International Journal of Computer, Information Systems and Knowledge Management
Vol.1, No.1, pp.1-7, March 2014
Published by British Research Institute UK (www.gbjournals.org)

APPLICATION OF HADOOP MAPREDUCE TECHNIQUE TO VIRTUAL DATABASE SYSTEM DESIGN

Neha Tiwari
Rahul Pandita
Nisha Chhatwani
Divyakalpa Patil
Prof. N.B.Kadu
PREC, Loni, India.

ABSTRACT- *Today in the world of cloud and grid computing integration of data from heterogeneous databases is inevitable. Virtual Database Technology (VDB) is one of the effective solutions for integration of data from heterogeneous sources. This will become complex when size of the database is very large. MapReduce is a new framework specifically designed for processing huge datasets on distributed sources. Apache's Hadoop is an implementation of MapReduce. Currently Hadoop has been applied successfully for file based datasets. This paper proposes to utilize the parallel and distributed processing capability of Hadoop MapReduce for handling heterogeneous query execution on large datasets. So, Virtual Database Engine built on top of this will result in effective high performance distributed data integration.*

KEYWORDS- Database integration, Virtual Database Technology, Hadoop MapReduce, heterogeneous databases, query optimization.

INTRODUCTION

Today for distributed systems like Cloud and Grid data integration from heterogeneous data sources is unavoidable. The data is made available in several structured and semistructured formats (HTML, XML, etc), as well as in tables, spreadsheets and statistical tools. Integration of these data is carried out at a tremendous cost, often at 35% of IT budget.

Virtual database technology is one of the effective solutions for integration of data from heterogeneous databases which is developed by Junglee Corporation. This technology is used in many data integration applications like *yahoo*, *teiid*, *talend*, *Datastage* etc. In VBD the users can manipulate information as if they were stored together in a single place with a single interface whereas it may actually be stored in multiple, possibly heterogeneous places.

MapReduce is a new framework specifically designed for processing huge datasets on distributed sources. Apache's Hadoop is an implementation of MapReduce. Currently Hadoop has been applied successfully for file based datasets. The execution engine that is developed on top of Hadoop applies Map and Reduce techniques to break down the parsing and execution stages for parallel and distributed processing. Moreover MapReduce will provide the fault tolerance, scalability and reliability because its library is designed to help process very large amount of data using hundred and thousands of machine, which must tolerate the machine failure. This paper proposes to utilize the parallel and distributed processing capability of Hadoop MapReduce for handling heterogeneous query execution on

large datasets. So Virtual Database Engine built on top of this will result in effective high performance distributed data integration.

VIRTUAL DATABASE SYSTEM

A virtual database (or VDB) is a container for components used to integrate data from multiple data sources, so that they can be accessed in an integrated manner through a single, uniform API. The standard VDB structure is shown in the Fig. 1. It consists of four components. The Mapper, Publisher, Executor and Wrapper [1][2].

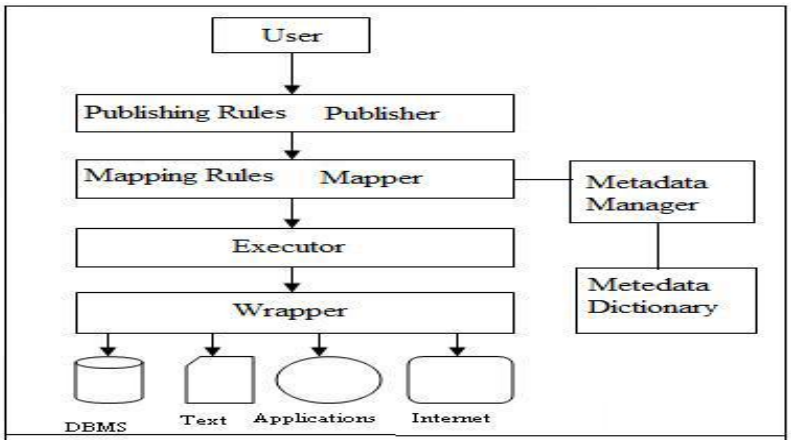

Figure: Virtual Database System-General structure

1) Publisher: The publisher provides a query language for the users to access the system.

2) Mapper: The mapper needs the Metadata information. The query given by the users are processed and decomposed into sub queries [12] according to the actual data retrieval instructions defined in the Metadata [3].

3) Executor: An Executor provides an abstraction layer between Query Engine and physical data source that knows how to convert issued user query commands into source specific commands and execute them using the wrapper. It also has intelligence logic to convert the result data that came from the physical source into a form that Query engine is expecting.

4) Wrapper: A wrapper provides the connectivity to the physical data source. This also provides way to natively issue commands and gather results. A wrapper can be a RDBMS data source, Web Service, text file, connection to main frame etc.

5) Metadata: Metadata is data that describes a specific item of content and where it is located. Metadata is capturing important information about the enterprise environment, data, and business logic to accelerate development, drive integration procedures, and improve integration efficiency [4]. Metadata captures all technical, operational, and business metadata in real time in a single open repository. This

repository ensures that metadata is always up to date, accurate, complete, and available.

Following figure shows the extraction of data from heterogeneous data sources using VDB.

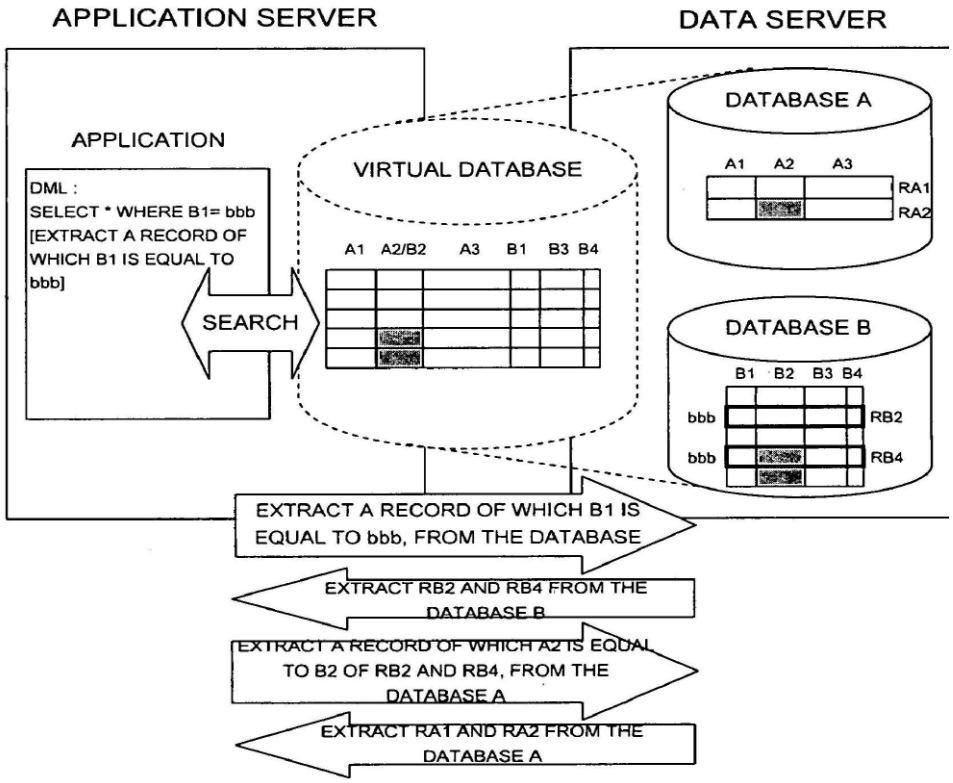

Figure: Extracting data using VDB

The most common interface to VBD is that of a relational database management system, effected through methods such as the Open Database Connectivity (ODBC) method, the Structured Query Language and the relational database model. However, the engine can be implemented with an eXtensible Markup Language (XML) interface. VDB can be accessed through JDBC-SQL, SOAP (Web Services), SOAP-SQL, or XQuery. This project uses the XML for maintaining the Metadata. XML metadata containing all process, map and schema designs integrated with a single, powerful integration engine allows tremendous flexibility and scalability.

MAP REDUCE

MapReduce is a programming model for expressing distributed computation on massive amount of data and an execution framework for large-scale data processing on clusters of commodity servers[5][6]. It was originally developed by Google and built on well known principles in parallel and distributed processing. Hadoop is the open source implementation of MapReduce [7][8][9] written in java which provides reliable, scalable and fault tolerance distributed computing. Hadoop environment set up involves a great number of parameters

which are crucial to achieve best performance. It allows programmers to develop distributed applications without any distributed knowledge.

Key-value pair forms the basic data structure in MapReduce. Keys and values may be primitives such as integers, floating point values, strings, and raw bytes or they may be arbitrary complex structures (lists, tuples, associative array, etc.). Programmers typically need to define their custom data types. The map function takes the input record and generates intermediate key and value pairs. The reduce function takes an intermediate key and a set of values to form a smaller set of values. Typically just zero or one output value is produced by the reducer. In MapReduce, the programmer defines a mapper and reducer with the following signature:

Map (k1, v1) → [(k2, v2)]

Reduce (k2, [v2]) → [(k3, v3)]

[….] denotes the list.

MapReduce framework is responsible for automatically splitting the input, distributing each chunk to workers (mappers) on multiple machines, grouping and sorting all intermediate values associated with the intermediate key, passing these values to workers (reducers) on multiple resources, this is shown in Fig.3. Monitoring the execution of mappers and reducers as to re-execute them when failures are detected is done by the master.

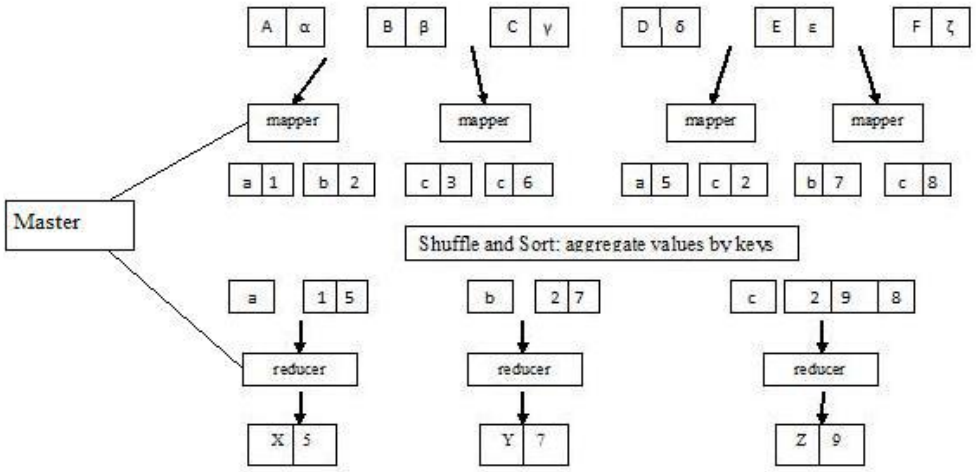

Figure: Simplified view of MapReduce

It is not uncommon for MapReduce jobs to have thousands of individual tasks that need to be assigned to nodes in the cluster. In large jobs, the total number of tasks may exceed the number of tasks that can be run on the clusters concurrently, making it necessary for the scheduler to maintain some sort of a task queue and to track the progress of running tasks, so that waiting tasks can be assigned to nodes as they become available.

IMPLEMENTATION

The goal of distributed query processing is to execute such queries as efficiently as possible in order to minimize the response time that users must wait for answers or the time application programs are delayed. And to minimize the total communication costs associated with a query, to improved throughput via parallel processing, sharing of data and equipment, and modular expansion of data management capacity. In addition, when redundant data is maintained, one also achieves increased data reliability and improved response time. This is achieved through introducing MapReduce with VDB.

In order to improve the performance efficiency of the VDB the Hadoop MapReduce is added at the executor phase. The executor will pass the mapper's sub query to the Master of the MapReduce. The master will automatically split the input into chunks (splits) and finds M mappers and R reducers. The splits can be processed in parallel by the mappers. Reduce invocations are distributed by partitioning the intermediate key space into R pieces using a partitioning function. The number of partitions (R) and partitioning functions are specified by the user. The output of the R Reducers stored in R output files. This output files will fit our needs. This is shown in Fig. given below. The output will be sent back to the user.

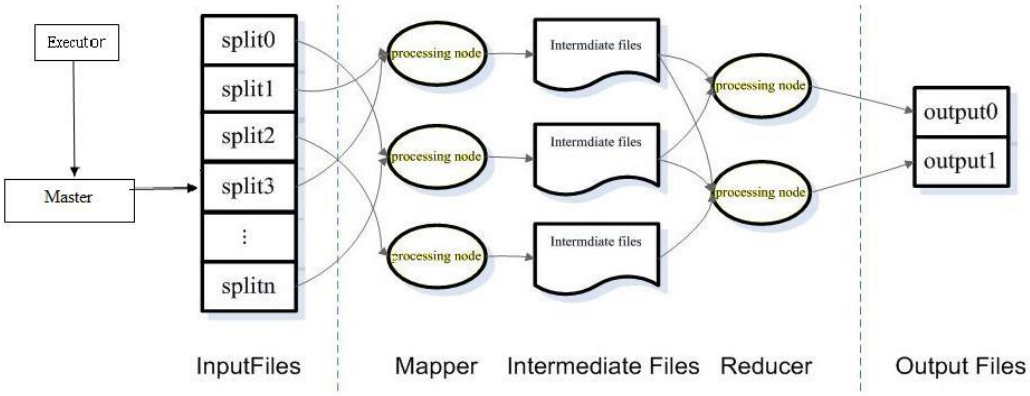

Figure: MapReduce execution flow with VDB

QUERY OPTIMIZATION

Although the parallel processing improves the efficiency of the engine, Query optimization[11] of heterogeneous data integration is one of the key issues to be resolved urgently, but the distribution of local data sources, autonomy, as well as the heterogeneous nature makes it very difficult to optimize.

In VDB engine the users' queries are decomposed into subqueries by the mapper with the help of the Metadata manager. The engine contains more than one mapper and executor. In distributed networks, high communication cost is the main reason of leading to low query response time. The query response time can be improved by minimizing communication overload and therefore improving optimizing the sub query processing.

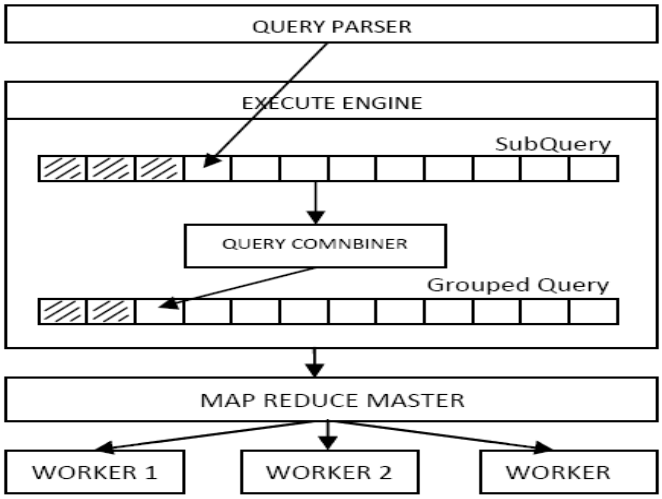

Figure: Query Optimization

For the purpose of grouping together similar queries to same datasource, a queue is introduced in the Execute Engine. The incoming subqueries from the Query Parser is queued in this queue. A Query Combiner acts on this queue in a periodic interval to look for similar queries to same data source, and groups them together. This grouped query is now sent to a Grouped Query Queue. The MapReduce Master removes the grouped queries from this queue and processes using different workers. This effectively optimizes the time spent in processing and executing duplicate queries.

CONCLUSIONS

Virtual Database system is to integrate data from heterogeneous databases. But its performance degrades when working with huge datasets. This problem is rectified by introducing the Hadoop MapReduce with VDB. The proposed system of this paper can be used in applications that use the VDB technology to handle huge datasets. This paper proposes an implementation methodology for leveraging MapReduce functionality in VBD engine. Also, it emphasizes the importance of Query optimization, i.e. grouping together of similar queries for same datasource. In this paper a query optimization logic has been proposed for handling similar queries with no parameters. The optimization of queries with parameters is out of this paper's scope and it could be taken up as a future work.

REFERENCES

[1] Asish Gupta, Venkey Harinarayan, Anand Rajaraman. Virtual Database Technology, ACM Sigmod Record 26 (4)(1994) 57-61.

[2] Wenhao Xu, Jing Li, Yongwei Wu, Xiaomeng Huang, Guangwen Yang, VDM: Virtual Database Management for Distributed and File System, Grid and Cooperative Computing (2008), IEEE.

[3] Yuji Wada, Yuta Watanabe, Keisuke Syoubu, Jun Sawamoto, Takashi Katoh. Virtual Database Technology for Distributed Database, 2010 IEEE 24th, International Conference on Advanced Information Networking and Applications Workshop.
[4] Ferreira.R,Moura-ires,J.,Martins,R.,Pntoquilho.M., XML based Metadata Repository for Information Systems, IEEE Artificial intelligence conference, 2005.
[5] Jeffrey Dean and Sanjay Ghemawat. MapReduce*:* Simplified Data Processing on Large Clusters. Google Research Publication (2004).
[6] Ralf Lammel. Google's MapReduce Programming Model Revisited.Science of Computer Programming archive. Volume 68, (2008).
[7] Apachee Hadoop,http://Hadoop.apache.org.
[8] Hammoud, S., Maozhen Li, Yang Liu, Alham N.K., Zelong Liu. MRSim: A discrete event based MapReduce simulator. Seventh International IEEE Conference on Fuzzy Systems and Knowledge Discovery (FSKD), 2010.
[9] Tom White. Hadoop: The Definitive Guide. O'Reilly, Scbastopol, California, 2009.
[10] Gang Chen, Yongwei Wu, Jia Liu, Guangwen Yang and Weimin Zheng. Optimization of subquery processing in distributed data integration systems. Journal of Network and Computer Applications (2010).
[11] Jong-Hyun Park,Ji-Hoon Kang. Optimization of XQuery Queries including FOR Clauses[C]// The Second nternational Conference on Internet and Web Applications and Services. Washington DC: IEEE Computer Society, 2007:37-44.
[12] Alon Y. Levy, Anand Rajaraman, Joann J.Ordille. Querying Heterogeneous Information Sources Using Source Descriptions [J].VLDB,1996,3(6):251-262.