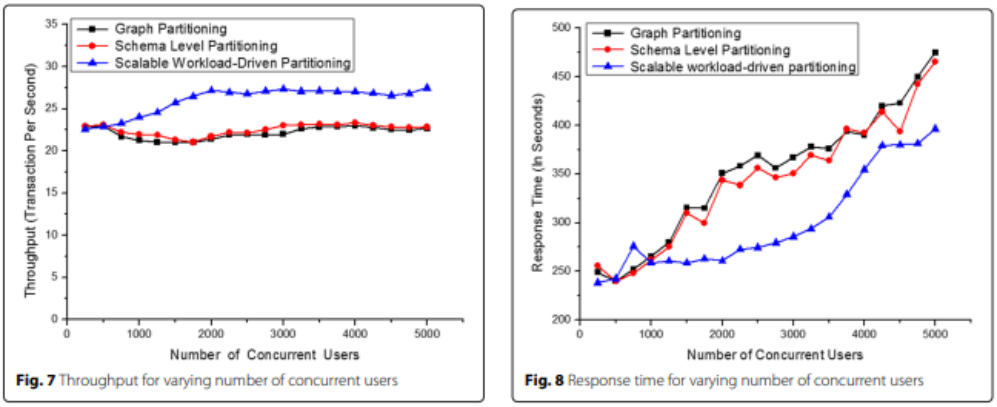

Figure 8. *Throughput and response time for varying number of concurrent users, using Amazon SimpleDB. [13]*

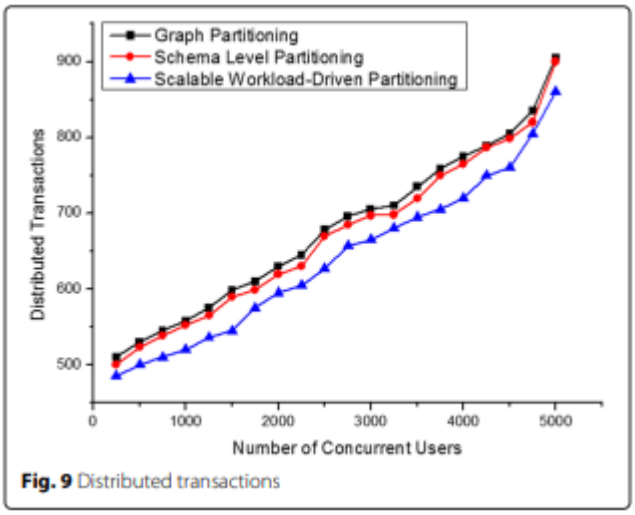

Figure 9. *Distributed transactions, using Amazon SimpleDB. [13]*

As shown in Figures 8 and 9, both the throughput and the response times are improved using the scalable partitioning method. Additionally, the number of distributed transactions was lessened when using the method, decreasing the expenditure of resources.

5.4.2 Tests using Hadoop HBase

The tests run on the Hadoop HBase infrastructure were similar to those run on the Amazon SimpleDB. The tests were run with 15, 30 and 60 warehouses used, and tested with benches of 20 to 100 concurrent users, with steps of 20.

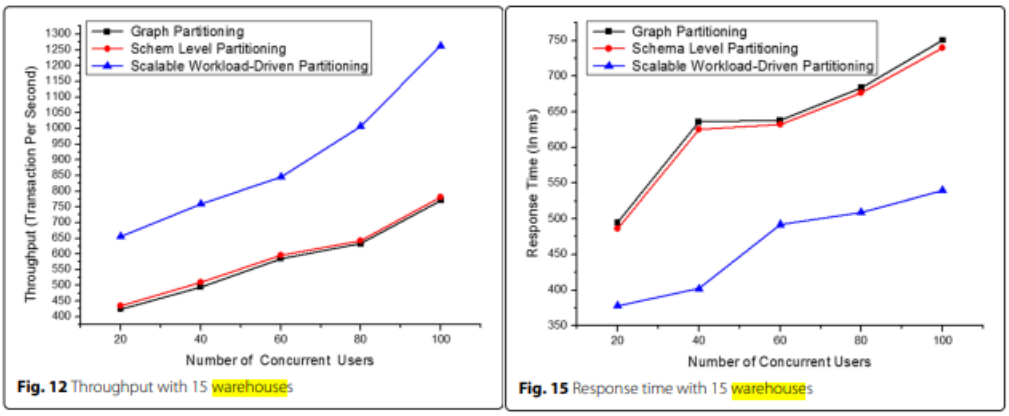

Figure 10. *Throughput and response time with 15 houses, using Hadoop HBase. [13]*

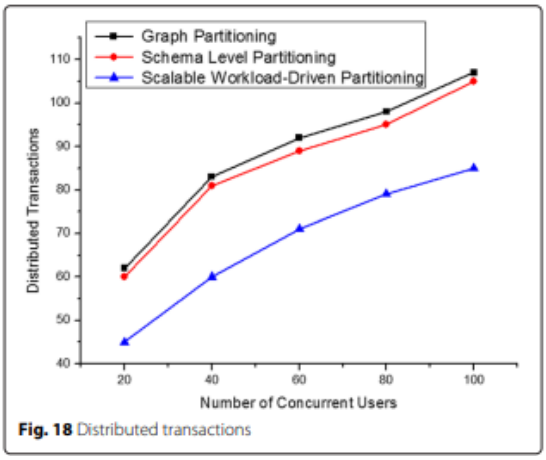

Figure 11. *Distributed transactions, using Hadoop HBase. [13]*

Only the tests for the 15-warehouse variant of the method are shown on Figure 10, as “there is no change in the performance of the system even if the database size is seen growing with 15, 30, 60 warehouses” and “it is observed that the response time is almost same with database size 15, 30, 60 warehouses”, their results mirroring that of the 15-warehouse figure. It is clearly visible that the partitioning method improved both the throughput and the response time of the database. In figure 11 it can be seen that the amount of distributed transactions was also reduced when using the partitioning method.

6. GATHERED RESULTS

From the gathered tests performed by several parties, we may see that the optimization and general performance of queries made to cloud databases has clearly improved over the years, with the most recent tests staying close or even out-performing the on-premises counterpart with the higher-end packages.

While the performance of the cloud database was clearly less than that of the on-premises alternative 10 years ago as shown by older tests [6], with modern advancements and the adoption of new technologies, as well as infrastructure improvements, cloud databases can be seen as a contender and a real option next to on-premises as shown by more recent tests [10].

Additionally, as more and more novel algorithms and partition methods are developed [11] [13], the most promising of them will eventually be taken into wider usage and improve performance through optimization.

The tests done by R. Győrödi et al. showed that while higher-end cloud alternatives are similar to or even higher in price to on-premises counterparts, the cheaper, more balanced packages may grant lesser, but still comparable performance at a fraction of the cost. For smaller companies, even free plans may be sufficient if the data flow of the company is not large, only accruing cost in the training of their staff.

7. CONCLUSION

Cloud databases have gained and will continue to possess a market share due to their ease of use and low initial investment, making them an attractive alternative to parties with less need for finetuning and exact knowledge of the operation behind the curtains.

While they will not make on-premises databases obsolete quite yet, both in part due to currently existing, working alternatives and internal resistance to changing away from them, it is likely that their market share within the data infrastructure market will keep growing, as more and more companies discover both the financial and logistical upsides out-weigh their need for personalization of the database structure.

It is also clear that cloud databases have become more viable from a performance perspective over years of development, with the stigma of their poor performance from past years fading away. The difference between similarly priced on-premises and cloud database performance has lessened, with the cloud database sometimes even outperforming the on-premises alternative. Therefore, even in very performance-intensive environments, cloud databases have been proven to be a valid option.

As on-premises databases have been in usage for multiple decades, the amount of research and already in-place optimization far outnumbers that in cloud databases. As the amount of research and wide-spread usage of optimization in cloud databases continues to grow, expected advancements in cloud databases yet to come such as more efficient cost models and optimization algorithms may have cloud databases reach or even overtake on-premises within the coming years.

The scalability of the cloud database, both up and down, should also be noted as key factors when deciding which alternative to go for. As the cloud option has less initial investment and long-term commitment, it may be wise to start and test the waters using it, especially if the use of a database is not considered mandatory, but it is seen as a possibility to increase the potential of the service in question.

REFERENCES

[1] Azhir, E., Navimipour, N. J., Hosseinzadeh, M., Sharifi, A., Darwesh, A. Query optimization mechanisms in the cloud environments: A systematic study. In: International journal of communication systems, 2019, Vol.32 (8), s.e3940-n/a

[2] Azhir, E., Navimipour, N. J., Hosseinzadeh, M., Sharifi, A., Darwesh, A. Deterministic and non-deterministic query optimization techniques in the cloud computing. In: Concurrency and computation, 2019, Vol.31 (17), s.e5240-n/a

[3] Sebaa, A. & Tari, A. Query optimization in cloud environments: challenges, taxonomy, and techniques. In: The Journal of supercomputing, 2019, Vol.75 (8), s.5420-5450

[4] Zhou, J., Bruno, N. & Jain, Z. Continuous Cloud-Scale Query Optimization and Processing. In: Proceedings of the VLDB Endowment, 2013, Vol.6 (11), s.961-972

[5] Zhao, J., Hu, X. & Meng, X. ESQP: An Efficient SQL Query Processing for Cloud Data Management. In: Proceedings of the second international workshop on cloud data management, 2010, s.1-8

[6] Donkena, K. & Gannamani, S. Performance Evaluation of Cloud Database and Traditional Database in terms of Response Time while Retrieving the Data. 2012. Available at: https://www.diva-portal.org/smash/get/diva2:829184/FULLTEXT01.pdf

[7] Khan, F. A., Jamjoom, M., Ahmad, A., Asif, M. An analytic study of architecture, security, privacy, query processing, and performance evaluation of database-as-a-service. In: Transactions on emerging telecommunications technologies, 2022, Vol.33 (2), p.n/a

[8] Brumm, B. What is a Database? 2019. Available at (cited 22.6.2022): https://learning.oreilly.com/library/view/beginning-oracle-sql/9781484244302/html/471705_1_En_1_Chapter.xhtml

[9] Jain, S. Comparative Study of Traditional Database and Cloud Computing Database. 2017. In: International Journal of Advanced Research in Computer Science, vol. 8 (2)

[10] Győrödi, R., Pavel, M. I., Győrödi, C.,Zmaranda, D. 2019. Performance of On-Prem Versus Azure SQL Server: A Case Study. In: IEEE Access, 2019, vol. 7, pp. 15894-15902, doi: 10.1109/ACCESS.2019.2893333.

[11] Wang, C., Arani, Z., Gruenwalk, L., d'Orazio, L. Adaptive Time, Monetary Cost Aware Query Optimization on Cloud Database Systems. In: 2018 IEEE International Conference on Big Data (Big Data), 2018, p.3374-3382

[12] Ling, L., Özsu, M. T. Encyclopedia of Database Systems. 2019. 1st edn. Springer Press. Available at (cited 20.8.2022): https://link-springer-com.libproxy.tuni.fi/referencework/10.1007/978-0-387-39940-9

[13] Ahirrao, S., Ingle, R. Scalable transactions in cloud data stores. In: Journal of cloud computing: advances, systems and applications, 2015, Vol.4 (1), p.1-14

[14] priyankagujral (username), SQL | Query Processing. 2018. Available at (cited 20.8.2022): https://www.geeksforgeeks.org/sql-query-processing/

[15] Lakshmi, N. G. Database Migration on Premises to AWS RDS. In: EAI Endorsed Transactions on Cloud Systems, 2018, Vol.3 (11), p.154463

APPENDIX A

Response time for	Retrieved results	On-premises database (ms)	Cloud database (ms)
Query 1			
30000 Entries	15	6	11
60000 Entries	15	7	9
120000 Entries	38	16	40
240000 Entries	89	20	74
480000 Entries	184	62	178
Query 2			
30000 Entries	25667	310	1546
60000 Entries	48360	387	2452
120000 Entries	95981	739	4996
240000 Entries	190696	1421	9287
480000 Entries	380755	2836	19056
Query 3			
30000 Entries	24499	324	1373
60000 Entries	48999	465	1928
120000 Entries	97999	885	4359
240000 Entries	195999	1690	7777
480000 Entries	391999	3235	15587
Query 4			
30000 Entries	8204	204	1107
60000 Entries	32430	1097	8921
120000 Entries	65006	1590	31258
240000 Entries	130479	3080	33973
480000 Entries	261537	7083	77654

Results used for the creation of Figure 1. [6]