International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-9 Issue-2, December 2019

Data Selection Strategy from Database for increasing availability in Cloud Hosted Database

Nikhil S. Gajjam, T. Gunasekhar

Abstract: ***Cloud is service-oriented computing emerged as a cost-effective paradigm for distributed applications because of its unique features. Distributed Datacenters of the cloud service provider allowed researchers to do their research in Cloud environment with the factors like increasing availability, reducing response time, Fault tolerance, performance etc. This paper aims at Problem of data selection from database for partitioning, Partitioning the databases relations, placing the partition in different data centers of the cloud with respect to Cloud Hosted Database. Scope of this work will focus on read intensive data operations/queries. In this paper, we focused on data selection and data partitioning technique so as to reduce overall network latency on cloud hosted database. General log is for finding the frequently accessed rows from the database. Experiment is carried on Amazon RDS.***

Keywords : Cloud hosted database, Availability, Partitioning

I. INTRODUCTION

Cloud Computing is a service model that provides on demand network access to computing resources to user over the internet. Computing resources include servers, network, database, storage etc. Services provided by Cloud computing is called as XaaS(Everything-as-a Service). Database-as-a Service (DaaS) is one of the services that cloud provides to the users. Database stored in Cloud Servers are of 2 types. One is Relational database and other is NOSQL.

SQL:

Traditional database systems for storage are based on the relational model and are accessed by the language called as Structured Query Language (SQL). The relational database is a data structure where data is stored in the form of multiple tables and each table is having key or index to access row.

The relational model has its advantages in the documentation, simplicity, familiarity, data integrity and reliability [31]. Relational database model on cloud environment has advantageous as it can handle more complex queries and aggregation.

Disadvantages of SQL on the cloud are scalability, performance, security. As multiple data centers are present in nowadays, a relation between the data in one of the challenging task when it comes to the scalability issue.

Revised Manuscript Received on December 30, 2019.
***** Correspondence Author
Nikhil S. Gajjam*, Research Scholar, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, AP, India.. nikhilgajjam@gmail.com
Dr. T. Gunasekhar, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, AP, India. tgunasekhar@kluniversity.in

Security is another issue as RDBMS implementations in the cloud are vulnerable to SQL injection [31]. Still, RDBMS is universally considered the best choice for NOSQL as it guarantees ACID properties of the transaction.
Applications were highly structured data and complex settings are required, RDBMS should be used rather than NOSQL. Another disadvantage of SQL is time consuming in designing of an overall schema for the database. Amazon RDS and Microsoft SQL Azure are the famous examples of the relational database on a cloud

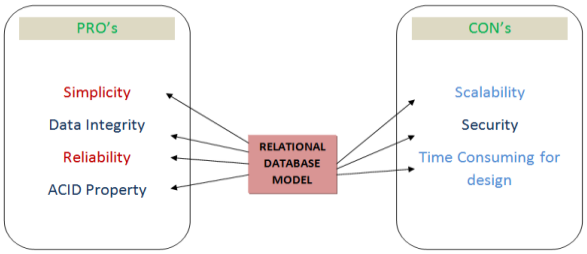

Fig 1: Pros and Cons of Relational Database System

NoSQL:

Another form of data store is called Not only SQL (NoSQL), has gained the importance to host applications having properties like scalability and availability. Applications on NoSQL databases don't require ACID properties of relational database systems. Presently when application majorly relies on availability, scalability and simplicity then NoSQL database is preferred. NoSQL is best suited for unstructured data as they are stored in document wise not in a table. Scalability is the major advantage of using NoSQL. Features of NoSQL [31] are quick data access, huge storage, scalability, low cost. So big companies like Google, Amazon, Yahoo uses NoSQL as data collected by their users is huge.

TABLE I
DIFFERENCE IN SQL AND NOSQL

	Relational Database	NOSQL
Performance	High	Low
Reliability	Low	High
Scalability	High	High but complex
Storage	High	Medium
Consistency	Low	High

SQL service offerings by Cloud Service Providers
AWS RDS

The Amazon Relational Database Service is a service provided by AWS for managing Relational database. Using SQL, users can access the data stored in the database.

AWS provides cost-effective, self-managed, industry-standard, scalable relational database service through RDS [32].

When availability and reliability is concerned, RDS is provides automated read-replicas those are stored in multiple regions across the world. During updates, RDS provides maintenance window.

IBM DB2

IBM DB2 is collection of servers developed by IBM for Linux, Windows, Unix with features like low administration, low cost

Most of the research work has been carried out regarding increasing availability, replication, partitioning, and data placement with respect to DaaS. In this paper, we focus on Relational database as a part of DaaS. This paper focuses on following problem with respect to Cloud Hosted Database

1.Problem of data selection from database for partitioning

2.Strategy for partitioning the database

1. Data selection for partitioning

Traditional way of increasing availability is through replication of complete schema or database at different data centers in the cloud. But when replication strategy has been adopted then it is difficult in taking decision of number of replicas and where to place those replicas [12]. Amount of data to be replicate is also a challenging issue as it directly affects on cost of data transfer.

Instead of replicating complete schema from database we select only important data should be replicated. Now this selection of the data based on factors like frequently accessed, workload aware etc.[14][20]. In our approach, we identify those rows from each schema which have been accessed frequently. This approach does the analysis of access pattern of database for each user.

2. Partitioning the data

Partitioning the relational database on cloud is performed for achieving 1. Scaling the single database 2. Granual placement and load balancing [33].

Aim of partitioning the schema based on access pattern is to increase the performance by decreasing data access cost. Horizontal and vertical are two partitioning techniques available for dividing the database relation into multiple fragments. Our approach will create partition from the original schema, and it will contain only those tuples which have been accessed frequently.

II. LITERATURE SURVEY:

1. Data selection strategies

Jon Olav Hauglid[13] et al. presented a paper for dynamic fragmentation and replica management on distributed database. He presented a strategy for fragmentation, replication and reallocation so as to reduce remote data access. This strategy is based on history of data access. Results are produced using DASCOSA distributed database but not on Cloud platform.

Rimma Nehme[19] et al. published a paper on Automated Partitioning Design in Parallel Database Systems. In this paper an author presented a partitioning advisor that recommends which data is to be replicated and which columns should be distributed across the network.

2. Data Placement strategies

Jinghui Zhang[34] et al. proposed an approach "Efficient Location-Aware Data Placement for Data-Intensive Applications in Geo-distributed Scientific Data Centers". He proposed a method in which data should be placed closer to the application for reducing network latency. He has implemented this approach for scientific data not for database.

Abdul Quamar[2] et al. presented a paper on "SWORD: Scalable Workload-Aware Data Placement for Transactional Workloads". He proposed a approach for data partitioning and replicating the data based on transactional workload. This approach considered for OLTP workloads. Transparency is not provided by this approach.

III. METHODOLOGY:

Objectives of the this approach are

1. To identify the data from Cloud hosted Database based on frequency of user's data access request.
2. To find strategy for data partition transparently.

Data Selection Strategy

Whenever user executes the query, database server stores the log of each executed query at back end. Log analysis based on general log, slow log etc. to find out which tables have been accessed frequently and from which region it is accessed. Finding data access pattern includes most frequently accessed tables, columns, rows, join attributes, IP address of application, AWS region etc.

Data Partitioning and Replication

Identify the rows from the tables which have been accessed frequently with some threshold value. Use proper horizontal partitioning technique for partitioning the table dynamically. Replicate this data by creating copy of replication.

Data Selection Strategy

Let S be the Schema in the database.

$FREQ_T$ be the frequency of the tuple T from S accessed by any user U.

$FREQ_{THRESHOLD}$ be the threshold frequency calculated by empirical analysis or by experiment.

$$F(T)= \begin{cases} 1 & \text{if } FREQ_T > FREQ_{THRESHOULD} \\ 0 & \text{if } FREQ_T < FREQ_{THRESHOULD} \end{cases}$$

S_{PART} is the part of Schema S which consists of tuples which are accessed frequently.

$S_{PART}=\{ T \mid T \in S \text{ and } T \text{ satisfies } F(T)\}$

Objective of this method is to find out S_{PART}

Data Placement

Once the data is partitioned by strategy mentioned in Phase 2, data placement strategy finds the region for placing the partitioned data based on following factors

- Access Time for execution of query.
- Performance

Once the data selection algorithm identifies the data to be replicate, replication algorithm replicate this data to the nearest AWS region to avail advantage of data locality property.

Data Placement

Let DC_1, DC_2, DC_3,…DC_N be the available data centers.
Let T_i be the time required for any user U to access the tuple T present at the DC_i
where i=1,2,3….N
$T_{MIN} = \min(T_i)$
Make $DC_X = DC_{LOCAL}$
Where DC_{LOCAL} is the local data center for User U
So, For User U_i
Create partition R_{PART} from relation R and store R_{PART} at location DC_{LOCAL}.

IV. RESULTS AND DISCUSSION

Experiments are carried out on AWS RDS and using database client- HeadiSQL. Once the instance created on AWS-RDS analysis is logged on to general log file. To work with data selection from the database, we focused on answering following questions

1. Which tables have been accessed most?
2. Which columns have been accessed most?
3. Identifying the AWS region from where most request have been requested?
4. Identifying the number of times each type of query fired?
5. Most number of times rows have been accessed?

JSQLparser is used for analyzing the each query from general log file to find out answers for each questions mentioned above. Following are the results achieved during our experiments.

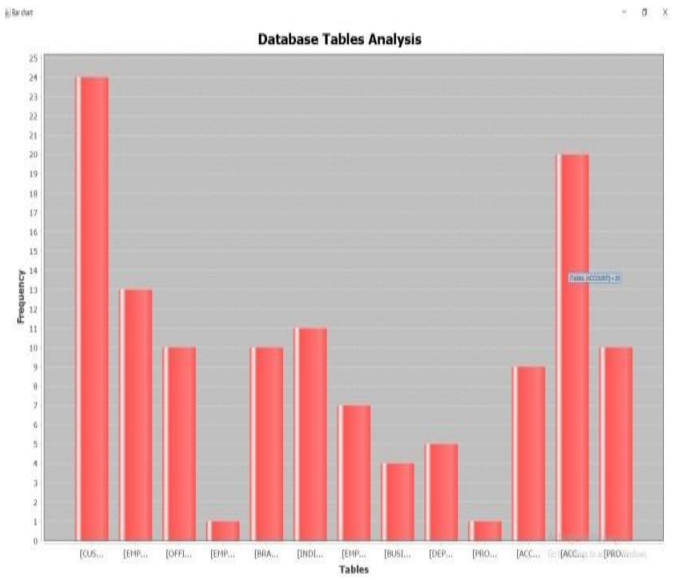

Fig 2: Count of accessing each table from database

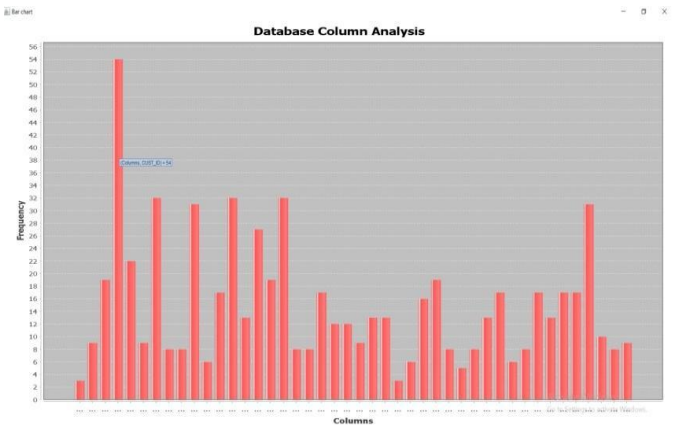

Fig 3: Count of accessing each column from database

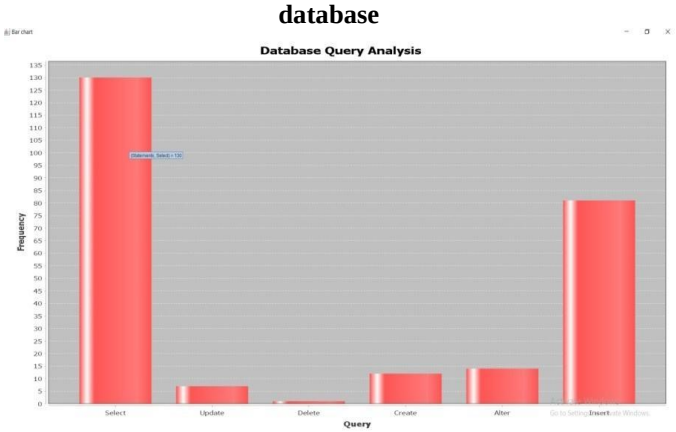

Fig 4: Count of accessing each type of query

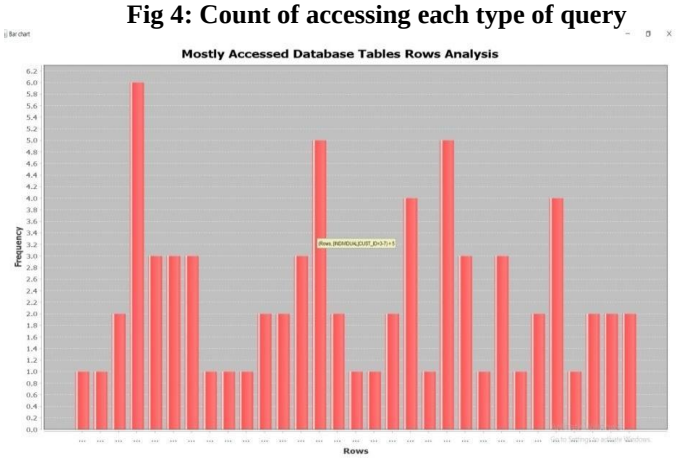

Fig 5: Count of accessing each row from table

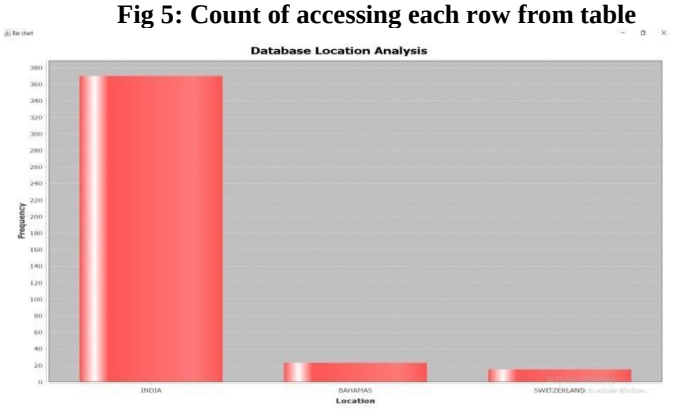

Fig 6: Identifying country and count of query request from country

V. CONCLUSION

In this paper, we focused on following problems: Problem of data selection from database for partitioning, partitioning the databases relations with respect to Cloud Hosted Database. Advantage of this work is to minimize the network latency for user by storing frequently accessed data in region nearer to the user. This approach will be effective for data intensive application hosted on cloud environment.

REFERENCES

1. Boru, Dejene, et al. "Models for efficient data replication in cloud computing datacenters." Communications (ICC), 2015 IEEE International Conference on. IEEE, 2015.
2. Kumar, K. Ashwin, et al. "SWORD: workload-aware data placement and replica selection for cloud data management systems." The VLDB Journal 23.6 (2014): 845-870.
3. Sousa, Flávio RC, and Javam C. Machado. "Towards elastic multi-tenant database replication with quality of service." Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing. IEEE Computer Society, 2012.
4. Elango, P. "Data Replication for the Distributed Database using Decision Support Systems." International Journal of Computer Applications 69.3 (2013).
5. Abad, Cristina L., Yi Lu, and Roy H. Campbell. "DARE: Adaptive data replication for efficient cluster scheduling." Cluster Computing (CLUSTER), 2011 IEEE International Conference on. Ieee, 2011.
6. Ye, Yunqi, et al. "Cloud storage design based on hybrid of replication and data partitioning." Parallel and Distributed Systems (ICPADS), 2010 IEEE 16th International Conference on. IEEE, 2010.
7. Mansouri, Yaser, Adel Nadjaran Toosi, and Rajkumar Buyya. "Brokering algorithms for optimizing the availability and cost of cloud storage services." Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on. Vol. 1. IEEE, 2013.
8. Bonvin, Nicolas, Thanasis G. Papaioannou, and Karl Aberer. "A self-organized, fault-tolerant and scalable replication scheme for cloud storage." Proceedings of the 1st ACM symposium on Cloud computing. ACM, 2010.
9. Hussein, Mohamed-K., and Mohamed-H. Mousa. "A light-weight data replication for cloud data centers environment." International Journal of Engineering and Innovative Technology 1.6 (2012): 169-175.
10.] Bsoul, Mohammad, et al. "A threshold-based dynamic data replication strategy." The Journal of Supercomputing 60.3 (2012): 301-310.
11. López, Cindy, Rene Heinsen, and Eui-Nam Huh. "Improving Availability Applying Intelligent Replication in Federated Cloud Storage Based on Log Analysis." Proceedings of the 2017 International Conference on Machine Learning and Soft Computing. ACM, 2017.
12. Sun, Da-Wei, et al. "Modeling a dynamic data replication strategy to increase system availability in cloud computing environments." Journal of computer science and technology27.2 (2012): 256-272.
13. Hauglid, Jon Olav, Norvald H. Ryeng, and Kjetil Nørvåg. "DYFRAM: dynamic fragmentation and replica management in distributed database systems." Distributed and Parallel Databases 28.2-3 (2010): 157-185.
14. Curino, Carlo, et al. "Schism: a workload-driven approach to database replication and partitioning." Proceedings of the VLDB Endowment 3.1-2 (2010): 48-57.
15. Zhang, Quanlu, et al. "CHARM: A cost-efficient multi-cloud data hosting scheme with high availability." IEEE Transactions on Cloud computing 3.3 (2015): 372-386.
16. Nguyen, Tung, Anthony Cutway, and Weisong Shi. "Differentiated replication strategy in data centers." IFIP International Conference on Network and Parallel Computing. Springer, Berlin, Heidelberg, 2010.
17. Mao, Bo, Suzhen Wu, and Hong Jiang. "Exploiting workload characteristics and service diversity to improve the availability of cloud storage systems." IEEE Transactions on Parallel and Distributed Systems 27.7 (2016): 2010-2021.
18. Kamal, Joarder Mohammad Mustafa, Manzur Murshed, and Rajkumar Buyya. "Workload-Aware Incremental Repartitioning of Shared-Nothing Distributed Databases for Scalable Cloud Applications." Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing. IEEE Computer Society, 2014.
19. Nehme, Rimma, and Nicolas Bruno. "Automated partitioning design in parallel database systems." Proceedings of the 2011 ACM SIGMOD International Conference on Management of data. ACM, 2011.
20. Rafiq, Taha. Elasca: Workload-aware elastic scalability for partition based database systems. MS thesis. University of Waterloo, 2013.
21. Ye, Yunqi, et al. "Cloud storage design based on hybrid of replication and data partitioning." Parallel and Distributed Systems (ICPADS), 2010 IEEE 16th International Conference on. IEEE, 2010.
22. Toosi, Adel Nadjaran, Richard O. Sinnott, and Rajkumar Buyya. "Resource provisioning for data-intensive applications with deadline constraints on hybrid clouds using Aneka." Future Generation Computer Systems 79 (2018): 765-775.
23. Yu, Tao, et al. "Intelligent database placement in cloud environment." Web Services (ICWS), 2012 IEEE 19th International Conference on. IEEE, 2012.
24. Kumar, K. Ashwin, et al. "SWORD: workload-aware data placement and replica selection for cloud data management systems." The VLDB Journal 23.6 (2014): 845-870
25. Agarwal, Sharad, et al. "Volley: Automated data placement for geo-distributed cloud services." (2010).
26. Chen, Gang, et al. "Federation in cloud data management: Challenges and opportunities." IEEE Transactions on Knowledge and Data Engineering 26.7 (2014): 1670-1678.
27. Agrawal, Divyakant, et al. "Managing geo-replicated data in multi-datacenters." International Workshop on Databases in Networked Information Systems. Springer, Berlin, Heidelberg, 2013.
28. Wang, Haiping, Xiaofeng Meng, and Yunpeng Chai. "Efficient data distribution strategy for join query processing in the cloud." Proceedings of the third international workshop on Cloud data management. ACM, 2011.
29. Sakr, Sherif, et al. "A survey of large scale data management approaches in cloud environments." IEEE Communications Surveys & Tutorials 13.3 (2011): 311-336.
30. Zhao, Jing, Xiangmei Hu, and Xiaofeng Meng. "ESQP: an efficient SQL query processing for cloud data management." Proceedings of the second international workshop on Cloud data management. ACM, 2010
31. Hammes, Dayne, Hiram Medero, and Harrison Mitchell. "Comparison of NoSQL and SQL Databases in the Cloud." Proceedings of the Southern Association for Information Systems (SAIS), Macon, GA (2014): 21-22.
32. Baron, Joseph, and Sanjay Kotecha. "Storage options in the aws cloud." Amazon Web Services, Washington DC, Tech. Rep (2013).
33. Curino, Carlo, et al. "Relational cloud: A database-as-a-service for the cloud." (2011).
34. Zhang, Jinghui, et al. "Efficient location-aware data placement for data-intensive applications in geo-distributed scientific data centers." Tsinghua Science and Technology 21.5 (2016): 471-481.
35. Darshan Pandit et al. "Software Engineering Oriented Approach For Iot Applications: Need Of The Day " International Journal of Recent Technology and Engineering (IJRTE) Volume-7, Issue-6 (2019): 886-895

AUTHORS PROFILE

Mr. Nikhil S. Gajjam received his BE and ME degree in Computer Science & Engineering from Walchand Institute of Technology Solapur, and pursuing Ph.D degree in Computer Science & Engineering from K L University, Guntur. He is currently working as Assistant Professor in Walchand Institute of Technology, Solapur. He has published 5 research papers in various International Journals. His areas of interests are Cloud Computing, Distributed Databases

T. Gunasekhar is working as Associate Professor, K L University, Vijayawada. He has completed Ph. D in 2017. He has received his Bachelor of Technology and Master of Technology from Jawaharlal Nehru Technological University Anantapur in 2011 and 2013 respectively. He has completed PhD from K L University in computer science and engineering stream. His area of research is cloud computing, Computer networks and Network security.