ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

BestPeer++: A Peer-to-Peer based Large-scale Data Processing Platform

Chatta Anilkumar
M.Tech Student
Sir C.V.Raman Institute of Technology and Sciences

Ch.Balaji
Guide
Sir C.V.Raman Institute of Technology and Sciences

ABSTRACT

The information sharing among the corporate companies is done with corporate network and facilitates collaboration in a certain industry sector where companies share a common interest. It does help the companies to reduce their costs and increase the revenues. However, the inter-company data sharing and processing poses unique challenges to such a data management system including scalability, performance, throughput, and security. Here, we present BestPeer++, a system which delivers elastic data sharing services for corporate network applications in the cloud based on BestPeer – a peer-to-peer (P2P) based data management platform.

BestPeer++ provides an economical, flexible and scalable platform for corporate network applications and delivers data sharing services to participants based on the widely accepted pay-as-you-go business model, by integrating cloud computing, database, and P2P technologies into one system. We evaluate BestPeer++ on Amazon EC2 Cloud platform. The benchmarking results show that BestPeer++ outperforms HadoopDB, a recently proposed large-scale data processing system, in performance when both systems are employed to handle typical corporate network workloads. The benchmarking results also demonstrate that BestPeer++ achieves near linear scalability for throughput with respect to the number of peer nodes.

INTRODUCTION

Different companies of same sector are often connected to a corporate network to collaborate with one another; each company maintains its own site and selectively shares a portion of its business data with the others. 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.

First, the corporate network needs to scale up to support thousands of participants, while the installation of a largescale 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. Second, companies want to fully customize the access control policy to determine which business partners can see which part of their shared data.

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. To address the aforementioned problems, this paper presents BestPeer++, a cloud enabled data sharing platform designed for corporate network applications. By integrating cloud computing, database, and peer-to-peer (P2P) technologies, Best-Peer++ achieves its query processing efficiency and 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 simply register their sites with the BestPeer++ service

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

provider, launch BestPeer++ instances in the cloud and finally export data to those instances for sharing. BestPeer++ adopts the pay-as-you-go business model popularized by cloud computing. The total cost of ownership is therefore substantially reduced since companies do not have to buy any hardware/software in advance. Instead, they pay for what they use in terms of BestPeer++ instance's hours and storage capacity. The BestPeer++ service provider elastically scales up the running instances and makes them always available. Therefore, companies can use the ROI driven approach to progressively invest on the data sharing system.

- BestPeer++ 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

- BestPeer++ employs P2P technology to retrieve data between business partners. BestPeer++ instances are organized as a structured P2P overlay network named BATON. The data are indexed by the table name, column name and data range for efficient retrieval.

- 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. BestPeer++ is mainly optimized for these queries. For infrequent time-consuming analytical tasks, we provide an interface for exporting the data from BestPeer++ to Hadoop and allow users to analyze those data using MapReduce.

OVERVIEW OF THE BESTPEER++ SYSTEM

While traditional P2P network has not been designed for enterprise applications, the ultimate goal of BestPeer is to bring the state-of-art database techniques into P2P systems. In its early stage, Best-Peer employs unstructured network and information retrieval technique to match columns of different tables automatically. After defining the mapping functions, queries can be sent to different nodes for processing. In its second stage, BestPeer introduces a series of techniques for improving query performance and result quality to enhance its suitability for corporate network applications. In particular, BestPeer provides efficient distributed search services with a balanced tree structured overlay network and partial indexing scheme for reducing the index size.

In the last stage of its evolution, BestPeer++ is enhanced with distributed access control, multiple types of indexes, and pay-as-you-go query processing for delivering elastic data sharing services in the cloud. The software components of BestPeer++ are separated into two parts: core and adapter. The core contains all the data sharing functionalities and is designed to be platform independent. 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).

Amazon Cloud Adapter

The Amazon Cloud Adapter provides an elastic hardware infrastructure for BestPeer++ to operate on by using Amazon Cloud services. We use Amazon EC2 service to provision the database server. Each time a new business joins the BestPeer++ network, we launch a dedicated EC2 virtual server for that business. The newly launched virtual server (called a BestPeer++ instance) runs 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 to back up and scale each BestPeer++ instance, we do use Amazon Relational Data Service (RDS). The whole MySQL database is backed up to Amazon's reliable EBS

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

storage devices in a four minute window. There will be no service interrupt during the process since the backup operation is performed asynchronously.

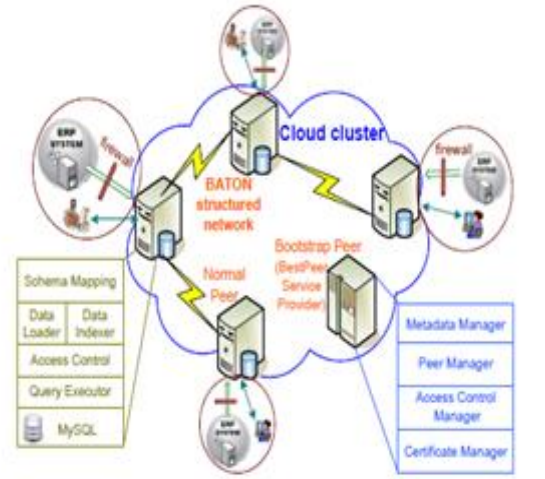

Fig. 1. The BestPeer++ network deployed on Amazon Cloud offering

The BestPeer++ Core

It contains all platform-independent logic, including query processing and P2P overlay. It runs on top of adapter and consists of two software components: bootstrap peer and normal peer. A BestPeer++ network can only have a single bootstrap peer instance which is always launched and maintained by the BestPeer++ service provider and a set of normal peer instances. The architecture is depicted in Figure 1.

The entry point for whole network is bootstrap peer. First it serves for various administration purposes, including monitoring and managing normal peers registration and also scheduling various network management events, then it acts as acts as a central repository for storing meta data of corporate network applications, including shared global schema, participant normal peer list, and role definitions. In addition, BestPeer++ employs the standard PKI encryption scheme to encrypt/decrypt data transmitted between normal peers in order to further increase the security of the system.

BOOTSTRAP PEER

is run by the BestPeer++ service provider, and its main functionality is to manage the Best- Peer++ network.

Managing Normal Peer Join/Departure

All normal peers must connect to bootsrap first, (in order to connect to corporate network). If the servce provider did accept the join request, then the bootstrap peer will put the newly joined peer into the peer list of the corporate network.

At the same time, the joined peer will receive the corporate network information including the current participants, global schema, role definitions, and an issued certificate. When the normal peer needs to leave the network, it will also notify the bootstrap peer. The bootstrap peer will put the departure peer on the black list and mark the certificate of the departing peer invalid. Then, the bootstrap peer will release all resources allocated for the departing peer back to the cloud and finally remove the departing peer from the peer list.

Auto Fail-over and Auto-Scaling

The bootstrap peer is also responsible for monitoring the health of normal peers and scheduling fail-over and auto-scaling events. If some peers are malfunctioned or crashed, the bootstrap peer will trigger an automatic fail-over event for each failed normal peer. The automatic fail-over is performed by first launching a new instance from the cloud. Then, the bootstrap peer asks the newly launched instance to perform database recovery from the latest database backup stored in Amazon EBS. Finally, the failed peer is put into the blacklist.

NORMAL PEER

There are two data flows inside the normal peer: an offline data flow and an online data flow and is shown in figure 2. In the offline data flow, the data are extracted periodically by a data loader from the business production system to the normal peer instance. In particular, the data loader extracts the data from the business production system, transforms the data format from its local schema to the shared global schema of the corporate network according to the schema mapping, and finally stores the results in the MySQL databases hosted in the normal peer.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Schema Mapping

It is a component that defines the mapping between the local schemas employed by the production system of each business and the global shared schema employed by the corporate network. Currently, BestPeer++ only supports relational schema mapping, namely both local schema and the global schema are relational. In general, the schema mapping process requires human to be involved and is time consuming. However, it only needs to perform once. Furthermore, Best-Peer++ adopts templates to facilitate the mapping process. For each popular production system (i.e., SAP or PeopleSoft), we provide a mapping template which defines the transformation of local schema of those systems to the global schema. The business only needs to modify the mapping template to meet its own needs.

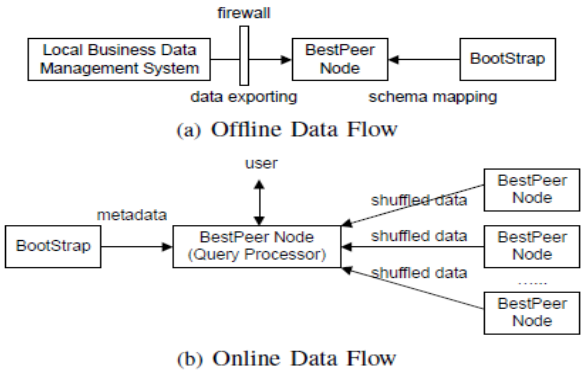

Fig. 2. Data Flow in BestPeer++

Data Loader

Data Loader is a component that extracts data from production systems to normal peer instances according to the schema mappings. While the process of extracting and transforming data is straightforward, the main challenge is in maintaining the consistency between raw data stored in the production systems and extracted data stored in the normal peer instance (and subsequently data indices created from these extracted data) when the raw data are updated inside the production systems.

When the data loader first extracts data from the production system, besides storing the results in the normal peer instance, the data loader also creates a snapshot of the newly inserted data 3. After that, at interval times, the data loader re-extracts data from the production system to create a new snapshot. This snapshot is then compared to the previously stored snapshot to detect data changes. Finally, the changes are used to update the MySQL database hosted in the normal peer.

Data Indexer

Here, the data are stored in the local MySQL database hosted by each normal peer. Thus, to process a query, we need to locate which normal peers host the tables involved in the query. For example, to process a simple query like select R.a from R where R.b=x, we need to know which peers store tuples belonging to the global table R. We adopt the peer-to-peer technology to solve the data locating problem and only send queries to normal peers which host related data. In particular, we employ BATON, a balanced binary tree overlay protocol to organize all normal peers.

Figure 3 shows the structure of BATON. In BestPeer++, the interface of BATON is abstracted as Table I. We provide three ways to locate data required for query evaluation: table index, column index, and range index. Each of them is designed for a separate purpose. Table II summarizes the index formats in BestPeer++. In query processing, the priorities of indices are (Range Index>Column Index>Table Index). We will use the more accurate index whenever possible

TABLE I
BATON INTERFACE

`join(P)`	Join the network
`leave(P)`	Leave the network
`put(k, v)`	Insert a key-value pair into the network
`remove(k, v)`	Delete the value with the key
`get(k)`	Retrieve the value with the single key
`get(begin, end)`	Retrieve values with the key range

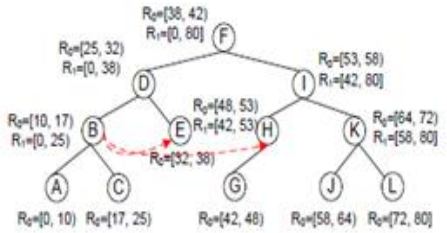

Fig. 3. BATON Overlay

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

TABLE II
INDEX FORMAT SUMMARIES

Type	Key	Indexed Value
Table Index	Table Name	A normal peer list
Column Index	Column Name	A list of peer-table pairs
Range Index	Table Name	A list of column-range pairs

PAY-AS-YOU-GO QUERY PROCESSING

Two services are provided by the BestPeer++ provides. The storage service and search service, both of which are charged in a pay-as-you-go model. This section presents the payas- you-go query processing module which offers an optimal performance within the user's budget.

Definition: Let ***T*** denote the QoS set by the user. The query latency must be less than ***T*** seconds with high probability. BestPeer++ generates a plan to minimize C_{extend}, while guaranteeing that $T_{extend} < T$.

BestPeer++'s query engine iterates possible query plans and selects the optimal one. The iteration algorithm is similar to the one used in conventional DBMS and thus is not repeated in the paper. The intuition of the query engine is to exploit the parallelism to meet the QoS and reduce the cost as much as possible.

BENCHMARKING

Performance Benchmarking

Comparison of the performance of BestPeer++ with HadoopDB can be done with this benchmark. We consider HadoopDB as our benchmark target since it is an alternative promising solution for our problem and adopts architecture similar to ours. Comparing the two systems (i.e., HadoopDB and BestPeer++) reveals the performance gap between a general data warehousing system and a data sharing system specially designed for corporate network applications.

1) Benchmark Environment: We run our experiments on Amazon m1.small DB instances launched in the ap-southeast-1 region. Each DB small instance has 1.7GB 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 30MB/sec to 120MB/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 90MB/sec during our benchmark. The end-to-end network bandwidth between DB small instances, measured by iperf, is approximately 100MB/sec.

2) BestPeer++ Settings: The configuration of a BestPeer++ normal peer consists of two parts: the underlying MySQL database server and the BestPeer++ software. For MySQL database, we use the default MyISAM storage engine which is optimized for read-only queries since no transactional processing overhead is introduced. We set up a large index memory buffer (500MB) and the maximum number of tables to be concurrently opened (50 tables). For BestPeer++ software, we set the maximum memory consumed by the MemTable to be 100MB. We also configure each normal peer to use 20 concurrent threads for fetching data from remote peers. Finally, we configure each normal peer to use the basic query processing strategy.

3) HadoopDB Settings: We carefully follow the instructions presented in the original HadoopDB paper to configure HadoopDB. The setting consists of the setup of a Hadoop cluster and the PostgreSQL database 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 256MB. 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 1024MB. The buffer size of read/write operations is set to 128KB. We also set the sort buffer of the map task to 512MB 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.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

4) Data loading: this process is performed by all normal peers in parallel and consists of two steps. In step 1, each normal peer invokes the data loader to load raw TPC-H data into the local MySQL databases. In addition to copying raw data, we also build indices to speedup query processing. First, we build a primary index for each TPC-H table on the primary key of that table. Second, we build additional secondary indices on selected columns of TPC-H tables. Table III summarizes the secondary indices that we built. After the data is loaded into the local MySQL database, each normal peer 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.

TABLE III
SECONDARY INDICES FOR TPC-H TABLES

LineItem	l_shipdate, l_commitdate, l_receiptdate
Orders	o_custkey, o_orderpriority, o_orderdate
Customer	c_mktsegment
PartSupp	ps_supplycost
Part	p_brand, p_type, p_size, p_mfgr

5) The Q1 Query Results: The first benchmark query Q1 evaluates a simple selection predicate on the **l_shipdate** and **l_commitdate** attributes from the **LineItem** table. The predicate yields approximately 3,000 tuples per normal peer.

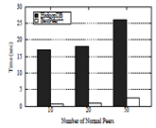
Fig. 5. Results for Q1

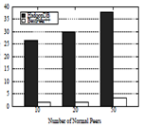
Fig. 6. Results for Q2

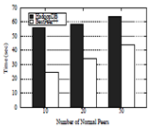
Fig. 7. Results for Q3

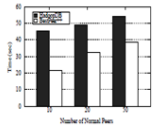
Fig. 8. Results for Q4

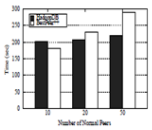
Fig. 9. Results for Q5

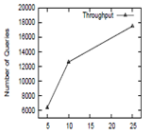
Fig. 10. Throughput of Suppliers

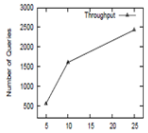
Fig. 11. Throughput of Retailers

Both systems (HadoopDB and BestPeer++) perform this query within a short time. This is because both systems benefit from the secondary indices built on l shipdate and l commitdate columns. However, the performance of BestPeer++ is significantly better than HadoopDB. The performance gap between HadoopDB and BestPeer++ is attributed to the startup costs of MapReduce job introduced by the Hadoop layer, including the cost of scheduling map tasks on available task tracker nodes and the cost of launching a fresh new Java process on each task tracker node to perform the map task.

Figure 6 shows the performance of each benchmarked system. BestPeer++ still outperforms HadoopDB by a factor of ten. The performance gap between HadoopDB and BestPeer++ comes from two factors. First, the startup costs introduced by Hadoop layer still dominates the execution time of HadoopDB. Second, Hadoop (and generally MapReduce) employs a pull based method to transfer intermediate data between map tasks and reduce tasks. The reduce task must periodically queries the job tracker for the map completion events and start to pull data after it has retrieved these completion events. We observe that, in Hadoop, there is a noticeable delay between the time point of map completion and the time point of those completion events being retrieved by the reduce task. Such delay slows down the query processing.

Figure 7 presents the performance of both systems. From Figure 7, we can see that the performance gap between BestPeer++ and HadoopDB becomes smaller. This is because this query requires processing more tuples than previous queries. Therefore, the Hadoop startup costs are amortized by the increased workload. We also see that as the number of nodes grows, the scalability of HadoopDB is slightly better than BestPeer++. Figure 8 presents the performance of both systems. We can see that BestPeer++ still outperforms HadoopDB. But the performance gaps between the two systems are much smaller. Also, HadoopDB achieves better scalability than BestPeer++.

This is because HadoopDB can benefit from parallelism by distributing the join and aggregation processing among worker nodes. However, to achieve that, we must manually set the number of reducers to be equal to the number of worker nodes. BestPeer++, on the other hand, only performs the join and the final aggregation at the query submitting peer. As more

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

nodes are involved, more data need to be processed at the query submitting peer, resulting in that peer to be over-loaded. Again, the performance problem of BestPeer++ can be mitigated by upgrading the normal peer to a larger instance. Figure 9 presents the results of this benchmark. Overall, HadoopDB performs better than BestPeer++ in evaluating this query. The fetching phase of BestPeer++ dominates the query processing since it needs to fetch all qualified tuples to the query submitting peer. HadoopDB, however, can utilize multiple reducers for transferring the data in parallel.

Figure 10 and Figure 11 present the results of this benchmark. For each setting, the results are presented in separate figures for suppliers and retailers, respectively (e.g., in a 50 node cluster, we have 25 supplier peers and 25 retailer peers). We can see that BestPeer++ achieves near linear scalability in both heavy-weight workloads (i.e., retailer queries) and lightweight workloads (i.e., supplier queries). The main reason for this is that BestPeer++ adopts a single peer optimization. In our benchmark, all queries will only touch just one normal peer. In the peer searching phase, if the query executor finds that a single normal peer hosts all required data, the query executor employs the single peer optimization and sends the entire SQL to that normal peer for execution. The results returned by that normal peer are directly sent back to the user. The final processing phase is entirely skipped.

CONCLUSION

Here, we proposed BestPeer++ a system which delivers elastic data sharing services, by integrating cloud computing, database, and peer-to-peer technologies and have discussed the unique challenges posed by sharing and processing data in an inter-businesses environment. The benchmark conducted on Amazon EC2 cloud platform proves that our system can deliver near linear query throughput as the number of normal peers grows, and thus can efficiently handle typical workloads in a corporate network. Therefore, BestPeer++ is a suitable solution for efficient data sharing within corporate networks.

References

[1] K. Aberer, A. Datta, and M. Hauswirth. Route Maintenance Overheads in DHT Overlays. In The 6th Workshop on Distributed Data and Structures, 2004.

[2] 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. PVLDB, 2(1):922–933, 2009.

[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. In SOSP, pages 205–220, 2007.

[4] H. Garcia-Molina and W. J. Labio. Efficient Snapshot Differential Algorithms for Data Warehousing. Technical report, Stanford, CA, USA, 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. In VLDB, pages 321–332, 2003.

[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. In SIGMOD, 2006.

[8] H. V. Jagadish, B. C. Ooi, K.-L. Tan, C. Yu, and R. Zhang. iDistance: An adaptive b+-tree based indexing method for nearest neighbor search. ACM Trans. Database Syst., 30:364–397, June 2005.

[9] H. V. Jagadish, B. C. Ooi, and Q. H. Vu. BATON: A Balanced Tree Structure for Peer-to-Peer Networks. In VLDB, pages 661–672, 2005.

[10] A. Lakshman and P. Malik. Cassandra: structured storage system on a p2p network. In PODC, pages 5–5, 2009.

[11] W. S. Ng, B. C. Ooi, K.-L. Tan, and A. Zhou. PeerDB: A P2P-based System for Distributed Data Sharing. In ICDE, pages 633–644, 2003.

[12] Oracle Inc. Achieving the Cloud Computing Vision. White Paper, 2010.

[13] V. Poosala and Y. E. Ioannidis. Selectivity estimation without the attribute value independence assumption. In VLDB, pages 486–495, 1997.

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

[15] P. Rodr´ıguez-Gianolli, M. Garzetti, L. Jiang, A. Kementsietsidis, I. Kiringa, M. Masud, R. J. Miller, and J. Mylopoulos. Data Sharing in the Hyperion Peer Database System. In VLDB, pages 1291–1294, 2005.

[16] Saepio Technologies Inc. The Enterprise Marketing Management Strategy Guide. White Paper, 2010.

[17] I. Tatarinov, Z. G. Ives, J. Madhavan, A. Y. Halevy, D. Suciu, N. N. Dalvi, X. Dong, Y. Kadiyska, G. Miklau, and P. Mork. The piazza peer data management project. SIGMOD Record, 32(3):47–52, 2003.

[18] H. T. Vo, C. Chen, and B. C. Ooi. Towards elastic transactional cloud storage with range query support. PVLDB, 3(1):506–517, 2010.

[19] S. Wu, S. Jiang, B. C. Ooi, and K.-L. Tan. Distributed online aggregation. PVLDB, 2(1):443–454, 2009.

[20] S. Wu, J. Li, B. C. Ooi, and K.-L. Tan. Just-in-time query retrieval over partially indexed data on structured p2p overlays. In SIGMOD, pages 279–290, 2008.