ISSN 23499842(Online), Volume 1, Special Issue 1(NCFCSIT 15), March 2015
International Journal of Innovative Trends and Emerging Technologies

MINING OF LARGE SCALE DATA USING BESTPEER++ STRATEGY

*S. ANUSUYA,*R.B. ARUNA,*V. DEEPASRI,**DR.T. AMITHA
*UG Students, **Professor
Department Of Computer Science and Engineering
Dhanalakshmi College of Engineering, Chennai
anusuyasankar1994@gmail.com, aruna.m.balaji@gmail.com, shri.deepa.26@gmail.com, tamitharaghu@gmail.com

Abstract*— Business organizations of same industry sector generally communicate with each other for selective data sharing purposes and to collaborate with each other to achieve their business goals. Hence to reduce the operational costs and to increase the revenues the corporate network should choose right sharing platform. In this paper, we present BestPeer++ strategy which is a system that delivers elastic data sharing services for corporate network applications in Cloud based on Peer to Peer (P2P) based data management platform.By integrating Cloud computing, Database and Peer to Peer technologies into one system, BestPeer++ provides an economical, flexible and scalable platform for corporate network applications and delivers data sharing services based on Pay-as-you-go business model. We evaluate BestPeer++ on Amazon EC2 Cloud platform that can demonstrate linear scalability for throughput with respect to the number of nodes.

Keywords— Peer to peer systems, Amazon EC2 Cloud, Pay-As-You-Go model, Map Reduce, Query processing

I. INTRODUCTION

The rise and evolution of the peer-to-peer (P2P) file sharing networks and some of the reasons for their popularity are introduced in this paper. In addition, the security implications to users' computers, networks, and information, are also examined. And finally, the status of P2P networks in business is discussed, as well as a summary of the current state of the security risks of this technology. Companies of the same industry sector are often connected into a corporate network for collaboration purposes. Each company maintains its own site and selectively shares a portion of its business data with the 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. Integrating cloud computing, database, and peer-to-peer (P2P) technologies, BestPeer++ achieves its query processing efficiency and is a promising approach for corporate network applications, with the following distinguished features.

II. PROBLEM STATEMENT

A warehousing solution like HadoopDB has some deficiencies in real deployment. First, the corporate network needs to scale up to support thousands of participants, while the installation of a large-scale centralized data warehouse system entails nontrivial costs including huge hardware/software investments and high maintenance cost. 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). Second, companies want to fully customize the access control policy to determine which business partners can see which part of their shared data. 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, most of the data warehouse solutions fail to offer such flexibilities. Finally, to maximize the revenues, companies often dynamically adjust their business process and may change their business partners. Therefore, the participants may join and leave the corporate networks at will. The data warehouse solution has not been designed to handle such dynamicity.

MERITS & DEMERITS

1. It can provide affordable store of all company's data for later use.
2. It can provide another copy available for use (Resilient to failure).
3. Most of the data warehouse solutions fail to offer such flexibilities.
4. Solution has not been designed to handle such dynamicity.

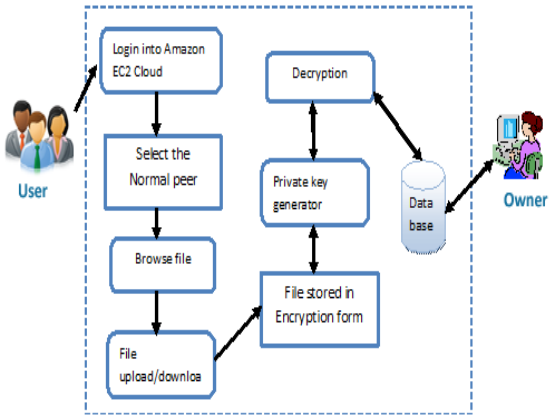


Fig 1: System Architecture

III. PROPOSED SYSTEM

The main contribution of this paper is the design of BestPeer++ system that provides economical, flexible and scalable solutions for corporate network applications. We demonstrate the efficiency of BestPeer++ by benchmarking BestPeer++ against HadoopDB, a recently proposed large-scale data processing system, over a set of queries designed for data sharing applications. By means of simple and low-overhead queries, the performance of BestPeer++ will be significantly better than HadoopDB. 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. BestPeer++ is deployed as a service in the cloud. To form a corporate network, companies simply register their sites with the BestPeer++ service provider, launch BestPeer++ instances in the cloud and finally export data to those instances for sharing.

MERITS & DEMERITS

1. Our system can efficiently handle typical workloads in a corporate network and can deliver near linear query throughput as the number of normal peers grows. BestPeer++ adopts the pay-as-you-go business model popularized by cloud computing.
2. BestPeer++ extends the role-based access control for the inherent distributed environment of corporate networks.
3. BestPeer++ employs P2P technology to retrieve data between business partners.
4. BestPeer++ is a promising solution for efficient data sharing within corporate networks.

IV. LIMITATION

PROBLEM:

Most of the data warehouse solutions fail to offer the access control policy to determine which business partners can see which part of their shared data.

SOLUTION:

BestPeer++ achieves its query processing efficiency to handle sharing data within the corporate networks.

V.METHODOLOGY

A. BestPeer++ :

BestPeer++ employs a hybrid design for achieving high performance query processing. The major workload of a corporate network is simple, low overhead queries. Such queries typically only involve querying a very small number of business partners and can be processed in short time. Best- Peer++ is mainly optimized for these queries. For infrequent time-consuming analytical tasks, we provide an interface for exporting the data from Best- Peer++ to Hadoop and allow users to analyze those data using MapReduce.

The software components of BestPeer++ are separated into two parts: core and adapter.

Core :

The core contains all the data sharing functionalities and is designed to be platform independent.

Adapter :

The adapter contains one abstract adapter which defines the elastic infrastructure service interface and a set of concrete adapter components which implement such an interface through APIs provided by specific cloud service providers (e.g., Amazon).

B. Amazon Cloud Adapter

Amazon EC2 service to provision the database server. Each time a new business joins the BestPeer++ network, a dedicated EC2 virtual server is launched for that business. The newly launched virtual server (called a BestPeer++ instance) runs a dedicated MySQL database software and the BestPeer++ software. The BestPeer++ instance is placed in a separate network security group (i.e., a VPN) to prevent invalid data access. Users can only use BestPeer++ software to submit queries to the network. Amazon relational data service (RDS) to back up and scale each BestPeer++ instance.2 The whole

MySQL database is backed up to Amazon's reliable EBS storage devices in a four-minute window. In order to provide high availability service, BestPeer++ performs asynchronous back-up operation, and there will be no service interrupt during the back-up process. The scaling scheme of BestPeer++ consists of two dimensions: processing and storage, which scale up independently according to user's computation requirement. Initially, each BestPeer++ instance is launched as a m1.small EC2 instance (1 virtual core, 1.7 GB memory) with 5 GB storage space. With the growth of business demand, user can scale up to a more powerful EC2 instance (e.g., m1.large instance which has four virtual cores and 7.5 GB memory). In another word, there is no limitation on the resources used. The Amazon Cloud Adapter also provides automatic fail-over service. In a BestPeer++ network, a special BestPeer++ instance (called bootstrap peer) monitors the health of all other BestPeer++ instances, by querying the Amazon Cloud Watch service. If an instance fails to respond to the bootstrap peer (e.g., crashed), Amazon Cloud Adapter is called to perform fail-over for that instance.

Map/Reduce

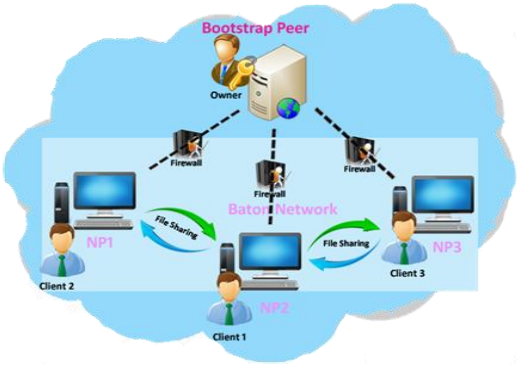
- Programming model from LISP(and other functional languages).
- Easy to distribute across nodes
- Nice retry/failure semantics

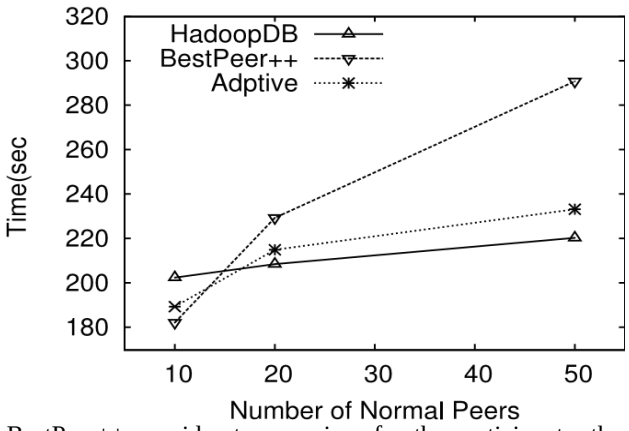
Pay-as-you-go Query Processing

BestPeer++ provides two services for the participants: the storage service and search service, both of which are charged in a pay-as-you-go model. The pay as-you-go query processing module which offers an optimal performance within the user's budget.

The semantics of query processing in the BestPeer++. After data are exported from the local business system into a BestPeer++ instance, the schema mapping rules to transform them into the predefined formats. In this way, given a table T in the global schema, each peer essentially maintains a horizontal partition of it.

BestPeer++ network deployed on Amazon Cloud





BestPeer++ provides two services for the participants: the storage service and search service, both of which are charged in a pay-as-you-go model. The semantics of query processing in the BestPeer++. After data are exported from thelocal business system into a BestPeer++ instance, the schema mapping rules to transform them into the predefined formats. In this way, given a table T inthe global schema, each peer essentially maintains a horizontal partition of it.

VI. ALGORITHM

Algorithm 1: BootStrapDaemon()

```
1:  while true do
2:      Status S = invokeCloudWatch()
3:      ArrayList peerList = BootStrap.getAllPeer()
4:      ArrayList newPeer= new ArrayList()
5:      for i=0 to peerList.size() do
6:          if peerList.get(i).fails() then
7:              Peer peer = new Peer()
8:              peer.loadMySQLBackUpFromRDS(peerList.get(i))
9:              newPeer.add(peer)
10:             BootStrap.setBlackList(peerList.get(i))
11:         else
12:             if peerList.get(i).overloaded() then
13:                 Peer peer = new Peer()
14:                 peer.upScale(peerList.get(i))
15:                 peer.clone(peerList.get(i).getDB())
16:                 BootStrap.setBlackList(peerList.get(i))
17:                 newPeer.add(peer)
18:     BootStrap.removeAllPeersInBlackList()
19:     BootStrap.addAllNewPeer(newPeer)
20:     BootStrap.broadcastNetworkStatus()
21:     sleep T seconds
```

Query Semantic :

Algorithm 2: Adaptive Query Processing

Input: Query Q
Output: Query configuration on a specific query engine

TableSet $S \longleftarrow TableParser(Q)$;
Cost $C_{min} \longleftarrow MAX\_VALUE$;
QueryPlan $Target \longleftarrow null$;
QueryPlanSet $QS \longleftarrow \emptyset$;
foreach Table $T \in S$ **do**
    // Generate Processing Graphs rooted on T
    GraphSet $GS = GraphGen(T)$;
    // Iterate through all Processing Graph rooted on T
    foreach Graph $G \in GS$ **do**
        QueryPlan $P_1 = P2PPlanGen(G)$;
        QueryPlan $P_2 = MapredPlanGen(G)$;
        $QS = QS \cap \{P_1\}$;
        $QS = QS \cap \{P_2\}$;
foreach QueryPlan $P \in QS$ **do**
    if $CostEst(P) < C_{min}$ **then**
        $C_{min} = CostEst(P)$;
        $Target = P$;
return $Target$;

VII. PERFORMANCE ANALYSIS

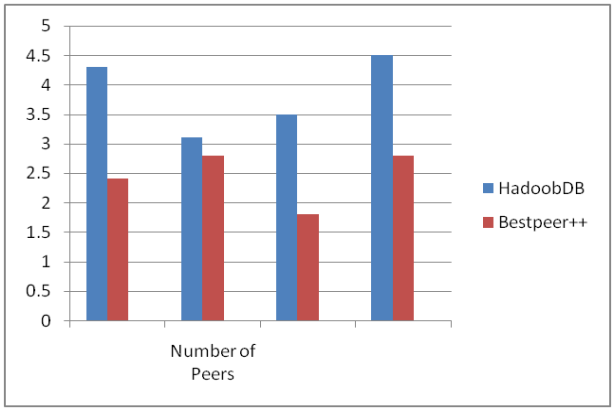
This section evaluates the performance and throughput of BestPeer++ on Amazon cloud platform. For the performance benchmark, we compare the query latency of BestPeer++ with HadoopDB using five queries selected from typical corporate network applications workloads. For the throughput benchmark, we create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

A. Throughput Benchmarking

This section studies the query throughput of BestPeer++. HadoopDB is not designed for high query throughput, therefore, we intentionally omit the results of HadoopDB and only present the results of BestPeer++. We conduct two tiers of benchmark evaluation for the performance and scalability of BestPeer++, respectively.

and is defined as follows:

```
SELECT c_custkey, c_name,
   SUM(l_extendedprice*(1-l_discount)) AS R
   FROM Customer, Orders, LineItem, Nation
   WHERE c_custkey = o_custkey
         AND l_orderkey = o_orderkey
         AND c_nationkey = n_nationkey
         AND l_returnflag = 'R'
         AND o_orderdate >= date(1993-10-01)
         AND o_orderdate < date(1993-12-01)
   GROUP BY c_custkey, c_name
```



Result of query

Adaptive Query Processing Results

To demonstrate our adaptive query processing strategy, we further evaluate query using three different engines separately alone, namely the P2P engine, the MapReduce engine and the adaptive query engine for BestPeer++. To start with, we compile and execute query on either the P2P engine or the MapReduce engine. In each case, we enforce our query planner to invoke either the P2P engine or the MapReduce engine alone, regardless of the possible cost. As described in the previous experiment, the execution strategies of these two engines differ from each other in the way that they shuffles intermediate data and organize the joins, which leads to a considerable performance gap. We then use our adaptive processing engine to make comparison.

The performance of these three processing strategy. The P2P engine works better in a smaller scale (10 data nodes). With the increase of data scale, we witness a decent performance gain from the MapReduce engine, who then outperforms the P2P engine at the scale of 20 and 50 data nodes. Such a trend complies the prediction of our cost model in the sense that the P2P engine handles lighter workload nicely, while on the contrary, the MapReduce scales better with more complex queries.

Taking use of the insight that our cost model gives, the adaptive engine switches between the P2P engine and the MapReduce engine to accommodate itself to a vaster variety of queries in a cost efficient way. The results from Fig. 11 shows the effectiveness end efficiency of the adaptive engine. With a negligible overhead for constructing plans for both engine and evaluating the cost, the performance of the adaptive engine approaches whatever the better one under different workload setups.

VIII. 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 that our system can efficiently handle typical workloads in a corporate network and can deliver near linear query throughput as the number of normal peers grows. Index and optimizer manage the local data and improve the query processing with the database. The performance of BestPeer++ is significantly better than HadoopDB. Therefore, BestPeer++ is a promising solution for efficient data sharing within corporate networks.

IX. FUTURE ENHANCEMENT

In future, Bestpeer++ will promise to handle the efficient data sharing within the corporate networks.

REFERENCES

[1] B. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking Cloud Serving Systems with YCSB," Proc. First ACM Symp. Cloud Computing, pp. 143-154, 2010.

[2] D. Bermbach and S. Tai, "Eventual Consistency: How Soon is Eventual? 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.

[3] J. Dittrich, J. Quian_e-Ruiz, A. Jindal, Y. Kargin, V. Setty, and J. Schad, "Hadoop++: Making a Yellow Elephant Run Like a Cheetah (without it Even Noticing)," Proc. VLDB Endowment, vol. 3, no. 1/2, pp. 515-529, 2010.

[4] A. Abouzeid, K. Bajda-Pawlikowski, D.J. Abadi, A. Rasin, and A. Silberschatz, "HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads," Proc. VLDB Endowment, vol. 2, no. 1, pp. 922-933, 2009.

[5] H. Garcia-Molina and W.J. Labio, "Efficient Snapshot Differential Algorithms for Data Warehousing," technical report, Stanford Univ., 1996.

[6] Google Inc., "Cloud Computing-What is its Potential Value for Your Company?" White Paper, 2010.

[7] H.V. Jagadish, B.C. Ooi, K.-L. Tan, Q.H. Vu, and R. Zhang, "Speeding up Search in Peer-to-Peer Networks with a Multi-Way Tree Structure," Proc. ACM SIGMOD Int'l Conf. Management of Data, 2006.

[8] A. Lakshman and P. Malik, "Cassandra: Structured Storage System on a P2P Network," Proc. 28th ACM Symp. Principles of Distributed Computing (PODC '09), p. 5, 2009.

[9] A. Thusoo, J. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy, "HIVE: A Warehousing Solution over a Map-Reduce Framework," Proc. VLDB Endowment, vol. 2, no. 2, pp. 1626-1629, 2009.

[10] H.T. Vo, C. Chen, and B.C. Ooi, "Towards Elastic Transactional Cloud Storage with Range Query Support," Proc. VLDB Endowment, vol. 3, no. 1, pp. 506-517, 2010.

[11] S. Wu, Q.H. Vu, J. Li, and K.-L. Tan, "Adaptive Multi-Join Query Processing in PDBMS," Proc. IEEE Int'l Conf. Data Eng. (ICDE '09),pp.1239-1242,2009.