Volume 3, Issue 5, May 2013 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

An Optimistic Approach for Query Construction and Execution in Cloud Computing Environment

P.Ravinder Rao
Department of CSE
Anurag Group of Institution,Hyderabad
India.

S.V.Sridhar
Department of CSE
HITS,Hyderabad
India.

V.RamaKrishna
Department of CSE
Anurag Group of Institution,Hyderabad
India.

Abstract: As the demand for cheaper commodity machines and large-scale databases rise, organizations have turned to cloud computing, or simply "the cloud," as the solution. Cloud data storage redefines the issues targeted on customer's out-sourced data (data that is not stored/retrieved from the customers own servers). In this work we observed that, from a customer's point of view, the data need to be accessed with in no time the user given the request ,Even though the data is stored on cloud severs the effective query optimizations may not be defined to access the data in an efficient way. This paper proposes the efficient available Query optimization techniques for efficient retrieval of data to satisfy the customer needs.

Keywords:* *Cloud Storage, Cloud Computing, Query Processing, Query Optimization,

I. Introduction

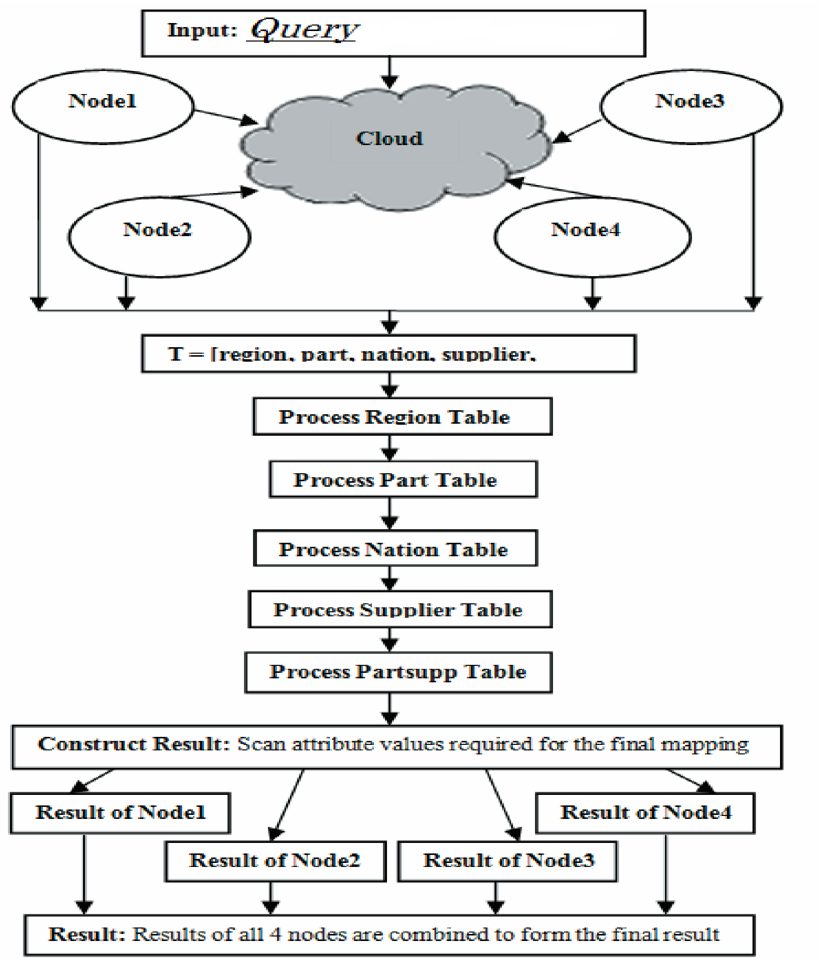
The industrial information technology towards a subscription based or pay-per-use service business model known as *cloud computing*. This paradigm provides users with a long list of advantages, such as provision computing capabilities; broad, heterogeneous network access; resource pooling and rapid elasticity with measured services .Huge amounts of data being retrieved from geographically distributed data sources, and non-localized data-handling requirements, creates such a change in technological as well as business model. One of the prominent services offered in *cloud computing* is the *cloud data storage*, in which, subscribers do not have to store their own data on their servers, where instead their data will be stored on the cloud service provider's servers. In cloud computing, subscribers have to pay the providers for this storage service. This service does not only provides flexibility and scalability data storage, it also provides customers with the benefit of paying only for the amount of data they needs to store for a particular period of time, without any concerns of efficient storage mechanisms and maintainability issues with large amounts of data storage[3].

In addition to these benefits, customers can easily access their data from any geographical region where the Cloud Service Provider's network or Internet can be accessed [1]. An example of the cloud computing is shown in Fig. 1. Since *cloud service providers* (*SP*) are separate market entities, data integrity and privacy and retrieval are the most critical issues that need to be addressed in *cloud computing*. Even though the cloud service providers have standard regulations and powerful infrastructure to ensure Customer's data privacy, data retrieval and provide a better availability [5], the reports of privacy breach and service outage have been apparent in last few years. Cloud [11]provides a flexible environment where user can load data, execute queries and scale resources on demand. However, cloud has its own challenges.

To reduce the inter-node communication during the execution of query, tables are horizontally partitioned on join attribute and then related partitions are stored on the same physical system. In cloud environment it is not possible to ensure that these related partitions are always stored on the same physical system. As the resources are scaled up, the number of nodes involved increases, resulting in the increased inter-node communication. This becomes critical when we have huge data stored across a large number of nodes In this work we observed that, from a customer's point of view, relying upon data retrieving which he needs by performing an effective query optimization In addition, providing reliability ,availability are crucial and equally important to query optimization. Query Optimization can be achieved by implementing the optimization techniques for effective retrieval of data.

II. Cloud storage system query processing:

Transform a high level query (of relation al calculus/SQL) on a stored database (i.e., a set of global relations) into an **equivalent** and **efficient** lower-level query (of relational algebra) on relation fragments.



Flow of Query Processing[11]

• Cloud storage system query processing is more complex
– Fragmentation/replication of relations
– Additional communication costs
– Parallel execution
Example: Transformation of an SQL-query into an RA-query.
Relations: EMP(ENO, ENAME, TITLE), ASG(ENO,PNO,RESP,DUR)
Query: *Find the names of employees who are managing a project*?
– High level query
SELECT ENAME **FROM** EMP,ASG **WHERE** EMP.ENO = ASG.ENO **AND** DUR > 37
– Two possible transformations of the query are:
_ Expression 1:
_ENAME (_DUR>37∧ EMP.ENO=ASG.ENO(EMP × ASG))
_ Expression 2: _ENAME (EMP ⋊⋉ENO (_DUR>37(ASG)))
– Expression 2 avoids the expensive and large intermediate Cartesian product, and therefore typically is better.

Select, Project (without duplicate elimination) O(n)	Project (with duplicate elimination) O(n log n)
Group ,Join, Semi-join & Division, Set Operators O(n log n)	Cartesian Product O(n2)

oud Storage System Query Optimization:
Query optimization is a crucial and difficult part of the overall query processing[2]
• Objective of query optimization is to **minimize** the following cost function: I/O cost + CPU cost + communication cost
• Two different scenarios are considered:
– Wide area networks : Communication cost dominates, low bandwidth, low speed, high protocol overhead
Most algorithms ignore all other cost components
– Local area networks: Communication cost not that dominant

Total cost function should be considered

Ordering of the operators of relational algebra is crucial for efficient query processing

• Rule of thumb: move expensive operators at the end of query processing

• Cost of RA operations:

Operation Complexity

Query Optimization Issues[2]

Several issues have to be considered in query optimization

• Types of query optimizers

– Wrt the search techniques (exhaustive search, heuristics)

– Wrt the time when the query is optimized (static, dynamic)

• Statistics

• Decision sites

• Network topology

• Use of semi joins

Types of Query Optimizers wrt Search Techniques[2]

– Exhaustive search

Cost-based

Optimal

Combinatorial complexity in the number of relations

– Heuristics

Not optimal

Regroups common sub-expressions

Performs selection, projection first

Replaces a join by a series of semijoins

Reorders operations to reduce intermediate relation size

Optimizes individual operations

Create a meta-optimizer designed to pick cloud instances that minimize cost. Further, we conceive the following specific optimizations:

· IO

· Database files - Accessing rows in a table.

· Index files - Add, read, write, remove a new key from the index file.

· Transaction logs - These files include info on the transactions (e.g. updates) performed on the RDBMS. They are used to recover the RDBMS after a failure of the server.

· Temporary tables - Temporary storage of the intermediate query results/table. This occurs when a query requires access to a large number of rows (or a sub-query created large intermediate tables) that cannot be stored in memory.

· CPU usage - Understanding the percentage of CPU usage for a given workload, and how it affects the query response latency.

· Memory usage - Monitoring usage of the buffer pool and other caches and the distribution across different types of data.

· Disk usage - What is the required disk space for our workload? This includes the space
required by indexes, materialized views, intermediate query results, or temp tables.

· Compression - What compression levels minimize storage on the cloud and simultaneously maximize efficacy for IO.

· Query response latency - Determining the average response latency of our workload for various query types

Buffer Pool and Indexing:

1.Naive - without IO optimizations. This test is designed to see if the location of data, log and temporary directories on local storage will require more IO than mounted volume. In addition, it tests to see if this reduced IO is more significant than the cost of the additional overhead of a persistent storage medium.

2. Augmented Buffer Pool Size - Tests to see if an increased buffer pool size decreases total read/writes across
transactions. According to Databases documentation, the buffer pool should be set to 75% of main memory, along with a log file size of 25% of the buffer pool size. M1.Small and C1.Medium have 1.7GB of memory, so a buffer pool size of 1.275GB and a log file size of 318MB were used.

3. Data Indexing-[10] - Indexing data is a rudimentary procedure for improving the speed of operations on a database table by creating a copy of part of a table. Though creating and storing the index requires more space and IO, the hope is to reduce total IO over the life of an instance by providing faster lookups, thereby reducing
the IO count of transactions.

4. Data Indexing with an Augmented Buffer: Pool Size - Combining steps 2 and 3.

III. Types of Query Optimizers wrt Optimization Timing:

Static: Query is optimized prior to the execution As a consequence it is difficult to estimate the size of the intermediate results. Typically amortizes over many executions

Dynamic: Optimization is done at run time Provides exact information on the intermediate relation sizes ,Have to re-optimize for multiple executions
Hybrid: First, the query is compiled using a static algorithm Then, if the error in estimate sizes greater than threshold, the query is re-optimized at run time.
Statistics: Relation/fragments,**,** Cardinality**,** Size of a tuple **,** Fraction of tuples participating in a join with another relation/fragment
Attribute: Cardinality of domain, Actual number of distinct values, Distribution of attribute values (e.g., histograms) Common assumptions: Independence between different attribute values, Uniform distribution of attribute values within their domain

Decision sites

Centralized: Single site determines the ”best” schedule Simple, Knowledge about the entire distributed database is needed
Distributed: Cooperation among sites to determine the schedule, Only local information is needed Cooperation comes with an overhead cost
Hybrid: One site determines the global schedule, Each site optimizes the local sub-queries

Use of Semi joins[2]:

Reduce the size of the join operands by first computing semijoins Particularly relevant when the main cost is the communication cost Improves the processing of distributed join operations by reducing the size of data exchange between sites However, the number of messages as well as local processing time is increased Query processing transforms a high level query (relational calculus) into an equivalent lower level query (relational algebra). The main difficulty is to achieve the efficiency in the transformation

• Query optimization aims to minimize the cost function: I/O cost + CPU cost + communication cost
• Query optimizers vary by search type (exhaustive search, heuristics) and by type of the algorithm (dynamic, static, hybrid). Different statistics are collected to support the query optimization process
• Query optimizers vary by decision sites (centralized, distributed, hybrid)
• Query processing is done in the following sequence: query decomposition data localization global optimization local optimization.

IV. Materialized Views and Compression:

This optimization was to determine the efficacy of advanced database techniques such as compression and materialized views. Materialized views help pre-compute the most common read intensive queries. Well-designed materialized views save enough in IO to amortize the cost of maintaining a view. Consequently this is most beneficial in a read-intensive environment. In order to determine whether a materialized view is a good choice for a given workload we calculate a break-even point for the projected cost of our query. Materialized views may be useful in cases where the select statement and predicates only access a small percentage of the columns in the original tables, saving IO.
Compression[10] **-** Compression helps save IO at the cost of more latency. This tradeoff benefits the clouds, but not traditional optimizers where latency is the primary concern. More intuitively, it saves space and thus money on the cloud.

V. Conclusion

In this paper, we proposed a various issues related to the Query Optimization in cloud computing, which seeks to provide each customer with better data retrieval from cloud data storage.

Acknowledgement

We are very much thankful to our Principal who encouraged us and given valuable suggestions in carrying out this paper

Reference

[1] Amazon.com, “Amazon s3 availablity event: July 20, 2008”, Online at http://status.aws.amazon.com/s3-20080720.html, 2008.
[2] M. Tamer Oezsu, Patrick Valduriez ``Principles of Distributed Database Systems, Second Edition'' Prentice Hall, ISBN 0-13-659707-6, 1999
[3] R. Gellman, “Privacy in the clouds: Risks to privacy and confidentiality from cloud computing”, Prepared for *the World Privacy Forum*, online at http://www.worldprivacyforum.org/pdf/WPF Cloud Privacy Report.pdf,Feb 2009.
[4] W. Itani, A. Kayssi, A. Chehab, “Privacy as a Service: Privacy-Aware Data Storage and Processing in Cloud Computing Architectures,” *Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing*, Dec 2009
[5] M. Jensen, J. Schwenk, N. Gruschka, L.L. Iacono, “On Technical Security Issues in Cloud Computing”, *IEEEInternational Conference on Cloud Computing, (CLOUD II 2009)*, Banglore, India, September 2009,109-116
[6] P. F. Oliveira, L. Lima, T. T. V. Vinhoza, J. Barros, M. M´edard,“Trusted storage over untrusted networks”, *IEEE GLOBECOM 2010*, Miami, FL. USA.

[7] S. H. Shin, K. Kobara, "Towards secure cloud storage", *Demo forCloudCom2010,* Dec 2010.
[8] Stefano ceri Giuseppe pelagati "Distributed Databases- Principles and systems"Tata MC Graw Hill 2008
[9] J . Du, W. Wei, X. Gu, T. Yu, "RunTest: assuring integrity of dataflow processing in cloud computing infrastructures", In Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security (ASIACCS '10), ACM, New York, NY, USA, 293-304
[10] Adam Conrad –"Database Economic Cost Optimization for Cloud Computing"
[11] Swathi Kurunji Tingjian Ge Benyuan Liu Cindy X. Chen "Communication Cost Optimization for Cloud Data Warehouse Queries"- 2012 IEEE 4th International Conference on Cloud Computing Technology and Science