A Peer-To-Peer Data Processing Infrastructure That Operates On The Largest Scale

SINGH ABHISHEK KUMAR
M.Tech Student, Dept of CSE, Malla Reddy College of Engineering and Technology, Hyderabad, T.S, India

P. HONEY DIANA
Assistant Professor, Dept of CSE, Malla Reddy College of Engineering and Technology, Hyderabad, T.S, India

Dr. M.SAMBASIVUDU
Associate Professor, Dept of CSE, Malla Reddy College of Engineering and Technology, Hyderabad, T.S, India

Abstract: **In the corporate network, information may be exchanged throughout different businesses, making it simpler for those with common interests to collaborate. With its help, firms may be able to reduce overhead costs and increase revenue. As data is exchanged and processed across firms, it increases the complexity of implementing a scalable, high-performing, and secure data management system. This article presents BP++, an expansion of the BestPeer P2P data management platform that offers elastic data sharing services for cloud-based business network applications. BP++ integrates cloud computing, databases, and P2P technologies to provide its members with data sharing services under the well-known pay-as-you-go pricing model. For our BP++ tests, we make use of Amazon's EC2 cloud infrastructure. Benchmarks show that BP++ outperforms the recently proposed HadoopDB large-scale data processing solution when both systems are employed to handle typical business network demands. The results demonstrate that BP++ is quite efficient, with throughput scaling practically linearly with the number of peer nodes.**

Keywords: **Peer-To-Peer Systems; Cloud Computing; Mapreduce; Query Processing;**

I. INTRODUCTION:

The ability of a database to provide a consistent, non-redundant representation of all the data that is handled inside an organization is one of the basic ideas that underpin the database method. This is only possible if approaches that facilitate integration across organizational and application boundaries are made accessible. The design activity is often carried out by methodologies for database design in the form of the production of many schemas, each of which represents a component of the application [1]. These schemas are then fused together. Integration of database schemas refers to the process of combining the schemas of several databases, whether they are already in existence or are still in the planning stages. The purpose of this study is to first provide a unified framework for the issue of schema integration and then conduct a comparative evaluation of the work that has been done so far in this field. The ability of a database to provide a consistent, non-redundant representation of all the data that is handled inside an organisation is one of the basic ideas that underpin the database method. This is only possible if approaches that facilitate integration across organizational and application boundaries are made accessible. The design activity is often carried out by methodologies for database design in the form of the production of many schemas, each of which represents a component of the application. These schemas are then fused together. Integration of database schemas refers to the process of combining the schemas of several databases, whether they are already in existence or are still in the planning stages. We present a balanced tree structure overlay over a peer-to-peer network that is efficient at enabling precise queries as well as range queries. This structure can accommodate both types of queries. We demonstrate that the load at each node is roughly equivalent, in spite of the fact that the tree structure makes it necessary to differentiate between nodes located at various levels in the tree [2][3]. We demonstrate that sideways routing tables kept at each node give sufficient fault tolerance to allow quick repair, in spite of the fact that the tree topology offers exactly one route between each pair of nodes. Distributed databases have traditionally operated on the assumption that the (relatively limited) set of nodes that participate in a query is known a priori, that the data is properly arranged, and that statistics are easily accessible. On the other hand, when applied to a peer-based database management system, these presumptions can no longer be trusted (PDBMS). As a result of this, processing and optimizing queries in a PDBMS may be a difficult task [4]. In this work, we provide our distributed answer to the issue of multi-way join queries that we encountered. Our method begins by processing a multi-way join query based on an initial query evaluation plan (which is generated using statistical data that may be out of date or inaccurate); as the query is processed, statistics obtained on-the-fly are used to continuously and accurately refine the current plan dynamically into a more efficient one.

II. PROBLEM STATEMENT:

When put into practice, a solution such as this one for warehousing reveals several flaws. First, the corporate network needs to be scaled up to support thousands of participants. Second, the installation of a large-scale central data warehouse entails nontrivial costs. These costs include huge hardware and software investments (also known as the "total cost of ownership") and high maintenance costs (also known as the "total cost of operations"). Third, the installation of a large-scale central data warehouse system entails nontrivial costs. In the real world, the vast majority of businesses are hesitant to make significant investments in new information systems unless they have a crystal clear picture of the possible return on investment (ROI). Second, businesses desire the ability to completely personalize the access control policy so that they can select which of their business partners have access to which portions of the shared data [5]. The vast majority of data warehouse technologies are unable to provide such degrees of adaptability. This kind of dynamicity is outside the scope of the solution, which was not built to manage it.

III. PROPOSED METHODOLOGIES:

The primary contribution made by this study is the creation of a BP++ system that, when applied to corporate network application problems, offers solutions that are affordable, adaptable, and scalable. We illustrate the efficacy of BP++ by comparing it against HadoopDB, a newly proposed large-scale data processing system, across a set of queries targeted for data sharing applications. This allows us to see how well BP++ performs in comparison to HadoopDB. The findings indicate that when it comes to straightforward queries with a modest amount of overhead, the performance of BP++ is noticeably superior to that of HadoopDB. BP++ is a system that delivers elastic data sharing services by integrating cloud computing, databases, and peer-to-peer technologies [6]. It was developed in response to the unique challenges that are presented by the act of sharing and processing data in an environment that involves multiple businesses. Our technology is able to effectively manage the common workloads that occur inside a business network and can provide nearly linear query performance even as the number of regular peers increases. The pay-as-you-go business model, which was made popular by cloud computing, has been adopted by BP++. As a result, there is a significant reduction in the total cost of ownership since businesses are spared the expense of purchasing any hardware or software in advance. Instead, users pay just for the hours and capacity of storage that their BP++ instance consumes. The role-based access control for the naturally dispersed environment that is present in business networks may be expanded using BP++. P2P technology is used by BP++ in order to facilitate the retrieval of data between various business partners. BP++ is a potentially useful solution for the effective exchange of data within the context of business networks.

IV. ENHANCED SYSTEM:

Basic processing and adaptive processing are the two methods that BP++ uses to handle query processing, respectively. The fundamental query processing approach is rather similar to the one used in the area of distributed databases. In general, the query that is sent to a regular peer P is assessed in two stages: the first stage is called "fetching," and the second stage is "processing." During the stage known as "fetching," the question is broken down into a series of sub-queries. These sub-queries are then sent to the distant normal peers that house the data that is relevant to the query (the list of these normal peers is determined by searching the indices stored in BATON). After that, the sub query is handled by each distant normal peer, and the intermediate results are shuffled back to the peer that submitted the query, which is P. In the initial phase of the processing stage, the normal peer P is in charge of gathering all of the necessary information from the other normal peers who are participating. The peer P produces a collection of memory tables to keep the data that is obtained from other peers. When a Mem Table is full, the peer P bulk writes this data into the local MySQL database. This helps to limit the amount of I/O that is required. The submitted query is eventually evaluated by the peer P when they have received all of the data that is required. We distribute the tuples for each join among a group of processing nodes so that the join may be processed in parallel on those nodes rather than sending all of the tuples at once to a single processing node. We eventually settle on the traditional replicated join methodology. To be more specific, a partition of the big table will be merged with the tiny table when it has been replicated to all of the processing nodes. The join processing is the primary point of difference between the Map Reduce approach and the native P2P method. With the Map Reduce technique, the replicate joins that are typically performed are replaced with the symmetric-hash join strategy instead. Every mapper pulls in the data from its own local repository before shuffling the intermediate tuple in accordance with the hash value of the join key. Because of this, each tuple only has to be shuffled once across all of the levels. It is important to keep in mind that the setup and launch of a Map Reduce task both incur some overhead, which, as a constant number, may be assessed throughout the duration of the operation. Since it does not have a startup cost and its database connect algorithms have been properly

tuned, the P2P engine performs better than the Map Reduce engine when it comes to the processing of smaller tasks. Nevertheless, the Map Reduce engine is more scalable for large-scale data analysis workloads because it does not suffer recursive data replications. This makes the Map Reduce engine the better choice. Our adaptive query processing strategy is proposed here on the basis of the cost models that were discussed before. The query planner performs an analysis of the query, collects associated histogram and index information from the bootstrap node, and then generates a processing graph for the query. This all happens when a query is submitted. Finally, based on the histograms and runtime parameters of the cost models, predictions are made on the costs associated with both the P2P engine and the Map Reduce engine. The query planner will analyze the costs of the two different approaches and then implement the one that has the lowest overall cost.

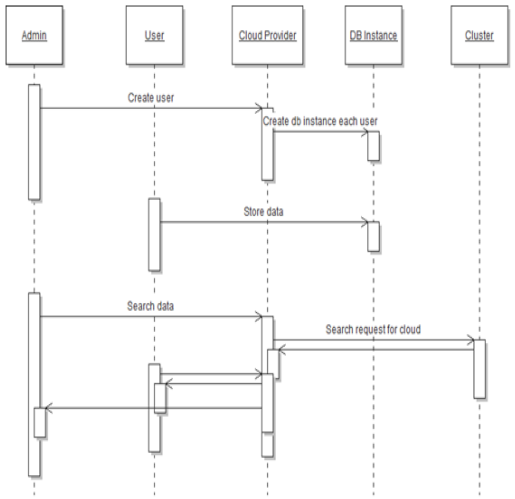

Fig 1: Sequence of System

V. CONCLUSIONS:

We have presented BP++, a system that delivers elastic data sharing services by integrating cloud computing, databases, and peer-to-peer technologies. In addition, we have discussed the one-of-a-kind challenges that are presented by the act of sharing and processing data in an environment that involves multiple businesses. The benchmark that was carried out on the Amazon EC2 cloud platform demonstrates that our system is capable of effectively managing typical workloads within a corporate network and of delivering nearly linear query throughput as the number of normal peers increases. As a result, BP++ is an intriguing candidate for a solution to the problem of ineffective data sharing across corporate networks.

REFERENCES:

[1] 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.

[2] 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.

[3] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: Amazon's Highly Available Key-Value Store," Proc. 21st ACM SIGOPS Symp. Operating Systems Principles (SOSP '07), pp. 205-220, 2007.

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

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

[6] R. Huebsch, J.M. Hellerstein, N. Lanham, B.T. Loo, S. Shenker, and I. Stoica, "Querying the Internet with PIER," Proc. 29th Int'l Conf. Very Large Data Bases, pp. 321-332, 2003.

[7] V. Poosala and Y.E. Ioannidis, "Selectivity Estimation without the Attribute Value Independence Assumption," Proc. 23rd Int'l Conf. Very Large Data Bases (VLDB '97), pp. 486-495, 1997.

[8] M.O. Rabin, "Fingerprinting by Random Polynomials," Technical Report TR-15-81, Harvard Aiken Computational Laboratory, 1981.

[9] E. Rahm and P. Bernstein, "A Survey of Approaches to Automatic Schema Matching," The VLDB J., vol. 10, no. 4, pp. 334-350, 2001.

[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.

