ISSN (e): 2250 – 3005 || Volume, 10 || Issue, 7|| July – 2020 ||
International Journal of Computational Engineering Research (IJCER)

Cloud Query Processor Using Cloud Databases

[1]S. BastinBritto[2]P. Dinadayalan[3]N. Gnanambigai[1]VettriSelvi A
[1]Research Scholar, Bharathiyar University, Coimbatore,India
[2]Department of Computer Science,KanchiMamunivar Govt. Institute for Post- Graduate Studies and Research, Puducherry, India
[3]Department of Computer Science, Indira Gandhi College of Arts and Science, Puducherry, India
Corresponding Author: S. BastinBritto

ABSTRACT
'Cloud Query Processor usingCloud Databases' is a new approach which combines Cloud Computing and Database for join query processing. This work tries to reveal the true power of Cloud Computing for database applications.A Cloud database is a distributed database thatdelivers computing as a service. Joins are one of the most important operations performed by a cloud database system. 'A Cloud Data Query Processor' is a new cloud data accessing method for efficient query processing which uses the SQL Join queries in cloud database. We propose a framework to systematically relax queries involving joins and selections. The objective is to create a solution that is viable for very large cloud datasets, thus allowing cloud data to be used as a solution for development of cloud databases. The bottleneck in the querying of large cloud datasets is performing joins and unions. Using this approach, we create tables that can be accessed and queried more efficiently. We introduce join tables that store the results of join executions. A Cloud Data Query Processor is a scalable and persistent data model for cloud data storage that improves the performance of queries.

KEYWORDS: *Cloud Computing; Virtual Machine; Cloud Database; Web Service; NoSQL*

Date of Submission: 29-08-2020 Date of acceptance: 14-09-2020

I. INTRODUCTION

Cloud computing is the use of computing resources such as hardware and software that are delivered as a service over Internet [9][10][11]. Cloud computing [9][10][11] provides computer resources on demand, in a self-service manner. Its main characteristics are: Minimal initial investments in cost and time, only pay for what you actually use and nearly unlimited scalability. Technically, this is realized by using virtualization and web technologies to enable easily accessible multi-tenancy environments. A database is structured collection of data. Databases [12][13][14] may be stored on a computer and examined using a program. These programs are often called databases, but more strictly are database management systems. One of the most common and powerful models is the relational model, and programs which use this model are known as relational database management systems.Relational databases are often normalized eliminate duplication of information when objects may have one-to-many relationships. A Cloud database management system (CDBMS) is a distributed database that delivers computing as a service instead of a product. It is the sharing of resources, software, and information between multiply devices over a network which is mostly the internet. A cloud database is a database that has been optimized or built for a virtualized computing environment.. A join [4][7] is an SQL operation performed to establish a connection between two or more database tables based on matching columns, thereby creating a relationship between the tables. Most complex queries in an SQL database management system involve join commands. Joins are one of the most important operations performed by a relational database system. An RDBMS [12][13][14] uses joins to match rows from one table with rows from another table.

Experiences gained in the last decade from some of the technology leaders that provide services over the Internet indicate that application infrastructures in the cloud context should be highly reliable, available, and scalable [15][16][17]. Reliability is a key requirement to ensure continuous access to a service and is defined as the probability that a given application or system will be functioning when needed as measured over a given period of time. Similarly, availability is the percentage of times that a given system will be functioning as required. The need for scalable design is to ensure that the system capacity can be augmented by adding

additional hardware resources whenever warranted by load fluctuations. Thus, scalability has emerged both as a critical requirement as well as a fundamental challenge in the context of cloud computing. In the context of most cloud-based application and service deployments, data and the database management system(DBMS) is an integral technology component in the overall service architecture. The reason for the creation of DBMS, in the cloud computing space is due to the success DBMSs and in particular Relational DBMSs have had in modeling a wide variety of applications. The key ingredients to this success are due to many features DBMSs offer: overall functionality (modeling diverse types of application using the relational model which is intuitive and relatively simple), consistency, performance, and reliability [15][16][17]. Section 2 elaborates a detailed study on cloud database with join queries.Section 3 introduces the new Cloud Data Query Processor for cloud computing. The architecture shows the components of Cloud Data Query Processor in section 3. Section 4 demonstrates the performance analysis. Section 5 summarizes the conclusion and gives future research directions. It also point outs the advantages and performance of proposed work.

II. RELATED WORKS

A cloud database is a database that typically runs on a cloud computing platform. There are two common deployment models: users can run databases on the cloud independently, using a virtual machine image, or they can purchase access to a database service, maintained by a cloud database provider. There are two common deployment models: users can run databases on the cloud independently, using a virtual machine image, or they can purchase access to a database service, maintained by a cloud database provider. There are two primary methods to run a database on the cloud:Virtual machine Image and Database as a service. In Virtual machine Image, Cloud platforms allow users to purchase virtual machine instances for a limited time. It is possible to run a database on these virtual machines. Users can either upload their own machine image with a database installed on it, or use ready-made machine images that already include an optimized installation of a database. Some cloud platforms offer options for using a database as a service, without physically launching a virtual machine instance for the database. In this configuration, application owners do not have to install and maintain the database on their own. Instead, the database service provider takes responsibility for installing and maintaining the database, and application owners pay according to their usage. A third option is managed database hosting on the cloud, where the database is not offered as a service, but the cloud provider hosts the database and manages it on the application owner's behalf.

Most database services offer web-based consoles, which the end user can use to provision and configure database instances. For example, the Amazon Web Services web console enables users to launch database instances, create snapshots (similar to backups) of databases, and monitor database statistics. Database services [15][16][17] consist of a database manager component, which controls the underlying database instances using a service API. The service API is exposed to the end user, and permits users to perform maintenance and scaling operations on their database instances. Database services make the underlying software stack transparent to the user - the stack typically includes the operating system, the database and third-party software used by the database. The service provider is responsible for installing, patching and updating the underlying software stack. Database services take care of scalability and high availability of the database.

Data model is also important to differentiate between cloud databases which are relational as opposed to non-relational or NoSQL: SQL database and NoSQL databases [4][10][11].SQL database such as NuoDB, Oracle Database, Microsoft SQL Server, and MySQL, are one type of database which can be run on the cloud either as a Virtual Machine Image or as a service, depending on the vendor. SQL databases are difficult to scale, meaning they are not natively suited to a cloud environment, although cloud database services based on SQL are attempting to address this challenge. NoSQL databases such as Apache Cassandra, CouchDB and MongoDB, are another type of database which can run on the cloud. NoSQL databases are built to service heavy read/write loads and are able scale up and down easily, and therefore they are more natively suited to running on the cloud. However, most contemporary applications are built around an SQL data model, so working with NoSQL databases often requires a complete rewrite of application code.

The current DBMS technology fails to provide adequate tools and guidance if an existing database deployment needs to scale-out from a few machines to a large number of machines. Technology leaders such as Google, Amazon, and Microsoft have demonstrated that data centers provide unprecedented economies-of-scale since multiple applications can share a common infrastructure [4][5][7][8][10]. All three companies have taken this notion of sharingbeyond their internal applications and provide frameworks such as Amazon's AWS, Google's AppEngine, and Microsoft Azure for hosting third-party applications in their respective data-center infrastructures. The main distinction is that in traditional DBMSs, all data within a database is treated as a "whole" and it is the responsibility of the DBMS to guarantee the consistency of the entire data. The requirement of making web-based applications scalable in cloud-computing platforms arises primarily to support virtually unlimited number of end-users. Scalability of a system only provides us a guarantee that a system can be scaled up from a few machines to a larger number of machines. All database systems must be

able to respond to requests for information from the user(process queries). Obtaining the desired information from a database system in a predictable and reliable fashion is the scientific art of Query Processing. These results back in a timely manner deals with the technique of Cloud Data Query Processor. The new data accessing technique introduces the architecture of Cloud Data Query Processor to the concepts of join query processing in the cloud database domain.

III. ARCHITECTURE OF CLOUD DATA QUERY PROCESSOR

The Cloud Data Query Processor is data accessing technique for join queries. The architecture of Cloud Data Query Processor is shown in figure 1. It consists of User Interface, Join Query Generator, Cloud Database and Join Query Producer. The User Interface component is the interface between user query and Join Query Generator. It receives the input from the user and forwards the query to Join Query Generator. Join Query Generator is a major component that provides the interface between the data stored in the cloud database and the application programs and queries submitted to the system. Join Query Producer is join query output producer to the user.

The User Interface component is used for client queries. This component is interacting with Join Query Generator. The client sent the Join queries to the next-level (Join Query Generator). More than one client can forward queries at the same. This component handles more number of queries at a time.

The heart of Cloud Data Query Processor is Join Query Generator. It handles all types join queries like inner join, outer join, full outer join, left outer join, right outer join, cross join, natural join and self-join. The User Interface component interacts with the Join Query Generator, and the Join Query Generator in turn interacts with the Cloud database. The Join Query Generator accepts join-SQL syntax, selects a plan for executing the syntax, and then executes the chosen plan. The Join Query Generator isolates the user from the details of execution: The user specifies the result, and the Join Query Generator determines how this result is obtained. Cloud Data Query Processor performs the following functions: accept SQL queries from clients, determine the type of data to access, transform SQL queries into database access language, optimize queries, and create relational result sets from database records and process post-query result sets as needed.

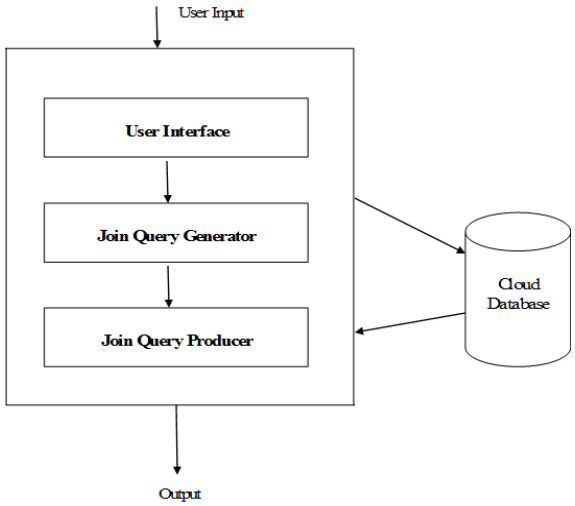

Figure: 3.1. The architecture of Cloud Data Query Processor

Cloud Database performs all data access. The role of the Cloud Database is service-based. The Cloud Database consists of services. A major service embedded in the Cloud Database is the Join Query Generator that acts as the relational engine. Each join query running in a cloud database processes requests for one or more user connections. When a user connects to the cloud database, Join Query Generator is assigned to the connection until the client disconnects from the cloud database. For each client connection, the Join Query Generator also maintains a separate transaction. A new transaction is implicitly started whenever the client issues a data access call after either connecting to the cloud database or ending a previous transaction with a request.

Join Query Producer is fourth component of Cloud Data Query Processor. The Join Query Producer generates the output of the given join queries. It interacts with Join Query Generator and Cloud Databases to execute several processes when a query is submitted. The Join query Producer covers the processing that takes place from the time a query is submitted until the time results are generated. The role of Join Query Processor is to find result in one or more databases and deliver it to the user quickly and efficiently.

IV. PERFORMANCE ANALYSIS

From the simulations, we have conducted experimental measures for join queries using Cloud Data Query Processor. We have made five tests for Cloud Data Query Processor using Cloud database. Each test contains 200 samples of datasets for inner join, left outer join, right outer join, full outer join and cross join of join queries.

Table: 4.1 Simulation Result Datasets for Cloud Data Query Processor

Join Query	No of Dataset	Accept	Reject	No Result
Inner Join	200	195	3	2
Left Outer Join	200	192	5	3
Right OuterJoin	200	188	10	2
Full outer Join	200	190	9	1
Cross Join	200	185	9	6

In Inner join query, 200 samples datasets have taken in which 195 datasets were given proper result, 3 datasets were rejected and 2 datasets were no result. The inner join datasets accepted percentage was 97.5%, the rejected percentage was 1.5% and the no result percentage was 1%.

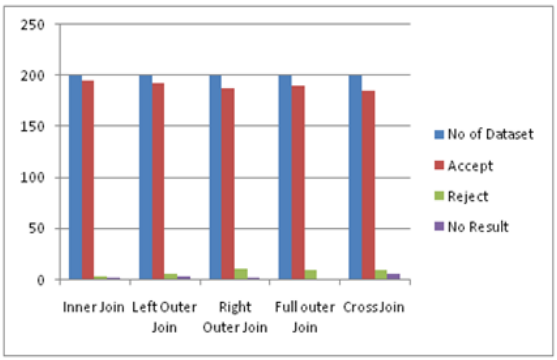

Figure: 4.1 Performance analysis for Cloud Data Query Processor

In Left outer join query, 200 samples datasets have taken in which 192 datasets were given proper result, 5 datasets were rejected and 3 datasets were no result. In Left outer join, the percentages of accepted, rejected and the no result of datasets were 96%, 2.5% and 1.5% respectively. In Right outer join query, 200 samples datasets have taken in which 188 datasets were given proper result, 10 datasets were rejected and 2 datasets were no result. The Right outer join datasets accepted percentage was 94%, the rejected percentage was 5% and the no result percentage was 1%.

In full outer join query, 200 samples datasets have taken in which 190 datasets were given proper result, 9 datasets were rejected and 1 dataset were no result. In Full outer join, the percentages of accepted, rejected and the no result of datasets were 95%, 4.5% and 0.5% respectively. In Cross join query, 200 samples datasets have taken in which 185 datasets were given proper result, 9 datasets were rejected and 6 datasets were no result. The Cross join datasets accepted percentage was 92.5% the rejected percentage was 4.5% and the no result percentage was 3%. From the simulation shown in the table 4.6, the Cloud Data Query Processor has been accepted 95% of datasets from the cloud database and 5% of datasets were rejected and no result in the table. Therefore the Cloud Data Query Processor is better data organization for Join queries.

Table : 4.2. Simulation Percentage of Accept, Reject And No Result datasets for Cloud Data Query Processor

Join Query	Accept	Reject	No Result
Inner Join	97.5%	1.5%	1%
Left Outer Join	96%	2.5%	1.5%
Right Outer Join	94%	5%	1%
Full outer Join	95%	4.5%	0.5%
Cross Join	92.5%	4.5%	3%
Total Percentage	95%	3.6%	1.4%

V. CONCLUSION

A detailed survey on 'Cloud Query Processor using Cloud Databases' resulted in the following conclusions. It has been observed that 'Cloud Query Processor using Cloud Databases' is used in the area of Cloud Computing. It contains work on Cloud Computing and database technology for join queries. The proposed framework provided a Cloud Database model that achieved both inner joins and outer joins of query processing. From the simulations, this structure suits for performing inner joins, outer joins, full join and semi-join are clearly explained with illustrations.Cloud Query Processor using Cloud Databases is answering all kind of join queries such as inner join, left outer join, right outer join, full outer join and cross join. The performance analysis shows that theCloud Query Processor using Cloud Databases is efficient and effective cloud data accessing for join queries.Future work consists in looking for better measures to incorporate this work with Data Mining for Clustering.It can also be developed for Object-Oriented Database System.

REFERENCES

[1]. AlpaJain,AnHaiDoan,LuisGravano, "Optimizing SQL Queries over Text Databases", ICDE '08: Proceedings of the 2008 IEEE 24th International Conference on Data EngineeringApril 2008 Pages 636–645.

[2]. Annita N. Wilschut and Jan Flokstra and Peter M.G. Apers, "Parallel Evaluation of Multi-join Queries", University of Twente, the Netherlands, 2012.

[3]. Antonio Corral ,YannisManolopoulos , YannisTheodoridis ,Michael Vassilakopoulos, "Cost models for distance joins queries using R-trees", 2005.

[4]. AreeratTrongratsameethong and Jarernsri L. Mitrpanont, "ESU-GOO: The join order algorithm for optimizing small join queries", Faculty of Information and Communication Technology, Mahidol University, Bangkok, Thailand, 2012.

[5]. Cyrus Shahabi, Latifur Khan, Dennis McLeod, "A Probe-Based Technique to Optimize Join Queries in Distributed Internet Databases" Integrated Media Systems Center and Department of Computer Science, University of Southern California, Los Angeles, CA, USA, 2000.

[6]. Dabek, J. Li, E. Sit, J. Robertson, M. FransKaashoek, and R. Morris. Designing a DHT for low latency and high throughput. In 1st Symposium on Networked Systems Design and Implementation, pages 85-98, 2004.

[7]. Donovan A. Schneider,David J. Dewitt, "Tradeoffs in Processing Complex Join Queries via Hashing in Multiprocessor Database Machines" Computer Sciences Department University of Wisconsin Madison, WI 53706,1990.

[8]. D. ZeinalipourYazti,Z. Vagena, D. Gunopulos, V. Kalogeraki, V. Tsotras, M. Vlachos N. Koudas D. SrivastavaIBM T.J Watson "The Threshold Join Algorithm for Top-k Queries in Distributed Sensor Networks", University of California Riverside, CA, USA ,csyiazti, foula, dg, vana, tsotras, Research University of Toronto AT&T Research Labs Hawthorne, NY, USA Toronto, ON, Canada Florham Park, NJ, USA, 2005.

[9]. Ihab F. IlyasWalid G. Aref Ahmed K. Elmagarmid, "Supporting Top-k Join Queries in Relational Databases",Department of Computer Sciences, Purdue University West Lafayette IN 47907-1398 filyas,aref, 2003.

[10]. James P. McGlothlin, R. Khan , " RDFJoin: A Scalable Data Model for Persistence andEfficient Querying of RDF Datasets" The University of Texas at DallasRichardson , 2009

[11]. Justin J. Levandoski Mohamed E. Khalefa Mohamed F. Mokbel, "PermJoin: An Efficient Algorithm for Producing Early Results in Multi-join Query Plans", Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN..

[12]. Maria Spiliopoulou, MichalisHatzopoulos, Costas Vassilakis, "Using Parallelism and Pipeline for the Optimisation of Join Queries"Department of Informatics, University of Athens Panepistmiopolis, TYPA Bldgs, Ilisia,GR 157 71,1999.

[13]. M.A. Kashem, Abu SayedChowdhury, Rupam Deb, and MoslemaJahan, "Query Optimization on Relational Databases for Supporting Top-k Query Processing Techniques",2010.

[14]. Nick Koudas1 Chen Li, Anthony K. H. Tung, RaresVernica"Relaxing Join and Selection Queries" University of Toronto, CanadaUniversity of California, Irvine, USANational University of Singapore, Singapore,2006.

[15]. Pankaj K. Agarwal Duke, "Input-Sensitive Scalable Continuous Join Query Processing", University JunyiXie Oracle Corporation, Jun Yang Duke University and Hai Yu Google Inc, 2009.

[16]. QiangWang ,"Efficient Range and Join Query Processing in Massively Distributed Peer-to-Peer Networks", 2008.

[17]. SwarupAcharya Phillip B. Gibbons ViswanathPoosala Sridhar Ramaswamy, "Join Synopses for Approximate Query Answering, Information Sciences Research Center Bell Laboratories 600 Mountain Avenue Murray Hill NJ 07974 November 5, 1998.