Cloud Data Storage and Query Processing Systems: A Review

Imran Ashraf, Tayyaba Altaf, Ayesha Rashid

Abstract -With massive increase in data size, databases have grown beyond size that can be managed at a central point. So databases need to be distributed with capabilities of scaling, aggregation and coordination. With the advent of cloud computing resources are easy than ever to access with the possibility of scalability, increased performance, automatic resource adaptation and up and down scaling. However, suitable distributed management systems need to be developed to get the desired benefits from the cloud. Effective and efficient execution of users complicated queries is a critical task of systems in cloud computing. Distributed data storage and query systems in cloud computing needs to insure two things: efficiently organized data storage and optimized query execution with reduced cost. Different systems have been developed and deployed in cloud computing for this purpose. This paper aims at discussing some most commonly used systems with their pros and cons.

Index Terms: Cloud computing, distributed storage systems, distributed query systems, scalability, load balancing, replication

1 Introduction

Databases have outgrown what single server systems can manage and they are still in the process of further growth. In order to fulfil the future demands of data management and data retrieval distributed database management systems need to be developed. These DBMSs should be able to scale accordingly; however scaling is not an easy task. In data distribution, coordination is needed which creates an overhead for the systems. This overhead grows as the system expands and this restricts the capacity of system's scalability. One solution to this scalability problem is the nodes or sites automation in distributed systems. Systems autonomy can decrease the coordination overhead. Moreover, distributed systems need to ensure two things data storage and query processing. These systems should be able to provide the facilities of efficient data management and optimized query processing. With the concept of cloud computing resource access and usage has become very easy and cheap. Now, it is easy to scale according to user requirements of up or down grading. Distributed systems come with the option of varying data placement. The idea of grid or cloud is that system should be able to scale with the load and storage requirements. It means that system should adapt automatically if workload or sites available for storage and query processing changes. Different systems have been developed that meets cloud's requirements of scalability, coordination and load balancing [1, 2]. Each of these systems provides its own functionality of scalability, load balancing, replication, concurrency etc. This review aims at discussing these systems' architecture, execution, and pros and cons.

- *Imran Ashraf is currently working as Lecturer at Information Technology Department of, University of The Punjab, Gujranwala, Pakistan. PH-00923456473032. E-mail: ashrafimran@live.com*
- *Tayyaba Altaf & Ayesha Rashid are serving are pursuing MS program in CS in NCBA&E and University of Gujrat, Pakistan respectively.*

2 Distributed Data Storage and Query Processing Systems

2.1 Piazza

The Piazza system [3, 4] is a peer-t-peer data management system for integrating data sources with heterogeneous schemas. Instead of requiring all sites to share a common schema, e.g. as PIER does, Piazza mediates between these schema. Both unstructured and super node overlay networks can be used but sites are not free to connect o any other site. A connection between two sites implies a partial or full schema mapping between the sites. Creating such schema mappings are heavy weight operations, which means that new connections are not added frequently. Because of this the system is assumed to have a low churn level with very few sites joining or leaving.

The focus of the Piazza project has been schema mediation and query reformulation according to schema mappings. Piazza sites store data in XML and issue queries in language inspired by XQuery. Queries that are sent over the network are reformulated to reject schema mappings as they propagate from one site to another. The result of a query is similarly rewritten to new schemas as it is passed back to the querying site.

Replication is possible in Piazza. Piazza storage description defines what content a site should store and this description can define parts of other sites' databases that should be replicated locally. Like network connections, update to this description is not expected to be a very frequent operation and hence not automated.

2.2 PeerDB

PeerDB [5-7] is a peer-to-peer relational DBMS using mobile agents in query processing. It is built on the BestPeer platform [8], which is a system for mobile agents in peer-to-peer networks. BestPeer uses a super node network and PeerDB combines this with the MySQL DBMS for data storage. Sites in a PeerDB system can have heterogeneous schemas. Each schema is described by keywords and query processing uses standard information retrieval techniques to send probable candidates for matching tables.

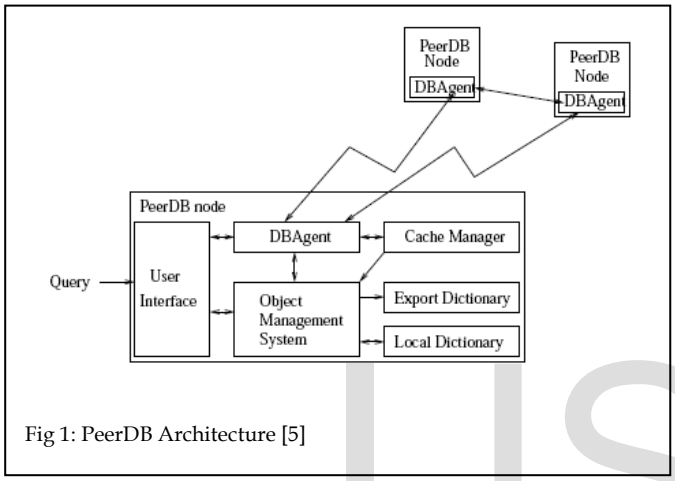

Fig 1: PeerDB Architecture [5]

When a query is issued, it is first parsed to extract a table and attribute names. These names are looked up in a local dictionary to send matched in the local database. Agents are also shipped onto neighbouring sites to send matching tables and attributes there. Possible matches are returned to the querying site where the user selects which tables are to be included in the final answer. After the user has made this choice the agents on the matching sites rewrite the query to match the local schema and return tuples a reply to the query.

2.3 AmbientDB

AmbientDB [9] uses a DHT to provide a self organizing peer-to-peer network of intelligent home appliances. These devices share information using database system. The DHT serve both as a way to connect he devices and as an indexing mechanism for data tree in the database.

The challenges facing AmbientDB include mobile devices with limited networking bandwidth and computing resources. The devices may also disconnect frequently and stay disconnected or long periods of time. AmbientDB provides database service both for single devices that are currently away from the rest of the network and for the rest of the network. Device that are able to connect to other devices nearby can access each other's data, and synchronize by updating data items that have been updated while the devices where disconnected.

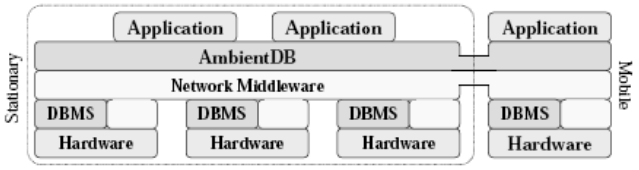

Figure: AmbientDB environment [9]

The self organizing property f AmbientDB makes it possible for devices to extend the database schema and data propagation strategies e.g. a thermometer may extend the database schema to include temperature recordings that can be used by other devices to automatically adjust to the surroundings.

2.4 APPA

The Atlas Peer-to-peer Architecture (APPA) [10] built on a super node or structured overlay network and provides a full stack of data storage and querying services.

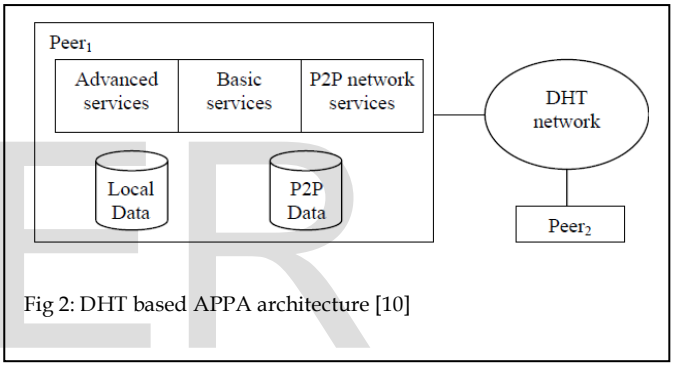

Fig 2: DHT based APPA architecture [10]

The APPA system consists of multiple layers. On the bottom layer, APPA provides a simple key value store, and higher level layers build more advance service.

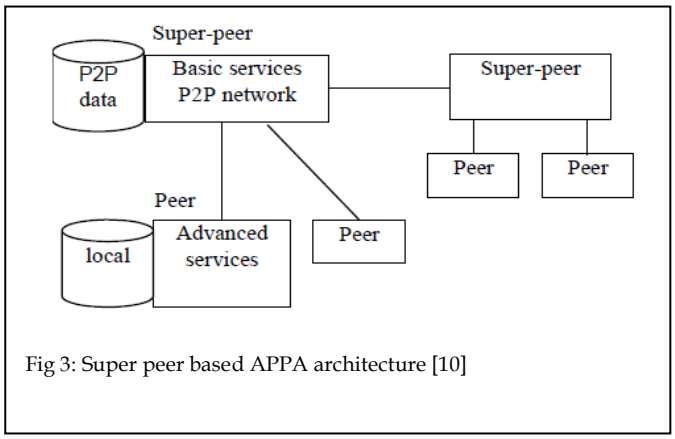

Fig 3: Super peer based APPA architecture [10]

On the op level is service such as schema management, replication and query processing. Schema mapping in APPA

is different from the pair wise schema mappings in Piaza. In APPA site agree on a common global schema and express their tables as view of this schema. Querying is done against the local view, and this query is reformulated against the global schema before the set of sites with data relevant to the query are found.

2.5 Mariposa

Mariposa [11] is a distributed DBMS that uses economic models to solve optimization problems. Table fragments are considered a resource and are bought and sold during a bidding process that is used to decide which sites should participate in processing a given query.

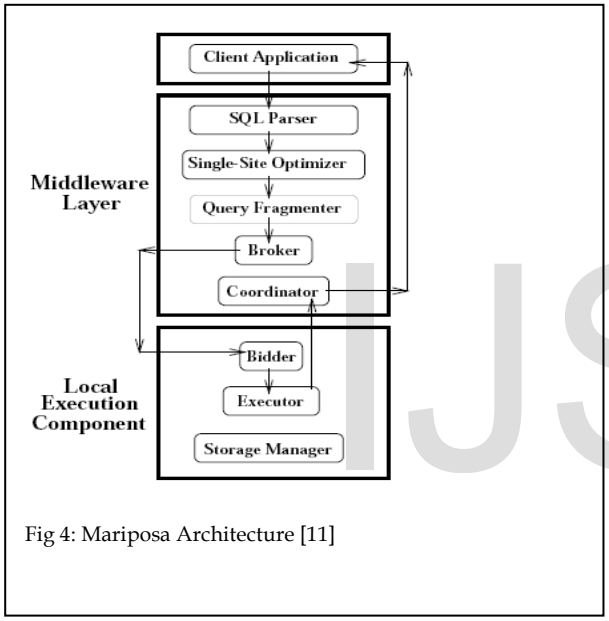

Fig 4: Mariposa Architecture [11]

A new query is given a budget to use on query processing. The sites bid for the execution of this query and the economy is constructed in such a way that the more expensive query plans are more efficient. This means that a query that has been gin a large budget is prioritized and can buy a more efficient execution. This bidding process includes the buying and selling of table fragments in order to move data closer to the processing sites.

2.6 ObjetGlobe

ObjectGlobe [12, 13] is a distributed storage and query processing system where data, query operators and computing power is traded. Each site can offer a combination of data sets, query operators and computing power. A site that wants to execute a query combines components from several other sites and buys computing resources to process the query. A pipeline is constructed that ships intermediate results from one operator to another.

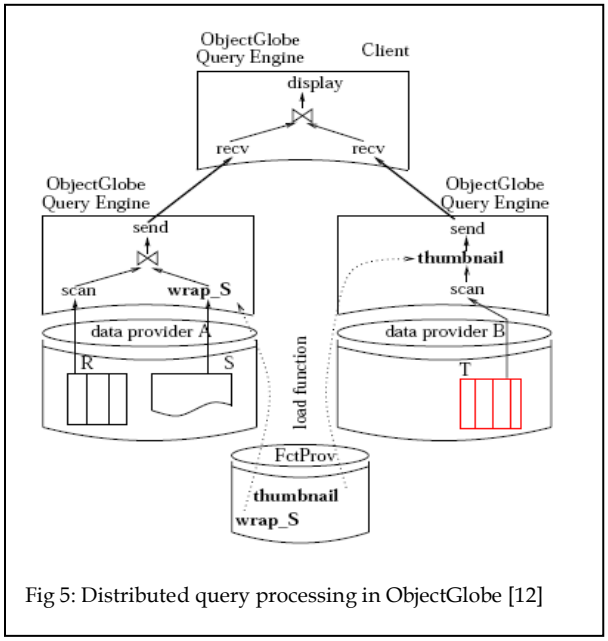

Fig 5: Distributed query processing in ObjectGlobe [12]

The difference between ObjectGlobe and Mariposa is that while Mariposa used the economic model to improve querying performance, ObjectGlobe has a more direct connection to a real economy, where data sets, operator implementations and computing power are actually sold.

2.7 HadoopDB

HadoopDB [14] is a database middleware system for the cloud. It is built on top of Hadoop, an implementation of Map Reduce [15]. Each site in the system has local DBMS that manages storage and these sites are connected by the Hadoop framework. The local database is integrated with the Map Reduce framework so that it can be accessed similarly to Hadoop's own distributed file system, HDFS. A metadata catalogue containing information about local DBMSs, data sets, partitioning and replication is stored in HDFS.

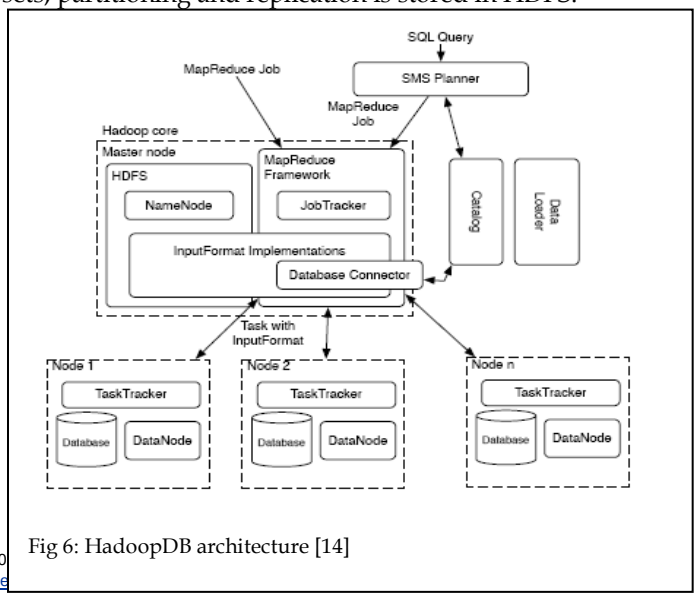

Fig 6: HadoopDB architecture [14]

The Hadoop framework is used to coordinate tasks and to distribute query processing, and its failure resilience properties are used to achieve fault tolerance. A modified implementation of Hive [16] is used to transform a query from the SQL like query language to Map Reduce tasks. Query plans are constructed so that as much query processing as possible is done in the local DBMS on each site. By doing this, HadoopDB is able to maintain the failure resilience properties of Hadoop and much of the query optimization of the DBMS.

2.8 MongoDB

MongoDB [17] is document-oriented like CouchDB, but provides a much more advanced querying system. The basic data unit is a semi-structured document and documents are grouped into collections. Typically, there is a collection for each document type. Data can be fragmented and replicated to increase availability and performance. MongoDB also provides automatic load balancing for query load and data distribution. Queries are posed n an imperative language by defining filters for a scan over a document collection and B-tree indices are used to increase performance.

2.9 VoltDB

A more traditional relational DBMSs is provided by VoltDB [18, 19].VoltDB is a distributed main memory DBMS that uses a traditional relational data model with schemas defined in SQL. However, it is not as traditional when it comes to querying.

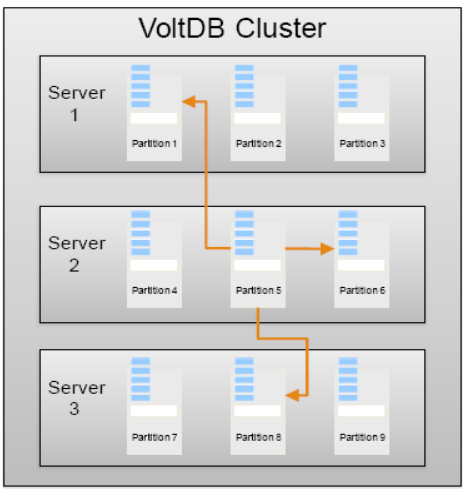

Fig 7: Cluster architecture [19]

Each site runs a single threaded server process. Because of single threading no locking is used and no concurrency issues occur. All access to the table is via stored procedures that constitute a micro transaction. Being a main memory DBMS, replication is used to provide data persistence. Active replication to other site in the same data centre is issued to protect against larger outages. When user needs data from multiple sites one site acts as a coordinator. It distributes the work, collects the result and returns it. Integrity is ensured and parallel partitions increases throughput.

3 Discussions and Conclusion

The discussed systems provide both data storage and query processing facilities for the cloud computing. Each system has been designed in a unique way to provide these facilities. These systems are not compared in the sense of being superior to one another. The chief aim is to describe how they work and support user needs of storage and query processing. These systems are most commonly used systems in order to provide the facilities of scalability, automatic load balancing, dynamic adaptability, data integration, query optimization, schema formulation, replication, fault tolerance, availability and durability. Their features and weaknesses are discussed in points in the given table. The future work is to do a performance evaluation of these systems with experiment.

TABLE 1
FEATURES AND WEAKNESSES OF DATA STORAGE & QUERY PROCESSING SYSTEMS

Attribute / System	Features	Weaknesses
AmbientDB	• Based on DHT • Self organizing • Suitable for networks with limited bandwidth	• Sites disconnect frequently • System may be offline for long periods of time
APPA	• Built on structured overlay network • Comprised of multiple layers; layers provide both low level and high level services • Replication supported • Has a common global schema	• Overhead due to both local query execution and its reformulation for global schema
HadoopDB	• Based on HDFS • Each site has a local database • Replication supported • Fault tolerant • Hive is used to transform SQL query to Map Reduce	• System is overhead due to frequent checkpointing, runtime scheduling and block-level restarting.
Mariposa	• Use economic models for optimization • Bidding is used for query performance	• Bidding process may be expensive
MongoDB	• Documented oriented • Basic unit is semi-structured document • Fragmentation and replication supported • Automatic load balancing • Uses B-tree indices for performance	• Increased number of connections or write requests seriously affects the performance of MongoDB
ObjectGlobe	• Bidding is done on data sets, operator implementations and computing power for optimization	• Increased cost due to dynamic extensibility
PeerDB	• Relational DBMS • Built on BestPeer platform • Supports heterogeneous schema •	• Overhead due to retrieval and user's selection of attribute list
Piazza	• For heterogeneous schemas • May use both structured and super node overlay networks • Stores data in XML • Queries are in a XQuery type • Replication supported	• Sites are not free to connect to any other site • Does heavy weight operations • New connections are not connected frequently • Have low churn level • Updates are not frequent
VoltDB	• Traditional relational DBMS • Main memory DBMS • Queries are SQL based • Single threading • Replication supported • No concurrency issues • High availability • ACID guarantee • High durability	• No multi threading • All access is via stored procedures

REFERENCES

[1] R. Agrawal, A. Ailamaki, P. A. Bernstein, E. A. Brewer, M. J. Carey, S. Chaudhuri, *et al.*, "The Claremont report on database research," *ACM SIGMOD Record,* vol. 37, pp. 9-19, 2008.

[2] K. Tsakalozos, H. Kllapi, E. Sitaridi, M. Roussopoulos, D. Paparas, and A. Delis, "Flexible use of cloud resources through profit maximization and price discrimination," in *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, 2011, pp. 75-86.

[3] A. Y. Halevy, Z. G. Ives, J. Madhavan, P. Mork, D. Suciu, and I. Tatarinov, "The piazza peer data management system," *Knowledge and Data Engineering, IEEE Transactions on,* vol. 16, pp. 787-798, 2004.

[4] I. Tatarinov, Z. Ives, J. Madhavan, A. Halevy, D. Suciu, N. Dalvi, *et al.*, "The Piazza peer data management project," *ACM Sigmod Record,* vol. 32, pp. 47-52, 2003.

[5] W. S. Ng, B. C. Ooi, K.-L. Tan, and A. Zhou, "PeerDB: A P2P-based system for distributed data sharing," in *Data Engineering, 2003. Proceedings. 19th International Conference on*, 2003, pp. 633-644.

[6] B. C. Ooi, Y. Shu, and K.-L. Tan, "Relational data sharing in peer-based data management systems," *ACM SIGMOD Record,* vol. 32, pp. 59-64, 2003.

[7] B. C. Ooi, K.-L. Tan, A. Zhou, C. H. Goh, Y. Li, C. Y. Liau, *et al.*, "PeerDB: peering into personal databases," in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, 2003, pp. 659-659.

[8] W. S. Ng, B. C. Ooi, and K.-L. Tan, "Bestpeer: A self-configurable peer-to-peer system," in *Data Engineering, 2002. Proceedings. 18th International Conference on*, 2002, p. 272.

[9] W. Fontijn and P. Boncz, "AmbientDB: P2P data management middleware for ambient intelligence," in *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, 2004, pp. 203-207.

[10] R. Akbarinia, V. Martins, E. Pacitti, and P. Valduriez, "Design and implementation of Atlas P2P architecture," *Global Data Management,* vol. 8, p. 98, 2006.

[11] M. Stonebraker, P. M. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, *et al.*, "Mariposa: a wide-area distributed database system," *The VLDB Journal,* vol. 5, pp. 48-63, 1996.

[12] R. Braumandl, M. Keidl, A. Kemper, D. Kossmann, S. Seltzsam, and K. Stocker, "Objectglobe: Open distributed query processing services on the internet," *IEEE Data Eng. Bull.,* vol. 24, pp. 64-70, 2001.

[13] R. Braumandl, M. Keidl, A. Kemper, D. Kossmann, A. Kreutz, S. Seltzsam, *et al.*, "ObjectGlobe: Ubiquitous query processing on the Internet," *The VLDB Journal,* vol. 10, pp. 48-71, 2001.

[14] A. Abouzeid, K. Bajda-Pawlikowski, D. Abadi, A. Silberschatz, and A. Rasin, "HadoopDB: an architectural hybrid of MapReduce and DBMS technologies for analytical workloads," *Proceedings of the VLDB Endowment,* vol. 2, pp. 922-933, 2009.

[15] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM,* vol. 51, pp. 107-113, 2008.

[16] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, *et al.*, "Hive: a warehousing solution over a map-reduce framework," *Proceedings of the VLDB Endowment,* vol. 2, pp. 1626-1629, 2009.

[17] M. homepage, "http://www.mongodb.org," ed, 2014.

[18] T. V. homepage, "http://voltdb.com," ed, 2014.

[19] L. VoltDB, "Voltdb technical overview," ed: Whitepaper, 2010.