The International Journal Of Engineering And Science (IJES)
|| Volume || 3 || Issue || 6 || Pages || 07-11 || 2014 ||
ISSN (e): 2319 – 1813 ISSN (p): 2319 – 1805

Design of File System Architecture with Cluster Formation Along With Mount Table: A Review

[1,]Sheetu Sharma, [2,]Vikas Gupta
[1,] *Computer Science & Engg., AIET, Faridkot*
[2,] *Assistant Professor, Department of ECE, AIET, Faridkot*

----------ABSTRACT ----------

A keyword cluster is used in the hash table to find corresponding cluster having all related files. And then all the files related to that keyword are displayed. Then, upon selection of one file, the whole content belonging to that file is displayed. This research is focused on solutions to get or retrieve data from large data source in less time and at less computation cost. Query optimization filters important words as keywords and then these are passed to hash table, i.e. indexing technique is used which finds corresponding value. In a cluster, same type of data is presented in different types of file formats .There is currently considerable enthusiasm around the MapReduce (MR) paradigm for large-scale Data analysis [17]. Although the basic control flow of this framework has existed in parallel SQL database management systems (DBMS) for over 20 years, some have called MR a dramatically new computing model [8, 17]. In this paper, description & comparison of both paradigms on an open source version of MR as well as on parallel DBMS have been carried out. The observed performance of these DBMSs was strikingly better. This gives a boost to the idea of implementing clustering in Query Optimization.

KEYWORDS: *MR paradigm, Query Optimization.*

Date of Submission: 12 June 2014 Date of Publication: 20 June 2014

I. INTRODUCTION

Recently "Cluster Computing" has brought about the revolution; this paradigm entails harnessing large numbers of (low-end) processors working in parallel to solve a computing problem. In effect, this suggests constructing a data center by lining up a large number of low-end servers instead of deploying a smaller set of high-end servers. With this increasing interest in clusters, there has become a proliferation of tools for programming them. One of the earliest and best known search tools are in Map Reduce (MR) [8]. Map Reduce is attractive because it provides a simple model through which users can express relatively sophisticated distributed programs, leading to a significant contribution to the educational community. For example, IBM and Google have announced plans to make a 1000 processor Map Reduce cluster available to teach distributed programming to the students. Research has been undertaken with the goal to understand the differences between the Map Reduce approach to performing large-scale data analysis and the approach taken by parallel database systems. The two classes of systems make different choices in several key areas. For example, all DBMSs require that data conform to a well-defined schema, whereas MR permits data to be in any arbitrary format. Other differences also include how each system provides indexing and compression optimizations, programming model.

II. OBJECTIVES OF RESEARCH WORK

The main work presented in the system is to define file system architecture with query optimization. The research work is divided in terms of some research objectives given as under.

[1] Design of File System architecture along with cluster formation and Mount table specification.
[2] Implementation of keyword based clustering.
[3] Generation of separate mounts table for each cluster.
[4] Implementation of the client side, respective to query fetching, analysis and getting results based on analysis.
[5] Fetching data from the query optimization process.

III. AVAILABLE APPROACHES TO LARGE SCALE DATA ANALYSIS

3.1 MapReduce

One of the attractive qualities about the MapReduce programming model is its simplicity: an MR program consists of two functions, called Map and Reduce**,** which are written by a user to process key/value data pairs. The input data set is stored in a collection of partitions on a distributed file system deployed on each node in the cluster. The program is then injected into a distributed processing framework and executed in a manner to be described.

3.2 Parallel DBMSs

Database systems capable of running on clusters of shared nothing nodes have existed since the late 1980s.A Shared Nothing Node is data architecture for distributed data storage in a clustered environment. Data is partitioned in some manner and spread across a set of machines with each machine having sole access, and hence sole responsibility, of the data it holds. In this architecture, each node is independent and self-sufficient, and there is no single point of contention across the system. More specifically, none of the nodes share memory or disk storage. Shared nothing is popular for web development because of its scalability. As Google has demonstrated, a pure SN system can scale almost infinitely simply by adding nodes in the form of inexpensive computers, since there is no single bottleneck to slow the system down. Google calls this Sharding. An SN system typically partitions its data among many nodes on different databases or may require every node to maintain its own copy of the application's data, using some kind of coordination protocol. This is often referred to as Database Sharding. These systems support standard relational tables and SQL, and thus the fact that the data stored on multiple machines is transparent to the end-user. Many of these systems are built on the pioneering research from the Gamma [10] and Grace [11] parallel DBMS projects. The two key aspects that enable parallel execution are that (1) most (or even all) tables are partitioned over the nodes in a cluster and that (2) the system uses an optimizer that translates SQL commands into a query plan whose execution is divided amongst multiple nodes. Because programmers only need to specify their goal in a high level language, they are not burdened by the underlying storage details, such as indexing options and join strategies.

Cluster-based solutions are widely accepted in current data warehouse systems as centralized servers do not provide scalable performance in the face of data explosion. In cluster-based systems, performances are improved by exploiting the parallelism. MapReduce is a new framework that simplifies the development of parallel applications [15].

3.3 Schema Support

Parallel DBMSs require data to fit into the relational paradigm of rows and columns. In contrast, the MR model does not require that data files adhere to a schema defined using the relational data model. That is, the MR programmer is free to structure their data in any manner or even to have no structure at all.

When no sharing is anticipated, the MR paradigm is quite flexible. If sharing is needed, however, then we argue that it is advantageous for the programmer to use a data description language and factor schema definitions and integrity constraints out of application programs. This information should be installed in common system catalogs accessible to the appropriate users and applications.

3.4 Indexing

All modern DBMSs use hashes to accelerate access to data. If one is looking for a subset of records (e.g., employees with a salary greater than $100,000), then using a proper index reduces the scope of the search dramatically. Most database systems also support multiple indexes per table. Thus, the query optimizer can decide which index to use for each query or whether to simply perform a brute-force. In computer science, brute-force search or exhaustive search, also known as generate and test, is a very general problem-solving technique that consists of systematically enumerating all possible candidates for the solution and checking whether each candidate satisfies the problem's statement. Advantages of brute force algorithm

- Wide applicability
- Simplicity
- Yields reasonable algorithms for some important problems (e.g., matrix multiplication, sorting, searching, string matching).

In Brute-Force Algorithm:

Step1. Align pattern at beginning of text.
Step2. Moving from left to right, compare each character of the pattern to the corresponding character in text until all characters are found to match (successful search) or a mismatch is detected.
Step3. While the pattern is not found and the text is not yet exhausted, realign pattern one position to the right and repeat Step 2.

Informal evidence from the MR community suggests that there is widespread sharing of MR code fragments to do common tasks, such as joining data sets. To alleviate the burden of having to re-implement repetitive tasks, the MR community is migrating high level languages on top of the current interface to move such functionality into the run time. Pig [15] and Hive [2] are two notable projects in this direction.

3.5 Data Distribution

The conventional wisdom for large-scale databases is to always send the computation to the data, rather than the other way around. In other words, one should send a small program over the network to a node, rather than importing a large amount of data from the node. Parallel DBMSs use the knowledge of data distribution and location to their advantage: a parallel query optimizer strives to balance computational workloads while minimizing the amount of data being transmitted over the network connecting the nodes of the cluster. The modern DBMS would rewrite the second query such that the view definition is substituted for the Keywords table in the FROM clause. Then, the optimizer can push the WHERE clause in the query down so that it is applied to the Documents table before the COUNT is computed, thus substantially reducing computation. If the documents are spread across multiple nodes, then this filter can be applied to each node before documents belonging to the same site are grouped together, generating much less network I/O.

3.6 Hadoop

The Hadoop system is the most popular open-source implementation of the MapReduce framework, under development by Yahoo! and the Apache Software Foundation [1]. Unlike the Google implementation of the original MR framework written in C++, the core Hadoop system is written entirely in Java. Hadoop uses a central job tracker and a "master" HDFS daemon to coordinate node activities [1].

IV. RESULTS

A.Time Consumed:

Webmining Cluster (Files)	Modified Hadoop (Time Consumed)	Hadoop (Time Consumed)
Rs	858millesecs	2730millesecs
Srs	905millesecs	2762millesecs
Advantages of webmining	796millesecs	1170millesecs
Disadvantages of webmining	795millesecs	2995millesecs
Webmining is datamining	390millesecs	1404millisecs

Table (1): Time consumed by Files in Modified Hadoop Vs Hadoop

B. Computation Cost:

(a)

Computation Cost	Modified Hadoop	Hadoop
Processor	2.20GHz	4.30GHz
RAM	2 GB	3 GB

Table (2): Resources are used in Modified Hadoop Vs Hadoop

(b)

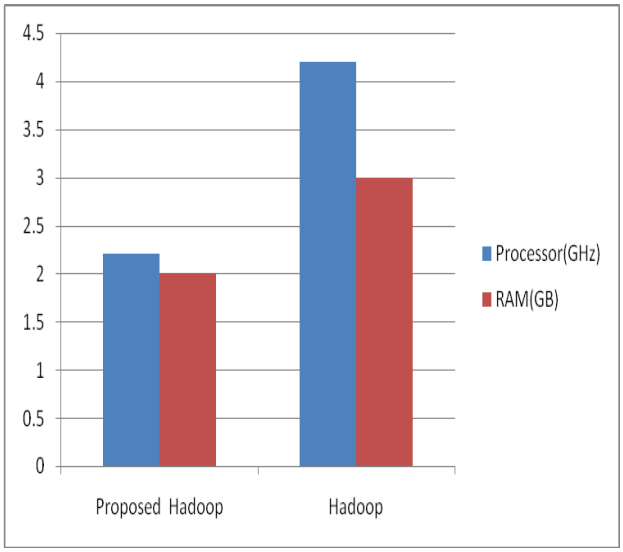

Figure (1): Graphical Representation of Resources used in Modified Hadoop Vs Hadoop

V. CONCLUSION

There are a number of interesting conclusions from the results of various conclusions from the results of various situations presented in this paper. First, at the scale of the experiments conducted, both parallel database systems displayed a significant performance over Hadoop MR in executing a variety of data intensive analysis benchmarks. The dual of these numbers is that a parallel database system that provides the same response time with far fewer processors will certainly use far less energy; the MapReduce model on multi-thousand node clusters is a brute force solution that wastes vast amounts of energy. Because of a number of technologies developed over the past 25 years, including (1) B-tree indices to speed the execution of selection operations, (2) novel storage mechanisms (e.g., column orientation), (3) aggressive compression techniques with the ability to operate directly on compressed data, and (4) sophisticated parallel algorithms for querying large amounts of relational data. [18] There is a lot to learn from both kinds of systems. Most importantly is that higher level interfaces, such as Pig [15], Hive [2], are being put on top of the MR foundation, and a number of tools similar in spirit but more expressive than MR are being developed, such as Dryad [13] and Scope [5]. This will make complex tasks easier to code in MR-style systems and remove one of the big advantages of SQL engines, namely that they take much less code on the tasks in our benchmark. For parallel databases, it is believed that both commercial and open-source systems will dramatically improve the parallelization of user-defined functions. In consideration of query optimization, clustering approach seems extremely beneficial to this task. Recent results with various forms of clustering strategies and implementation schemes as cited in this review paper make the idea of clustering more practical and useful.

VI. FUTURE SCOPE

When work is performed for particular enterprises; it contains a vast collection of files over the system. In such case the management of these files and handling the file system query is itself is a challenging task.

REFERENCES

[1] Hadoop. http://hadoop.apache.org/.
[2] Hive. http://hadoop.apache.org/hive/.
[3] Vertica. http://www.vertica.com/.
[4] konstantin Shvachko, Hairong Kuang, Sanjay Radia, Robert Chansler Sunnyvale," The Hadoop Distributed File System" in 2010.
[5] R. Chaiken, B. Jenkins, P.-A. Larson, B. Ramsey, D. Shakib,S. Weaver, and J. Zhou. Scope: easy and efficient parallel processing of massive data sets. Proc. VLDB Endow.,1(2):1265–1276, 2008.
[6] Ms. M.C. Nikose," Query Optimization in Object Oriented Databases through Detecting Independent Subqueries", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 2, February 2012.
[7] Haroun Rababaah "Distributed Databases Fundamentals and Research",Advanced Database–B561.Spring2005.Dr. H.Hakimzadeh Department of Computer and Information Sciences Indiana University South Bend in 2005.
[8] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In OSDI '04, pages 10–10,2004.
[9] Surajit Chaudhuri, "An Overview of Query Optimization in Relational Systems" in 2012

[10] D. J. DeWitt, R. H. Gerber, G. Graefe, M. L. Heytens, K. B.Kumar, and M. Muralikrishna. GAMMA - A High Performance Dataflow Database Machine. In VLDB ’86, pages 228–237, 1986.
[11] S. Fushimi, M. Kitsuregawa, and H. Tanaka. An Overview of the System Software of A Parallel Relational Database Machine. In VLDB ’86, pages 209–219, 1986.
[12] VivekShrivastava,”An Idea of Extraction of Information Using Query Optimization and Rank Query”, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 4, February 2012.
[13] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad:Distributed Data-parallel Programs from Sequential Building Blocks. In EuroSys ’07, pages 59–72, 2007.
[14] Queryoptimization,http.//queryoptimization.org.
[15] C. Olston, B. Reed, U. Srivastavas, R. Kumar, and A. Tomkins. Pig Latin: A Not-So-Foreign Language for Data Processing. In SIGMOD ’08, pages 1099–1110, 2008.
[16] Surajit Chaudhuri, “An Overview of Query Optimization in Relational Systems” in 2012.
[17] D. A. Patterson. Technical Perspective: The Data Center is the Computer. Commun. ACM, 51(1):105–105, 2008.
[18] Sai Wu, Feng Li, Sharad Mehrotra, Beng Chin Ooi. Query Optimization for Massively Parallel Data Processing.
[19] M. Stonebraker and J. Hellerstein. What Goes Around Comes Around. In Readings in Database Systems, pages2–41. The MIT Press, 4th edition, 2005.