Grenze International Journal of Computer Theory and Engineering , Feb 2016

A Cost Model for Query Execution in Cloud Computing based on both Shared-disk and Shared-nothing Architecture

Uttam Kumar Dash[1] and Chhanda Ray[2]
[1]Heritage Institute of Technology, Kolkata INDIA
uttamkumar.dash@heritageit.edu
[2]State Council of Educational Research & Training, Kolkata INDIA
raysasmal@rediffmail.com

***Abstract*—Cloud system is an emerging paradigm and a large pool of computing resources that makes use of existing technologies such as virtualization, service-orientation and grid computing. In cloud computing, on demand services are delivered to users depending on the service level agreements (SLA) between the service provider and the user. In order to provide scalable business services and cost allocation flexibility for customers so that they can choose their preferred services according to their budget, defining a cost-effective query execution strategy within cloud environment is utmost important. Towards this effort, a cost model for query execution at cloud computing based on both shared-disk and shared-nothing architecture is introduced in this paper. The complexity of the cost model has been analyzed considering different cost factors that are demanding for cloud computing. Finally, the cost model is illustrated with the help of an example.**

***Index Terms*— Cloud computing, Cost Model, Query Execution, shared-disk architecture, Shared-nothing Architecture.**

I. INTRODUCTION

Cloud computing is an emerging paradigm that will lead to the next generation of Internet and it provides optimized and efficient computing through enhanced collaboration, agility, scalability and, availability. Cloud computing is a large-scale of distributed computing paradigm that makes use of existing technologies such as virtualization, service-orientation, and grid computing [9]. It aims to share data, calculations, and services transparently among users of a massive grid. Software, platform and infrastructure as a service, are three main service delivery models for cloud computing.
As the popularity of cloud computing grows, the cloud service providers are facing ever increasing challenges because they have to maintain huge amount of heterogeneous data while providing efficient information retrieval. Thus, scalability and query efficiency are the key emphasis for cloud computing solutions. In order to provide scalability and efficient query execution in cloud databases, high availability of data, load balancing and data consistency are necessarily to be ensured. In this context, architecture of cloud computing plays an important role and it is the focus of attention of many researchers in recent times [2, 3, 4, 6, 8, 10]. The cloud computing model and the cloud delivery architecture models are explained in [2]. The

© Grenze Scientific Society, 2016

exemplary implementations of cloud services are also analyzed in this paper [2] and intriguing reports and facts about the current status of cloud computing and its future are shared. Paper [3] introduces an elaborated study of IaaS (Infrastructure as a service) component's security and finally a security model for IaaS to guide security assessment and enhancement in IaaS layer has been proposed. The concept and architecture of cloud computing as well as its security and privacy is comprehensively surveyed in [4]. The pros and cons for each cloud computing security model and architecture strategy are discussed in this paper [4]. In [6], a set of security protocols for ensuring the privacy and legal compliance of customer data in cloud computing architecture is presented.

In general, there are two different categories of database architecture for cloud, namely, shared-disk architecture and shared-nothing architecture [8]. In Shared-nothing database architecture, the data is partitioned in such a way that the database server exclusively executes a particular piece of data. Shared-disk database is analogous to a large trough of data where different database nodes can access or process any portion of the data. Data partitioning or shipping is not required in Shared-disk database architecture and three major features for cloud computing such as high availability, load balancing and data consistency can be easily implemented by this architecture. In [8], the characteristic, advantages and disadvantages of both database architectures for cloud computing have been illustrated. Paper [10] has made an attempt to use the basic concept of service-oriented architecture in system architecture of cloud system.

Efficient query processing and execution is a major concern in cloud computing [1, 5, 7, 11, 12]. In [1], the problem of distributed query optimization has been studied, the basic components of the distributed query optimizer have been focused and some future work highlighted based on some recent work that uses mobile agent technologies. A popular open source framework to store and retrieve large numbers of RDF triples in cloud computing is described and an algorithm to generate the best possible query plan for query execution based on a cost model is introduced in [5]. Paper [7] proposes a sophisticated query model to answer frequent queries which will cache statistics for frequent queries and uses dynamic programming to exploit the statistics. To provide updated history information, the history maintenance module is also proposed in [7]. In [11], the efficient available query optimization techniques for efficient retrieval of data to satisfy the customer needs in cloud computing is focused. A novel approach of query optimization for distributed database systems is presented in [12].

Due to different diverse architecture of cloud computing, defining a cost-effective query execution strategy for cloud computing is not trivial. Towards defining the cost-effective query execution strategy at cloud computing, the first step is to develop a cost model for query execution at cloud environment. Moreover, the cost model enables the cloud to provide cost allocation flexibility for customers so that they can choose their preferred services according to their budget. In this context, a cost model for query execution at cloud computing database is focused in this paper. To define the cost model for query execution, both the shared-nothing and shared-disk database architectures are considered in this work. As the cloud computing is a large scale paradigm of distributed computing so in this cost model the cost factors for distributed databases as well as the cost factors which are demanding for cloud computing are included. The complexity of the cost model has been analyzed and the model is illustrated with the help of an example.

The organization of this paper is as follows. Section II represents the cost model for query execution in cloud computing. The time complexity for the cost model is calculated in Section III. The cost model is illustrated with the help of an example in Section IV and Section V introduces a conclusion to the work.

II. The Cost Model

This section introduces a cost model for calculating the query processing cost in cloud environment. In this cost model, there are two major components, namely, the *external cost* and the *internal cost.* The external cost involves the *communication cost* and the *cloud cost* which are not directly associated with the data processing. The internal cost involves the *CPU cost* and the *I/O cost* that are directly related to the data processing. The communication cost reflects a transaction's behaviors on a cloud database in a network environment. Thus, the communication cost can be represented by using the following expression.

$$C_{cost} = CC_{data\ load} + CC_{trans\ proc} + CC_{trans\ failure} + CC_{site\ stat\ report} + CC_{net\ trans} \qquad (1)$$

where $CC_{data\ load}$ indicates the cost for data loading, $CC_{trans\ proc}$ represents the cost for data processing, $CC_{trans\ failure}$ indicates the cost for handling the link or communication failure, $CC_{site\ stat\ report}$ represents the cost for updating site status report, and $CC_{net\ trans}$ indicates the cost for network transmission. The data loading cost

$CC_{data\ load}$ is the joining cost of all fragmented tables where the desired data located and it involves two different components including the *cardinality of the fragmented tables*, say, CF, and *the size of the fragmented tables*, say, SF. The total number of frames to be generated for network transmission depends on the size of the fragmented tables. The data processing cost $CC_{trans\ proc}$ consists of two components, namely, *the data retrieval cost*, say, TR, and *the data update cost*, say, TU. Similarly, the link failure cost is measured by two different components such as *the repair cost for data retrieval*, RTR, and *the repair cost for data update*, RTU. To execute a transaction if one particular site under a selected path does not work in one particular instance then by applying traditional routing algorithm another optimally available site will be selected in place of the blocked site to continue the transaction. In case of transaction failure, the update made to one table should not be reflected in its replicas, thus, it is essential to update all associated replicas after the recovery of the failures. The cost component $CC_{site\ stat\ report}$ is incurred to update status report for each site to check whether the site is active or down before starting the execution of a transaction and thus it requires the accessing of routing table.

In order to measure the cloud cost for accessing data in cloud environment both for shared disk and shared nothing architecture, mainly two properties have been considered in this work. These are *load balancing* and *high availability*. In shared disk architecture, any site can access the entire data set, thus, data processing can be done at any site against a database request which results in more fluid load balancing. The driven factor behind the shared disks' ability to smoothly accommodate temporal and evolutionary changes in usage patterns is the fluidity in load balancing. Shared disk is Master-Master architecture while shared nothing is Master-Slave architecture. The total number of nodes/sites required to execute a query depends on *the volume of the data needed for the query* and *the maximum speed of the server* based on which partitioning will be done. Thus, the cost for allocating the master node in both shared disk and shared nothing cloud architecture is as follows.

$$CL_{cost} = \text{Data Volume} + \text{Max Speed of the Server} \qquad (2)$$ and

$$CL_{cost} = \text{Data Volume} + \text{Max Speed of the Server} + \text{tuning cost} + \text{the partitioning cost for slave nodes} \quad (3)$$

In case of shared nothing architecture, another two cost factors are associated including *the tuning cost* and *the slave nodes partitioning cost.* It is necessary to replace the master node with the slave node in case of master node failure and thus it involves a tuning cost. The slave nodes partitioning cost is measured in terms of the number of slave nodes that are needed to support the functionality of the master node.

The internal cost involves two different components including *the CPU cost* and *the I/O cost*. The CPU cost can be represented by using the following expression.

$$U_{cost} = UC_{card\_op} + UC_{int\_res} + UC_{tup\_op} + UC_{ind} \qquad (4)$$

where UC_{card_op} represents the *cardinality of the operand table*, UC_{int_res} indicates *the size of the intermediate result*, UC_{tup_op} represents *the tuple length of the operand table* and UC_{ind} indicates *the cost for Indexing.* The cost UC_{card_op} is measured by checking the mapping of one operand table with others where the mapping can be either one to one or one to many or many to many. The I/O cost for executing a query can be represented by using the following expression.

$$I_{cost} = IC_{acc\_dd} + IC_{up\_dd} + IC_{siz\_op} + IC_{siz\_res} \qquad (5)$$

where IC_{acc_dd} is the cost for accessing the Data Dictionary, IC_{up_dd} is the cost for update the data dictionary, IC_{siz_op} indicates the physical sizes of operand tables needed to execute a query, and IC_{siz_res} indicates the size of the resultant table. Therefore, the cost function for executing a query in cloud computing environment is as follows.

$$\begin{aligned} Tot_{cost} &= \text{External cost} + \text{Internal cost} \\ &= (\text{Communication cost} + \text{Cloud cost}) + (\text{CPU cost} + \text{I/O cost}) \\ &= C_{cost} + CL_{cost} + U_{cost} + I_{cost} \qquad (6) \end{aligned}$$

In the above cost model (6), each part is generated by considering the different parameters that are required to execute a database query in cloud computing where each parameter is again defined in terms of its subcomponents.

III. Complexity Analysis

The time complexity of the cost model is computed by adding the individual time complexity for each of the component.

A. Time Complexity for Communication Cost

The time complexity for the cardinality of the fragmented tables (CF) is $O(n^2)$ for many to many relation, $O(n)$ for one to many relation, and $O(1)$ for one to one relation, where n is the number of fragment tables. The time complexity for the size of the fragmented table (SF) is $O(r \times c)$, where r is the number of rows and c is the number of columns. The time complexity for the transaction retrieval (TR) and the transaction update (TU) are $O(n)$ and $O(n)$ respectively by considering one fragment at one site without any replica, where n is the number of fragmented tables. For executing a transaction, if a selected site under a selecting path does not work at one particular instance then by applying traditional routing algorithm another optimally available site may be selected to continue the execution of the transaction. Therefore, the time complexity of Repair for Transaction Retrieval (RTR) is $O(1)$ considering the time required to replace a blocked site by a new available site is constant. Whenever a failure occurs, it is necessary to update all replicas of a relation in order to maintain the data consistency. Thus, the complexity of Repair for Transaction Update (RTU) is $O(m)$, where m represents the total number of replicas of a relation. To update the status report for each site whether it is UP or DOWN, accessing to the routing table is needed which is constant and thus the time complexity for updating site status report is $O(1)$. The time complexity for the network transmission is $O(h)$, where h is the number of intermediate site between the source and the destination. Therefore, the time complexity for communication cost in worst case is

$$O(n^2) \times O(r \times c) + (O(n) + O(n)) + (O(1) + O(m)) + O(1) + O(h) \approx O(n^4).$$

B. Time Complexity for Cloud Cost

The complexity of the Cloud Cost may vary if the volume of the data increases or decreases and the server speed is either scale up or scale out. The complexity for data volume is $O(i \times r)$ where i is the number of interconnected operand tables, r is the number of rows to be processed per unit time. The tuning cost can be measured by $O(r1 \times c1)$, where r1 is the number of rows and c1 is the number of columns of the last updated resultant table at Slave node. The partitioning cost of Slave nodes is measured by the number of slave nodes required to support the Master node and its complexity is $O(i \times r \times c)$ where i is the mapping cardinality of the operand table, r is the number of rows and c is the number of columns in the resultant table. Therefore, the time complexity for cloud cost with shared disk architecture is $O(i \times r) + O(r) \approx O(n^2)$ and the time complexity for cloud cost with shared nothing architecture is

$$O(i \times r) + O(r) + O(r1 \times c1) + O(i \times r \times c) \approx O(n^3).$$

C. Time Complexity for CPU Cost

The cardinality of the operand table depends on the mapping of one operand table with others. The time complexity of the cardinality is $O(n^2)$ for many to many relation, $O(n)$ for one to many relation, and $O(1)$ for one to one relation, where n represents the total number of operand tables. The size of the intermediate result is represented by the total number of rows of the intermediate resultant table and thus the complexity is $O(r)$. The tuple length of the operand table is determined by the number of tuples and its size where the tuple's size depends on the total number of columns associated with the tuple. Thus, the time complexity of the tuple length of the operand table is $O(t \times c)$, where t indicates the total number of tuples selected to execute a particular query. The execution of a query may need accessing to multi-level indexing for retrieving desired data and at each level there is at least an index table. The time complexity for accessing index tables is $O(i \times t)$, where i represents the total number of indexing layers and t indicates the total number of index tables at each layer. Hence, it is assumed that accessing time to an index table is constant. Therefore, the time complexity for CPU cost is

$$O(n^2) + O(r) + O(t \times c) + O(i \times t) \approx O(n^2).$$

D. Time Complexity for I/O Cost

The time complexity for accessing the data dictionary and for update the data dictionary is constant, that is, equivalent to $O(1)$. The time complexity for accessing operand tables is $O(i \times r \times c)$, where i is the total number of operand tables, r is the number of rows and c is the number of columns at each operand table. The complexity for accessing resultant tables is also same as operand tables. Therefore, the time complexity for I/O cost is

$$O(1) + O(1) + O(i \times r \times c) + O(i \times r \times c) \approx O(n^3).$$

Based on the above cost model, the time complexity for executing a query in the cloud environment with shared disk or shared nothing architecture is $O(n^4)$.

IV. An Example

Let us assume that a query "*Select the names of all employees who are working for the project P1*" has been initiated in the cloud environment. This query involves two different relations *Employee (Eid, Ename, Designation)* and *Project (Pno, Eno, Duration*) respectively where *Eid* is the primary key of the *Employee* relation and *(Pno, Eno*) is the primary key of the *Project* relation. Hence, *Eno* of the *Project* relation is a foreign key of the *Employee* relation referencing the attribute *Eid*. It is assumed that the Employee relation is horizontally fragmented without any replication and stored in two different locations, say, node1 and node2 respectively. It is also assumed that the Project relation is derived fragmented without any replication and stored in two different locations, say, node3 and node4 respectively. By using relational algebra, the above query is represented in the following.

$$\pi_{Ename} (\pi_{Eid,Ename} (Employee) \bowtie_{Employee.Eid = Project.Eno} \pi_{Eno}(\sigma_{Pno='P1'}(Project)))$$

In order to calculate the query execution cost for the above query, it is considered that there are 400 tuples in Employee relation and 1000 tuples in Project relation that are uniformly distributed across node1, node2 and node3, node4 respectively. There are indexes on primary keys at all nodes and direct tuple access is possible on local sites. It is also assumed that all nodes can directly communicate with each other. The cost for retrieving tuples of Project relation at node3 and node4 is 2 x (10 x 1) = 20 units assuming the cost for accessing a tuple is 1 unit. The cost for transferring the resultant tuples of Project relation to the cloud is 2 x (10 x 10) = 200 units considering the cost for transferring a tuple from one location to another in the cloud is 10 units. Similarly, the cost for retrieving tuples of Employee relation at node1 and node2 is 2 x (10 x 1) = 20 units and the cost for transferring the resultant tuples of Employee relation to the cloud is 2 x (10 x 10) = 200 units. Therefore, the total cost for executing the above query at cloud environment is 440 units.

V. Conclusion

This paper introduces a cost model for query execution at cloud computing based on both shared-disk and shared-nothing architecture. The time complexity of the cost model is also computed in this work consider different parameters that are applicable for cloud computing. However, the proposed cost model mainly concentrate on two properties of cloud databases, namely, high availability and load balancing. In future scope, a cost-effective optimized query execution strategy can be developed based on this cost model for efficient query execution in cloud environment.

References

[1] Aljanaby, E. Abuelrub, and M. Odeh "A Survey of Distributed Query Optimization", Proceedings of the International Arab Journal of Information Technology, Vol. 2, No. 1, January 2005.

[2] I. Bojanova, A. Samba, "Analysis of Cloud Computing Delivery Architecture Models", Proceedings of the International Conference on Advanced Information Networking and Applications, 2011, pp 453-458.

[3] Wesam Dawoud, Ibrahim Takouna, Christoph Meinel, "Infrastructure as a Service Security: Challenges and Solutions", Proceedings of the IEEE 2010.

[4] F. Hu, M. Qiu, J. Li, T. Grant, D. Tylor, S. Mccaleb, L. Butler, R. Hamner, "A Review on Cloud Computing : Design Challenges in Architecture and Security", Proceedings of the Journal of Computing and Information Technology, Vol. 1, 2011, pp 25-55.

[5] M.F. Husain, L. Khan, M. Kantarcioglu, B. Thuraisingham, "Data Intensive Query Processing for Large RDF Graphs using Cloud Computing Tools", Proceedings of IEEE 3rd International Conference on Cloud Computing, 2010.

[6] Wassim Itani, Ayman Kayssi, Ali Chehab, “Privacy as a Service: Privacy-Aware Data Storage and Processing in Cloud Computing Architectures”, Proceedings of the 8th IEEE International Conference on Dependable, Autonomic and Secure Computing, 2009, pp 711-716.

[7] S. J. Jebamani, K. Padmaveni, “Optimal Query Processing in Semantic Web using Cloud Computing”, Proceedings of the International Journal of Science and Research (online), Vol. 2, Issue 3, March 2013, pp 185-189.

[8] S. Lee, “Shared-Nothing vs. Shared-Disk Cloud Database Architecture”, Proceedings of the International Journal of Energy, Information and Communications, Vol. 2, Issue 4, November, 2011.

[9] Lijun Mei, W.K. Chan, T.H. Tse, “A Tale of Clouds: Paradigm Comparisons and Some Thoughts on Research Issues”, Proceedings of the IEEE Asia-Pacific Services Computing Conference, 2008, pp 464-469.

[10] Manish Pokharel, YoungHyun Yoon, Jong Sou Park, “Cloud Computing in System Architecture”, Proceedings of the IEEE 2009.

[11] N. Samatha, K. Vijay Chandu, P. Raja Sekhar Reddy, “Query Optimization Issues for Data Retrieval in Cloud Computing”, Proceedings of the International Journal of Computational Engineering Research (ijceronline.com), Vol. 2, Issue 5, pp 1361-1364.

[12] D. Sukheja and U. K. Singh “A Novel Approach of Query Optimization for Distributed Database Systems”, International Journal of Computer Science Issues (IJCSI), Vol. 8, Issue 4, No 1, July 2011.