ISSN (ONLINE): 2454-9762
ISSN (PRINT): 2454-9762
Available online at www.ijarmate.com

International Journal of Advanced Research in Management, Architecture, Technology and Engineering (IJARMATE) Vol. 2,Special Issue 6, March 2016

DATA PROCESSING IN CLOUD WITH AVL STRUCTURE AND BOOTSTRAP ACCESS CONTROL

HEMAVATHI.K, PRIYANKA.A
MAGNA COLLEGE OF ENGINEERING

ABSTRACT:The corporate networks such as supply chain networks have to share relevant information among the companies which are in collaboration in the same industry sector. So, they have to store, retrieve and process a huge amount of data. This requires huge databases and servers. Hence they choose third party data warehouse to store their data.. Also a data warehouse is static that is its storage procedure is constant. With the advent of cloud computing, corporate networks store their data in the shared knowledge plane which is cloudy. But they have to decide which part of the data would be visible for which users. To accomplish this task, the bootstrap access control mechanism is used which decides the accessibility of the user while entering the database. So, the data are processed with map reduce algorithm and stored in a BalancedOverlay Network (BATON). BATON uses AVL tree structure for storing data. Also, it supports peer to peer system in which a separate server is provided for each company in the corporate network and in each server, the required redundancy is maintained. "Pay as you go" pricing policy is followed in this system which is a two way (Service provider and corporate network) beneficial pricing policy. In this policy, the user will be charged by categorizing his needs based on a range of memory required, duration and security policy.

I.INTRODUCTION

COMPANIES of the same industry sector are often connected into a corporate network for collaboration purposes. Each company maintains its own site and installation of alarge-scale centralized data warehouse system entailsnontrivial costs including huge hardware/software investments(a.k.a total cost of ownership) and high maintenancecost (a.k.a total cost of operations) [16]. In the realworld, most companies are not keen to invest heavily onadditional information systems until they can

selectively shares a portion of its business data with From a technical perspective, the key for the success of a corporate network is choosing the right data sharing platform, a system which enables the shared data (stored and maintained by different companies) network-wide visible and supports efficient analytical queries over those data. Traditionally, data sharing is achieved by building a centralized data warehouse, which periodically extracts data from the internal production systems (e.g., ERP) of each company for subsequent querying. Unfortunately, such a warehousing solution has some deficiencies in real deployment.

First, the corporate network needs to scale up to support thousands of participants, while theinstallation of a large-scale centralized data warehouse system entails nontrivial costs including huge hardware/software investments(a.k.a total cost of ownership) and high maintenance cost (a.k.a total cost of operations) [16]. In the real world, most companies are not keen to invest heavily on additional information systems until they can clearly see the potential return on investment (ROI) [21]. Second, companies want to fully customize the access control policy to determine which business partners can see which part of their shared data. Unfortunately, most of the data warehouse solutions fail tothe others. Examples of such corporate networks include supply chain networks where organizations such as suppliers, manufacturers, and retailers collaborate with each other to achieve their very own business goals including planning production-line, making acquisition strategies and choosing marketing solutions.

clearly seethe potential return on investment (ROI) [21]. Second,companies want to fully customize the access
control policyto determine which business partners can see whichpart of their shared data.
Unfortunately, most of the data warehouse solutions fail

to offer such flexibilities. Finally,to maximize the revenues, companies often dynamicallyadjust their business process and may change their businesspartners. Therefore, the participants may join andleave the corporate networks at will. The data warehousesolution has not been designed to handle such dynamicity.

To address the aforementioned problems, this paperpresents BestPeer++, a cloud enabled data sharing platformdesigned for corporate network applications. By integratingcloud computing, database, and peer-to-peer (P2P) technologies,BestPeer++ achieves its query processing efficiencyand is a promising approach for corporate network applications,with the following distinguished features.

BestPeer++ is deployed as a service in the cloud.To form a corporate network, companies simplyregister their sites with the BestPeer++ service provider,launch BestPeer++ instances in the cloudand finally export data to those instances for sharing.BestPeer++ adopts the pay-as-you-go businessmodel popularized by cloud computing [9]. Thetotal cost of ownership is therefore substantiallyreduced since companies do not have to buy anyhardware/software in advance. Instead, they payfor what they use in terms of BestPeer++ instance's hours and storage capacity.

BestPeer++ employs a hybrid design for achievinghigh performance query processing. The majorworkload of a corporate network is simple, lowoverheadqueries. Such queries typically onlyinvolve querying a very small number of businesspartners and can be processed in short time. Best-Peer++ is mainly optimized for these queries. Forinfrequent time-consuming analytical tasks, we providean interface for exporting the data from Best-Peer++ to Hadoop and allow users to analyze thosedata using MapReduce.

In summary, the main contribution of this paper is thedesign of BestPeer++ system that provides economical,flexible and scalable solutions for corporate network applications.We demonstrate the efficiency of BestPeer++ bybenchmarking BestPeer++ against HadoopDB [2], arecently proposed large-scale data processing system,over a set of queries designed for data sharing applications.

The results show that for simple, low-overheadqueries, the performance of BestPeer++ is significantly betterthan HadoopDB.The rest of the paper is organized as follows. Section 2presents the overview of BestPeer++ system. We subsequentlydescribe the design of BestPeer++ core components,including the bootstrap peer in Section 3 and thenormal peer in Section 4. The pay-as-you-go query processingstrategy adopted in BestPeer++ is presented in Section 5.Section 6 evaluates the performance of BestPeer++ in termsof efficiency and throughput.

II.EXISTING SYSTEM

The corporate networks stores their bulk data in a third party data warehouse which may have security threats and high cost.Also access control mechanism is not well defined for various categories of users like suppliers, manufacturers and retailers.The data are stored in an unstructured manner so that retrieval of data and processing them are tedious.

There is no processing like sorting or clustering before storing the data from the server. So, while retrieving, the complexity of the system will be increased because it has to search for the whole database.

The scalability of the system is very low since it cannot scale up to thousands of articipants.

The storage of data in the data warehouse system entails non trivial costs, including hardware/software investment and high maintenance cost.

The inside processing of data marts and classification of fact tables and dimension tables is complex tasks when we store data in the data warehouse.

The system is not supported for heterogeneous environment that is participants of the network using different platforms cannot be supported.

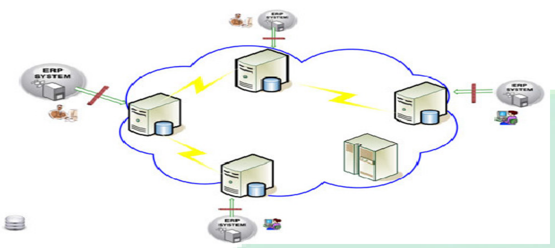
Best Peer++ extends the role-based access control for the inherent distributed environment of corporate networks. Through a web console interface, companies can easily configure their access control policies and prevent undesired business partners to access their shared data. Best Peer++ employs P2P technology to retrieve data between business partners. Best Peer++ instances are organized as a structured P2P overlay network named BATON [13]. The data are indexed by the table name, column name and data range for efficient retrieval.

III.PROPOSED SYSTEM

The data to be shared in the in the corporate network are stored in the cloud database after proper processing.base. The data are structured and stored in the balanced tree overlay network, which uses the AVL tree structure.The retrieval of data from these databases is simple since they are stored in a well structured manner.

The access control for various categories of participants of the corporate network can be implemented using bootstrap server.The scalability of the system is high as it can hold thousands of participants without any complexity.

SYSTEM ARCHITECTURE



MAP REDUCE ALGORITHM

MapReduce is a programming model and an associated implementation for processing and generate large data sets with a parallel, distributed algorithm on a cluster.

A MapReduce program is composed of a **Map()** procedure that performs filtering and sorting (such as sorting students by first name into queues, one queue for each name) and a **Reduce()** procedure that performs a summary operation (such as counting the number of students in each queue, yielding name frequencies). The "MapReduce System" (also called "infrastructure" or "framework") orchestrates the processing by marshalling the distributed servers, running the various tasks in parallel, managing all communications and data transfers between the various parts of the system, and providing for redundancy and fault tolerance.The model is inspired by the map and reduce functions commonly used in functional programming, although their purpose in the MapReduce framework is not the same as in their original forms. The key contributions of the MapReduce framework are not the actual map and reduce functions, but the scalability and fault-tolerance achieved for a variety of applications by optimizing the execution engine once. As such, a single-threaded implementation of MapReduce (such as MongoDB) will usually not be faster than a traditional (non-MapReduce) implementation; any gains are usually only seen with multi-threaded implementations. Only when the optimized distributed shuffle operation (which reduces network communication cost) and fault tolerance features of the MapReduce framework come into play, is the use of this model beneficial.MapReduce libraries have been written in many programming languages, with different levels of optimization. A popular open-source implementation is Apache Hadoop. The name MapReduce originally referred to the proprietary Google technology, but has since been generalized.

BENCH MARKING

This section evaluates the performance and throughput of BestPeer++ on Amazon cloud platform. For the performancebenchmark, we compare the query latency of Best-Peer++ with HadoopDB using five queries selected from typical corporate network applications workloads. For the throughput benchmark, we create a simple supply-chain network consisting of suppliers and retailers and study the query throughput of the system.

Performance Benchmarking

This benchmark compares the performance of BestPeer++ with HadoopDB. We choose HadoopDB as our benchmarktarget since it is an alternative promising solutionfor our problem and adopts an architecture similar toours. Comparing the two systems (i.e., HadoopDB andBestPeer++) reveals the performance gap between a generaldata warehousing system and a data sharing systemspecially designed for corporate network applications.

Benchmark Environment

We run our experiments on Amazon m1.small DBinstances launched in the ap-southeast-1 region. Each DB small instance has 1.7 GB memory, 1 EC2 Compute Unit (1 CPU virtual core). We attach each instance with 50GB storage space. We observe that the I/O performance of Amazon cloud is not stable. The hdparm reports that the buffered read performance of each instance ranges from 30 to 120 MB/sec. To produce a consistent benchmark
result, we run our experiments at the weekend when most of the instances are idle. Overall, the buffered read performance of each small instance is about 90 MB/sec during our benchmark. The end-to-end network bandwidth
between DB small instances, measured by iperf, is approximately 100 MB/sec. We execute each benchmarkquery three times and report the average execution time. The benchmark is performed on cluster sizes of 10,20, 50 nodes. For the Best Peer++ system, these nodes are normal peers. We launch an additional dedicated node asthe bootstrap peer. For HadoopDB system, each launched cluster node acts as a worker node which hosts a Hadooptask tracker node and a

PostgreSQL database server instance. We also use a dedicated node as the Hadoop jobtracker node and HDFS name node.

ALGORITHM (BOOTSTRAP DAEMON())

```
While true do
Status S=invokeCloudWatch ()
Array List peerList=BootStrap.getAllPeer ()
Array List newPeer=new Array List()
for i=0 to peerList.size () do
if peerList.get (i).fails() then
Peer peer=new Peer ()
peer.loadMySQLBackUpFrontRDS (peerList.get (i))
newPeer.add (peer)
BootStrap.setBlackList (peerList.get (i))
else
If peerList.get (i).overloaded () then
Peer peer=new Peer ()
peer. Upscale((peerList.get (i))
peer. Clone (peerList.get (i_.getDB ())
BootStrap.setBlack.List (peerList.get (i))
newPeer.add (peer)
BootStrap.removeAllPeersInBlackList ()
BootStrap.addAllNewPeer (newPeer)
BootStrap.broadcastNetworkStatus ()
Sleep T seconds
```

HADOOPDB SETTINGS

We carefully follow the instructions presented in the original HadoopDB paper to configure HadoopDB. The settingconsists of the setup of a Hadoop cluster and the PostgreSQLdatabase server hosted at each worker node. We use Hadoop version 0.19.2 running on Java 1.6.0_20. The block size of HDFS is set to be 256 MB. The replication factor is set to 3. For each task tracker node, we run one map task and one reduce task. The maximum Java heap size consumed by the map task or the reduce task is 1024 MB. The buffer size of read/write operations is set to 128 KB. We also set the sort buffer of the map task to 512 MB with 200 concurrent streams for merging. For reduce task, we set the number of threads used for parallel file copying in the shuffle phase to be 50. We also enable the buffer reusebetween the shuffling phase and the merging phase. As a final optimization, we enable JVM reuse. For the PostgreSQL instance, we run PostgreSQL version 8.2.5 on eachworker node. The shared buffers used by PostgreSQL is set to 512MB. Theworking memory size is 1 GB.We only present the results for SMS-coded HadoopDB, i.e., the query plan is generated byHadoopDB's SMS planner.

DATA LOADING

The data loading process of BestPeer++ is performed by all normal peers in parallel and is consisted of two steps. In thefirst step, each normal peer invokes the data loader to load raw TPC-H data into the local MySQL databases. In additionto copying raw data, we also build indices to speedup query processing. First, a primary index is built for each TPC-Htable on the primary key. Second, some additional secondary indices are built on selected columns of TPC-H tables. Table 4 summarizes the secondary indices that we built. After the data is loaded into the local MySQL database, each normalpeer invokes the data indexer to publish index entries to the BestPeer++ network. For each table, the data indexer publishes a table index entry and a column index entry for each column. Since the values in TPC-H data sets follow uniform distribution, each normal peer holds approximately the same data range for each column of the table, therefore, we do not configure normal peer to publish range index. For HadoopDB, data loading process is straightforward.For each worker node, we load only raw data into the local PostgreSQL database instance using SQL COPYcommand and build corresponding primary and secondary indices for each table. We did not employ the GlobalHasher and Local Hasher to further co-partition tables. HadoopDB co-partitions tables among worker nodes onjoin key in order to speed up join processing.5 However, a corporate , data is fully controlled by eachbusiness. It is undesirable for a certain business to movedata to normal peers managed by other businesses due toprivacy and safety concern. Therefore, we disabled thisco-partition function for HadoopDB.

DATA SETS

Our benchmark consists of five queries, denoted as Q1, Q2, Q3, Q4, and Q5 which are executed on the TPC-H data sets. We implement the benchmark queries by ourselves since the TPC-H queries are complex and time-consumingqueries which are not suitable for benchmarking corporate network applications. The TPC-H benchmark data set consists of eight tables. We use the original TPC-H schema as the shared global schema. HadoopDB does not support schema mapping and access control. To benchmark the two systems in the same environment, we perform additional configurations on Best- Peer++ as follows. First, we set the local schema of each normal peer to be identical to the global schema.

BENCHMARK SETTINGS

We establish a simple supply-chain network to benchmark the query throughput of the BestPeer++ system. The supply-chain network consists of a group of suppliers and a group of retailers which query data from each other. Each normal peer either acts as a supplier or a retailer. We set the number of suppliers to be equal to the number of retailers. Thus, in the cluster with 10, 20, and 50 normal peers, there are 5, 10, and 25 suppliers and retailers, respectively. We still use the TPC-H schema as the global shared schema, but partition the schema into two sub-schema, one for suppliers and the other for retailers. The supplier schema consists of the following tables: Supplier, PartSupp, and Part. The retailer schema involvesLineItem, Orders, and Customer tables. The Nation and Region tables are commonly owned by both supplier peers and retailers peers. We partition the TPC-H datasets into 25 data sets, one data set for each nation, andconfigure each normal peer to only host data from aunique nation. The data partition is performed by firstpartitioning Customer and Supplier tables accordingto their nation keys. Then, joining each Supplier andCustomer data set with the other four tables (i.e., Part,PartSupp, Orders, LineItem respectively, the joinedtuples in those tables finally form the corresponding partitioneddata sets. To reflect the fact that each table is partitionedbased on nations, we modify the original TPC-Hschema and add a nation key column in each table.For scalability evaluation, we scale-up the amount ofdata and the number of normal peer proportionally. Eventually,we generate a 50 GB raw TPC-H data set on 50 normalpeers, which consists of 25 suppliers and 25 retailers,and measure the absolute system throughput for the twotypes of peers respectively. In the performance evaluation,we retain the data size and peer scale (50 normal peers and50 GB data in our setup), and increase the throughput, untilthe point at which the system is saturated and throughputstops increasing. We report the average latency versusthroughput curve, as in the YCSB [5] tool's terminology.We configure the access control module as follows. Weset up two roles: supplier and retailer. The supplier role isgranted full access to tables hosted by retailer peers. Theretailer role is granted full access to tables hosted by supplierpeers. We should not be confused with the supplier.

IV.CONCLUSION

We have discussed the unique challenges posed by sharing and processing data in an inter-businesses environment and proposed BestPeer++, a system which delivers elastic data sharing services, by integrating cloud computing, database, and peer-to-peer technologies. The benchmark conducted on Amazon EC2 cloud platform shows. This work was supported by the Singapore Ministry of Education Grants No. MOE2010-T2-2-104 named epic. We would also like to thank anonymous reviewers for insightful comments.

REFERENCES

[1] K. Aberer, A. Datta, and M. Hauswirth, "Route MaintenanceOverheads in DHT Overlays," in 6th Workshop Distrib. DataStruct., 2004.
[2] A. Abouzeid, K. Bajda-Pawlikowski, D.J. Abadi, A. Rasin, and A.Silberschatz, "HadoopDB: An Architectural Hybrid of MapReduceand DBMS Technologies for Analytical Workloads," Proc.
VLDB Endowment, vol. 2, no. 1, pp. 922-933, 2009.
[3] C. Batini, M. Lenzerini, and S. Navathe, "A Comparative Analysisof Methodologies for Database Schema Integration," ACM ComputingSurveys, vol. 18, no. 4, pp. 323-364, 1986.
[4] D. Bermbach and S. Tai, "Eventual Consistency: How Soon isEventual? An Evaluation of Amazon s3's Consistency Behavior,"
in Proc. 6th Workshop Middleware Serv. Oriented Comput. (MW4SOC'11), pp. 1:1-1:6, NY, USA, 2011.
[5] B. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears,"Benchmarking Cloud Serving Systems with YCSB," Proc. FirstACM Symp. Cloud Computing, pp. 143-154, 2010.