Balancing Scalability, Performance and Fault Tolerance for Structured Data (BSPF)

Amna Khalid*, Hammad Afzal*, Shoohira Aftab*

*Department of Computer Software Engineering, Military College of Signals, National University of Sciences and Technology, Islamabad, Pakistan.

amna.khalid.nust@gmail.com, hammadafzal@mcs.edu.pk, shoohira@gmail.com

Abstract— **Analytical business applications generate reports that give a trend predicting insight into the organization's future, estimating the financial graphs and risk factors. These applications work on huge amounts of data, which comprises of decades of market and company records, and decision logs of an organization. Today, limit of big data is touching zeta-bytes and the structured data makes only 20% of today's data. 20% of a giga-byte can be ignorable in comparison to big data but 20% of big data itself cannot be neglected. Traditional data management tools are like step-dads when it comes to running cross table analytical queries on structured data in distributed processing environment; response time to these data management tools are high because of the ill-aligned data sets and complex hierarchy of distributed computing environment. Data alignment requires a complete shift in data deployment paradigm from row oriented storage layout to column oriented storage layout, and complex hierarchy of distributed computing environment can be handled by keeping metadata of entire data set. Paper proposes an approach to ease the deployment of structured data into the distributed processing environment by arranging data into column-wise combinational entities. Response time to analytical queries can be lowered with the support of two concepts; Shared architecture and Multi path query execution. Highly scalable systems are Shared Nothing architecture based but degradation in performance and fault tolerance are the side effects that came with high scalability. Proposed method is an effort to balance the equation between scalability, performance and fault tolerance. And due to the limited scope of this paper we concentrate on issues and solutions for structured data only. Shared architecture and active backup helps improving the system's performance by sharing the work-load-per-node. BSPF's clustering methodology sheds the data pressure points to minimize the data loss per node crash.**

Keywords— **Big data, Distributed and Cloud Computing**

I. INTRODUCTION

Data that cannot be represented in the form of a model, pre-defined manner or relational table is called unstructured data [1] like tweets from Twitter or likes from Facebook, whereas structured data is the one that can be modeled and classified in the form of a data model or related tables like relational tables. Data that grows beyond the process capability of a data management tools is called big data. Value of big data is revised on the annual basis as the data management tools keeps on hitting the market with their incredible capacity to handle more and more data. Social media is the source of an enormous accumulation of data called big data, cloud is storing it and providing ways to process it via apps; and this generation and processing of big data seems to be an infinite loop. Structured data make almost 20% [2] of the world's data, it's no doubt a small figure but the concept of organized data is way too strong to die at least in near future. Business organizations work with structured data and with this exponential increase in data the concept of centralized databases is on a death bed. Alternate of centralized database is the concept of distributed databases. Deployment of structured databases in distributed environment was easy but deployment of structured data in distributed environment such that it can be processed by unstructured data management tools in a distributed processing environment was the challenge. Organizations demand high performance, high availability, fault tolerance and scalability all in a single tool. A decade ago when scalable systems were developed for the first time, people found this new technology so glittering that they overlooked the degradation in system performance and fault tolerance. But today, they need that missing performance and fault tolerance back. Scalability was added by introducing two concepts; individual tasking and decoupling of service providing nodes. These two concepts affected system performance and fault tolerance. BSPF converts structured data to a form of data that has pre-executed attribute combinations, which are then replicated for backup, clustered and deployed over various nodes (described in Section III). A query handler and executioner is a part of BSPF that handle queries meant to work on structured data and translates them to equivalent queries that works on the new data set (described in Section IV).

Proposed method has applications in the protocol of handling structured data by scalable data management tools like Hadoop. Hadoop deploys structured data table-wise and stores the inter-table relations as metadata. Tables lie in different nodes of different racks which make the cross-table analytical queries really expensive and resource consuming. BSPF suggests an improvement to these tools to the way they handle structured data in order to decrease the systems response time. A prototype of the proposed system is under-testing and is planned to challenge Hadoop's response time for cross-table analytical query execution.

Rest of the paper has the background and related study in Section II. Section III is the Proposed Algorithm (BSPF), Section IV is description of BSPF in action and Section V is concluding this paper.

II. Background and Related Study

This section introduces terms and concepts used in devising this algorithm.

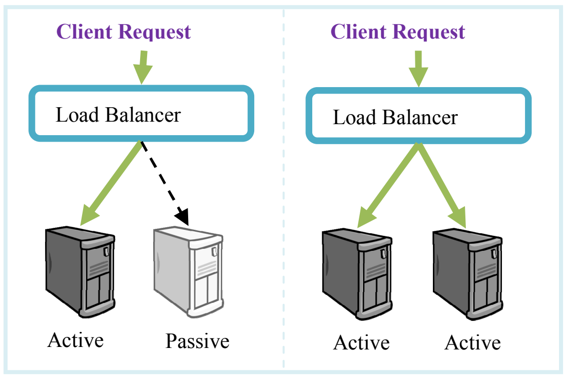


Figure 1. Active-Passive backup and Active-Active backup

Data layout is the way data is structured in a data file or simply stored [3]. There are two basic data layouts; row oriented storage layout and column oriented storage layout. If an entity is a collection of attributes of data, its row oriented; and if a single attribute of data is an entity then it's a column oriented storage layout. Hybrid of the two layouts is also seen in practice.

Applications that measure the performance of business are called analytical applications. These applications are used to produce analytical reports, sometimes used to create predictive analysis; mostly estimating the trends and its ripple propagation in various business layers. Special features of analytical queries are: 1) analytical queries are less predictable, the structure of query changes dynamically as the variable in focus changes 2) analytical queries are mostly read oriented 3) analytical applications focus on attributes instead of entities. In databases functions like 'average', 'count', 'variance' etc are known analytical functions.

A. Active-Passive backup and Active-Active Backup

Primary node is always 'Active' unless it crashes. Traditionally backups are kept 'Passive'. Passive machine only listens for updates and takes over the system control only if the primary system crash. Whereas in active-active scheme, backup shares the work load of the primary system by entertaining queries. If primary crashes, backup takes up the entire work load and vice versa. Figure 1 is graphical comparison of two concepts. Active Passive backup is the traditional approach. [4]

B. Shared Nothing Architecture

A system can have a shared-nothing architecture at various layers. If a network is of shared-nothing architecture at the hardware level then it means that machines in the network are independent of each other, which are not affected by the work load or crash of each other. If two processes are of shared-nothing architecture than it means that they don't have a common memory, processor, socket or port. Crash effect of a process is not rippled to the other process. Similarly, if two clusters or data sets are said to be of shared-nothing architecture than the data intersection of the two clusters or data sets is 'none'. This architecture adds scalability to the system. [5]

C. Hadoop

Hadoop is an open source software framework that supports data-intensive distributed applications, licensed under Apache. It has two layers; first is the data storage layer and second is the data processing layer. Hadoop has its own file system called Hadoop distributed file system (HDFS) and the data processing algorithm called Map Reduce. [8]

HDFS has its own ways to store and process structured and unstructured data. Structured data is stored table wise (in blocks of 64MB) and unstructured data is divided into the blocks of 64MB leaving the last block of the file smaller than the rest [9]. File handler of Hadoop are Datanodes, Racks and Namenodes. Datanodes are the machines that hold the data blocks. Racks are the sets of Datanodes located physically at one place connected via high bandwidth network. Namenode is a machine that holds information about each rack, datanode and directory of each block on every datanode. Hadoop keep two replicas of each block. Due to the underlining shared nothing architecture, the HDFS files are strictly one-writer at a time type.

Hadoop works best for unstructured data. Scalability of Hadoop is the best in the market yet. Immaculate resolution of issues of heterogeneous networks and heterogeneous operating systems is another achievement of Hadoop.

For structured data, the tables are stored in different blocks which may reside on different Datanodes. Relationships between tables are stored by Namenode as metadata. So a cross table analytical query can become way too expensive computationally. Performance issues associated to Hadoop are due to the Shared-nothing architecture and single point failure of map-reduce.

D. Map Reduce

Map reduce is a data processing algorithm that works best for big unstructured data. There are two base functions of this algorithm: Map and reduce. Map finds the block of relevant data and Reduce applies the required function. There are various helping functions that assist map and reduce to work on different data blocks [6], [7]. Common example to explain map reduce is the Word Count.

Figure 2 shows a file lead into the map reduce for word count. Splitting prepares data for Map nodes, map nodes perform preliminary function on input data and produces the intermediate pairs. Shuffling collects related values and reduce performs the required function on individual intermediate pairs and final result is compiled by combining results from all the related reduce nodes.

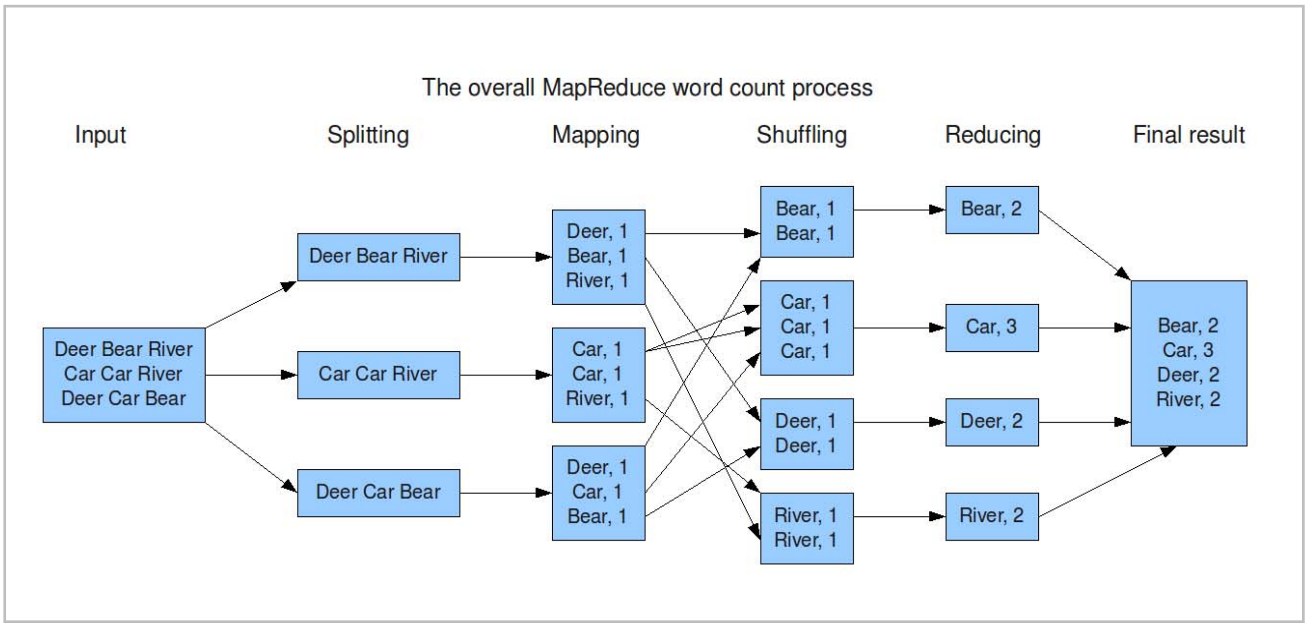


Figure 3. Illustration of Map Reduce

Two major drawbacks of map reduce are the node failure and shared-nothing clusters [9]. A single node may be running multiple maps and reduces. All processed results are in nodes local memory and are read directly from there. If a node crashes, all results in its memory are lost too, and they have to be recalculated. Secondly, shared-nothing clusters cannot share each other's load. Backups are the parasite to these systems as they sit idle and wait for the primary node to crash. If a task is equally divided into several nodes, each with different processing capacity, than the system response time will be the response time of the slowest node.

III. Proposed Algorithm (BSPF)

BSPF converts the structured data into a vertically scalable dataset, creates back up, makes shared-data clusters of the secondary dataset and deploys it over a network of machines and in this way it can ensure multiple backups of an entity, and multiple ways to get to some particular information out of the data base at a quicker response time. Figure 3 is the exemplary database to explain the algorithm.

A. Conversion from row oriented storage layout to column oriented storage layout

First thing in the deployment of this database onto the distributed environment is conversion of this row oriented database layout into the column oriented data layout. Reason of this conversion is to align the data for smooth running of analytical queries. Following are the steps of conversion.

1) ***Mapping keys:*** First step is to discover the schema of relational database by mapping its primary and foreign keys. This results in a list of tables and the keys in it along with the referenced table names. Example: The mapping keys results in a data structure holding the keys and their reference tables. For illustration purposes we use a table to demonstrate it. Figure 4a is a table that holds the list of all keys in the database (primary or foreign) and records the table these keys belong to.

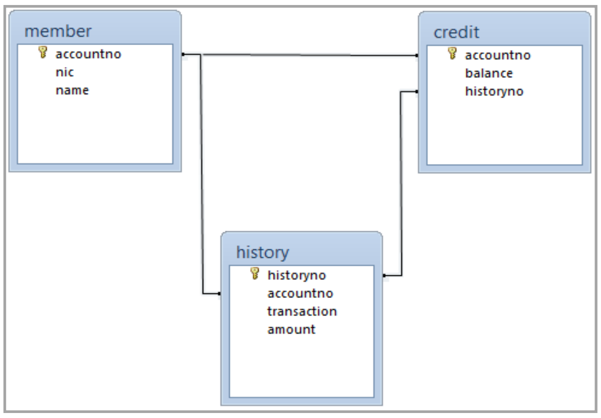


Figure 2. Exemplary Database

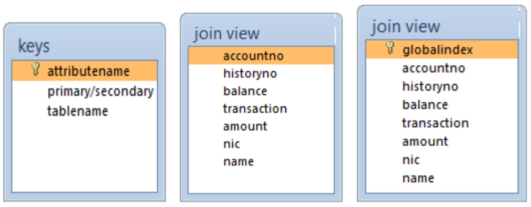


Figure 4 a,b,c Intermediate tables

2) ***Joining tables:*** Next step is to run join queries on all the tables creating a view that holds all the attributes of the data. This join finishes the purpose of primary and foreign keys. Redundant fields are removed after the join. Example: Figure 4b is a table named 'join view' and it holds the result of join query.

TABLE 1. ATTRIBUTE COMBINATIONS

Table name	Attribute 1	Attribute 2	Attribute 3	Attribute 4
combo1	globalindex	accountno	historyno	n/a
combo2	globalindex	accountno	balance	n/a
combo3	globalindex	accountno	transaction	n/a
combo4	globalindex	accountno	amount	n/a
combo5	globalindex	accountno	nic	n/a
combo6	globalindex	accountno	name	n/a
combo7	globalindex	historyno	balance	n/a
combo8	globalindex	historyno	transaction	n/a
combo9	globalindex	historyno	amount	n/a
combo10	globalindex	historyno	nic	n/a
combo11	globalindex	historyno	name	n/a
combo12	globalindex	balance	transaction	n/a
combo13	globalindex	balance	amount	n/a
combo14	globalindex	balance	nic	n/a
combo15	globalindex	balance	name	n/a
combo16	globalindex	transaction	amount	n/a
combo17	globalindex	transaction	nic	n/a
combo18	globalindex	transaction	name	n/a
combo19	globalindex	amount	nic	n/a
combo20	globalindex	amount	name	n/a
combo21	globalindex	nic	name	n/a
combo22	globalindex	accountno	historyno	balance
combo23	globalindex	accountno	historyno	transaction
combo24	globalindex	accountno	historyno	amount
combo25	globalindex	accountno	historyno	nic
combo26	globalindex	accountno	historyno	name
combo27	globalindex	accountno	balance	transaction
combo28	globalindex	accountno	balance	amount
combo29	globalindex	accountno	balance	nic
combo30	globalindex	accountno	balance	name
combo31	globalindex	accountno	transaction	amount
combo32	globalindex	accountno	transaction	nic
combo33	globalindex	accountno	transaction	name
combo34	globalindex	accountno	amount	nic
combo35	globalindex	accountno	amount	name
combo36	globalindex	accountno	nic	name
combo37	globalindex	historyno	balance	transaction
combo38	globalindex	historyno	balance	amount
combo39	globalindex	historyno	balance	nic
combo40	globalindex	historyno	balance	name
combo41	globalindex	historyno	transaction	amount
combo42	globalindex	historyno	transaction	nic
combo43	globalindex	historyno	transaction	name
combo44	globalindex	historyno	amount	nic
combo45	globalindex	historyno	amount	name
combo46	globalindex	historyno	nic	name
combo47	globalindex	balance	transaction	amount
combo48	globalindex	balance	transaction	nic
combo49	globalindex	balance	transaction	name
combo50	globalindex	balance	amount	nic
combo51	globalindex	balance	amount	name
combo52	globalindex	balance	nic	name
combo53	globalindex	transaction	amount	nic
combo54	globalindex	transaction	amount	name
combo55	globalindex	transaction	nic	name
combo56	globalindex	amount	nic	name

3) ***Global index:*** Then comes the global indexing, it helps keeping the strings between records attached. In order to transform the row oriented database to the column oriented database we have to dissolve the schema without disconnecting the linked data, this is done with the help of a unique identifier that is assigned to each record in the join view table. Example: In this step we append the table with a field named global index, as shown in Figure 4c.

4) ***Dissolving keys:*** After dissolving the keys, second step towards the column oriented layout is table resolution. Attribute combinations is used to resolve the tables. The term combination in 'attribute combination' is the mathematical term; the concept that is used in Probabilistic Mathematics (e.g. fruit salad is a combination of apples, grapes and bananas). In this method we take all the columns of the join view and create tables with every possible combination (these combinations does not include the repeating values). The global index is a mandatory part of every newly created table**.** Example: One big table 'join view' is divided into many small tables. Table I is the graphical representation of all possible two and three pair combinations of attributes of table 'join view'.

B. Clustering and Cluster index

- Clustering is the logical partition of data that eases data search. Aim of cluster generation in this algorithm is to divide data into clusters so that if a cluster crashes rest of the clusters come together to regenerate it. Clusters are formed on the basis of cluster criteria. For this particular database example we assume the cluster criteria to be 'similar attribute2 value of each table'. It means table where the value of attribute2 is similar will fall under the same cluster as shown in Figure 5.
- Each cluster has various sub clusters based on the values it holds. Number of sub clusters may vary from cluster to cluster. These sub clusters are meant to reside on different nodes so that if a node holding parent cluster crashes, these sub clusters can come together to regenerate the lost cluster. The colour coding in Figure 5 explains the concept of subclustering. All table names written in a similar color falls in the same subcluster with respect to a particular cluster. For example: combo1, combo2 and combo 3 are a part of cluster 1 and make a subcluster named 1a.
- Cluster index is a table that holds information about all clusters and nodes. It is formed in two stages. Cluster information is filled after the cluster generation and Node information is filled into the cluster index after the node formation. A node that holds the cluster index is called 'data manager'.

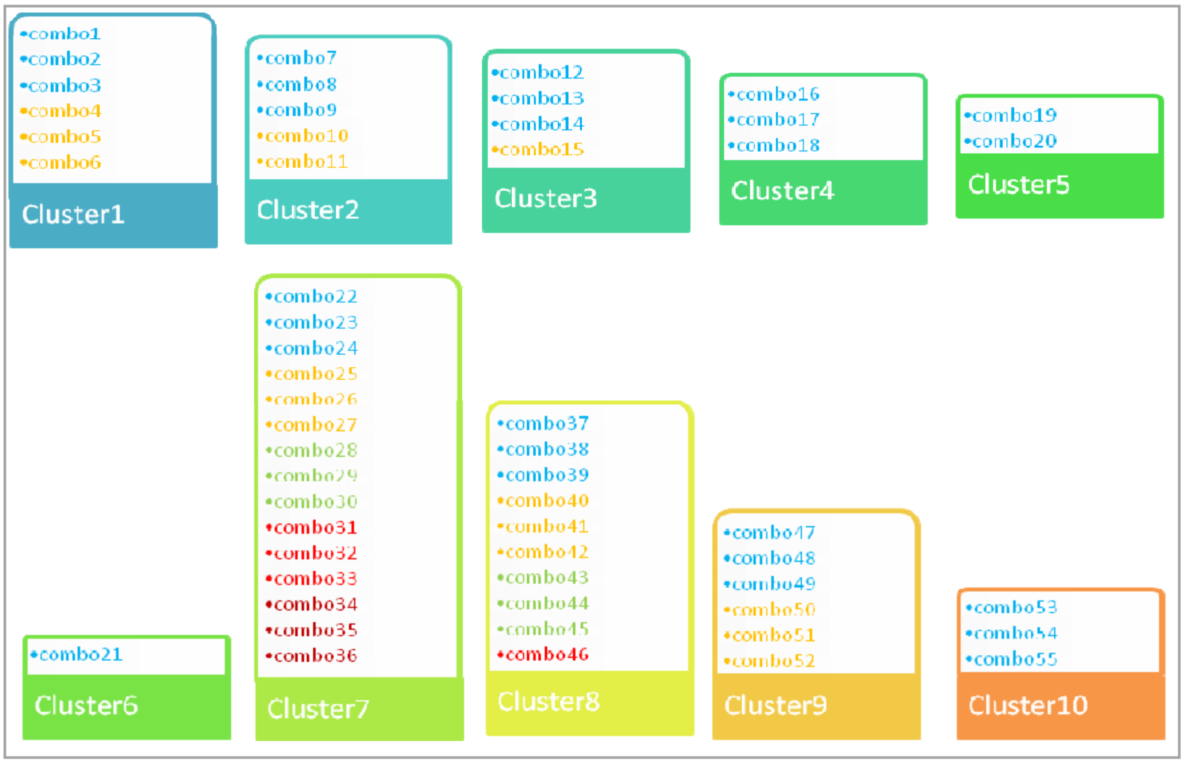


Figure 5 Cluster index Stage 1

Cluster names, list of its subclusters, tables in each subcluster and attributes of each table are all recorded in cluster index. Reason of creating cluster index is to keep record of data and its backup placement.

C. Node Creation

Node creation is the process that connects different machines that are available and form queries to run on machines which results in table creation. Reason of creating nodes is to setup a distributed processing environment. Following are the three parts of a node.

*1) **Load value:*** Load value is a variable that defines the amount of work load on a node. The greater the load value the greater is the load on a node. When it comes to assigning task to nodes; a node with a lesser load value is preferred over a node with a larger load value. A threshold defines when a node can or cannot address anymore tasks. Minimum and maximum load values are 0.0 and 3.0 respectively. 0.0 states that a node is free whereas 3.0 state that node is full. Every task adds a value of 0.1 to the node's load value.

*2) **Primary data set:*** It is a set of subclusters marked as primary on a particular node. No two nodes have same primary data. A node (node1) that holds Cluster1 as primary data is preferred in case of an equal load value with some other node. That makes node1 primary query addresser for cluster1.

*3) **Secondary data set:*** It is the portion where a node keeps back-up for its peer nodes. Node is responsible for receiving updates for this portion of data and generating an alert when an intended update is not received.

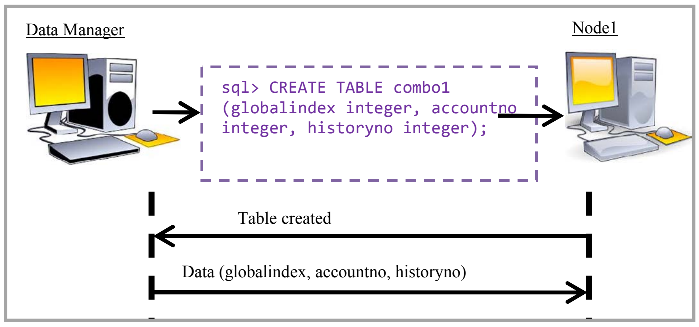
A node doesn't know whose holding backup for its primary data and whose backup it is holding, this is known to cluster index only. But in case of a load increase this node can be tasked to process a subcluster in its secondary data and generate its update.

D. Data Deployment

This process runs over the data manager and initiates certain processes on nodes. It is composed of three crucial sub-processes

- Connecting nodes is a process that makes sure the nodes are in connection. The IP pings and port check are the parts of this process.
- Table generation forms the desired query, sends its over to the node and makes sure that the table is formed at the node. Table creation statements are created on data manager and sent over to nodes to execute.
- Data deployment sends the data to the node. This process takes place once the node acknowledges table creation.

Figure 6 is the sequence diagram of data deployment.



```
sql> CREATE TABLE combo1
(globalindex integer, accountno
integer, historyno integer);
```

Figure 6. Sequence Diagram Table Creation

Up till this point we have converted the data from row oriented storage layout to column oriented storage layout, indexed and clustered the data and deployed it over several nodes in the network. With this ends the initial deployment of structured data into BSPF environment. Next Section shows the operational side of BSPF.

IV. BSPF IN ACTION

A. Query execution

Since the data is deployed at various active nodes, query can be processed at more than one node. A multipath query execution system is devised to take full advantage of the all nodes available. Decision making parameters are the primary data clusters and load value of the node. Here are few use-cases that describe how multipath query execution runs over the system.

```
sql> SELECT AVERAGE (amount)
FROM database
WHERE nic=”36101-928857-6”;
```

Figure 8 Example Query

At first the query executioner identifies the variable names and the analytical function called in the presented query like.

```
Function: AVERAGE
Variables: amount, nic
```

Figure 9 Extracted Parameters

Table name	Subcluster name	Primary node	Secondary node
combo19	subcluster5A	Node2	Node6
combo34	subcluster7E	Node4	Node5
combo44	subcluster8C	Node7	Node8
combo53	subcluster10A	Node5	Node8

Figure 10 Locating Data

Then it looks into the cluster index for a table with attributes amount and nic. It prepares a list containing the table name, subcluster name and the node name.

1) ***Case 1: (Ideal Case):*** Now to choose which node is suitable for handling this query, data manager carries out a load analysis. For load-analysis it prepares another list of nodes and their load values as shown in figure 13. From this list the data manager looks for smallest load

Primary node	Load value
Node2	0.0
Node4	1.2
Node7	3.0
Node5	2.2
Secondary node	**Load value**
Node6	0.7
Node5	2.6
Node8	2.1

Figure 7 Case 1

value. In ideal case one of the primary nodes is free having a load value of 0.0 and is appointed as handler to the query. Primary nodes are always the priority. From Fig. 13 it is more than evident that Node2 is the appointed handler of query.

Primary node	Load value
Node2	1.9
Node4	2.8
Node7	1.3
Node5	2.9
Secondary node	**Load value**
Node6	1.4
Node5	2.9
Node8	0.2

Figure 11 Case 2

2) ***Case 2:*** A secondary node has the least load value

Considering the same query, we assume a different test set of load values (see Fig. 14). Scenario created by the load values shown in Fig. 14 states that a secondary node has a least load value. In such case algorithm takes difference between least load value among primary nodes and secondary nodes

Minimum load value among primary nodes (P) = 1.3

Minimum load value among secondary nodes (S) = 0.2

Difference (d) = |(P- S)|=|1.3 - 0.2| = 1.1

If (d ≤ 0.5) query is assigned to Primary node else it is assigned to the Secondary node. Since 1.1 > 0.5, query is assigned to Node8

B. Cluster Regeneration

Cluster Regeneration is the real beauty of this algorithm. In case of cluster loss, cluster index prepares a list of two things; subclusters lost and the secondary holders of the same data. When a regeneration process is initiated secondary data holders are asked to send particular subclusters and the cluster is regenerated on a node with the least load value.

Sequence of Fig. 15 is important, following steps are numbered in correspondence to the red markings in Fig. 15.

Step 1: A node crashes and the data is unavailable.

Step 2: Data Manager prepares a loss list in which it enlists the secondary locations of the lost subclusters.

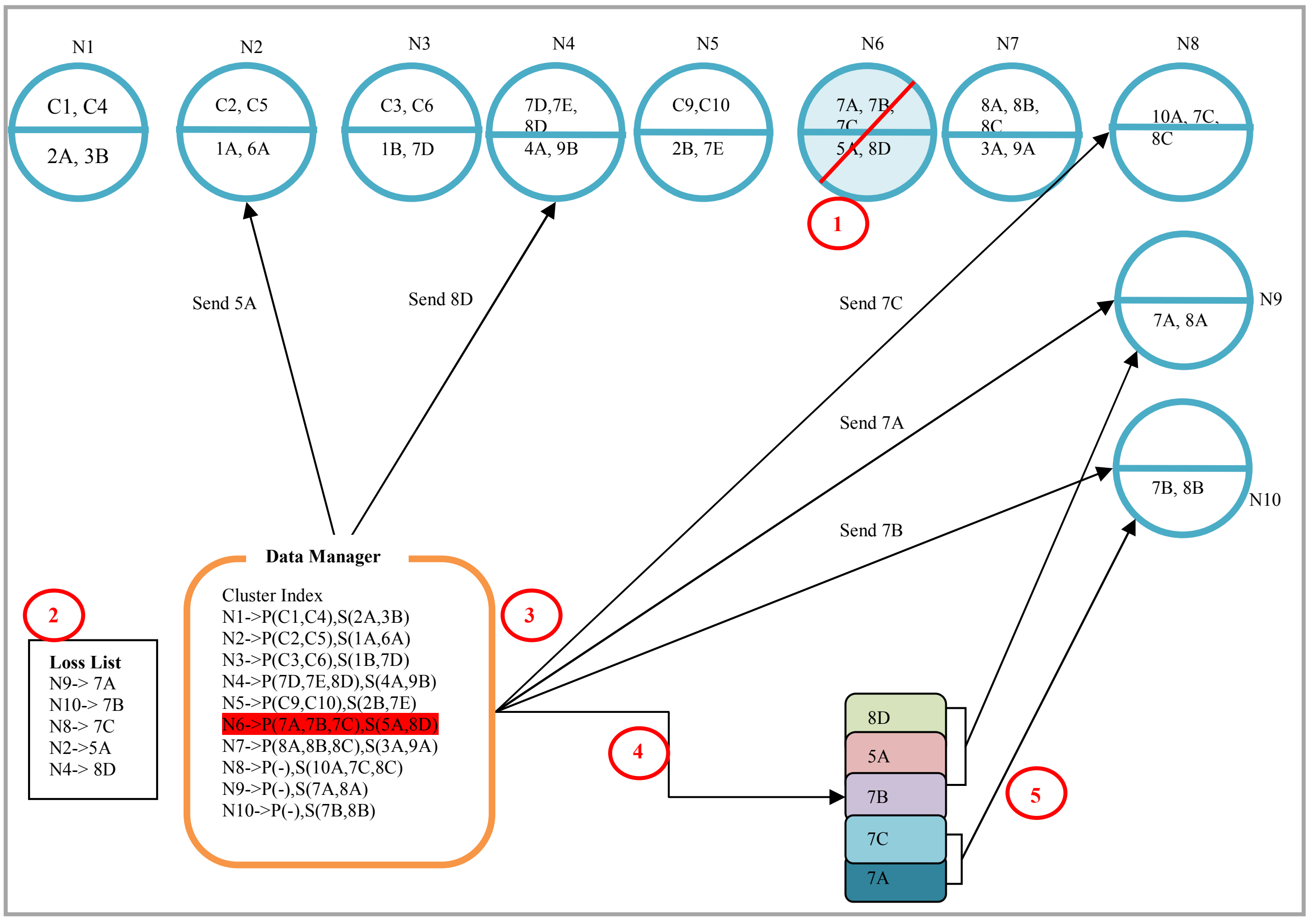


Figure 12 Cluster Recovery

Step 3: Cluster Recovery Protocol is initiated and the relevant nodes are asked to send particular subclusters. For example: Node 2 is asked to send subcluster5A.

Step 4: Subclusters are collected by the data manager.

Step 5: Based on the basic law of primary and secondary data placement in a node, these collected subclusters are deployed on the nodes that have lesser load value.

Lost subclusters are divided over different nodes to keep the total load value of a node constant. Updating Cluster Index is also part of Cluster Recovery Protocol.

V. Conclusions

It is normal human behavior in corporate environment that employees with same knowledge base helps each other in their leisure times, contributing in organizations performance. Similarly, the notion behind Shared Architectures is that the nodes should share each other's work load to enhance the system performance. Shared architecture and active-active backup are the concepts that help developing a load sharing architecture for distributed processing of unstructured data. We have presented a complete framework (BPSF) for handling the distributed structured databases, incorporating all the necessary steps. BPSF presents the methodology of data distribution and deployment on multiple nodes with enhanced performance and fault tolerance. The data loss problem (in case of hardware failure i.e node crash) is handled by developing a *Cluster Regeneration Module*; *Multipath Query Execution* provides a solution to process crash and *Active Backup* shares systems overall work load. BSPF have its cons too. Approximately, 8 times more storage is required to deploy this type of architecture, but thanks to Cloud, today storage is available at very cheap rates. Scalability is devalued. Addition of a node or table is an expensive process for the first time, but once it is in the system it will provide enhanced performance and fault tolerance. The extensive testing of the developed framework is under progress where its performance will be compared with other existing tools. However, the unique blend of the capabilities like Shared Architecture/Active-Active backup makes it more suitable for systems that are relatively of smaller size (and therefore cannot afford to have backup machines sitting idle, waiting for crashes).**References**

[1] Rolf Sint, Sebastian Schaert, Stephani Stroka and Roland Ferst, “Combining Unstructured, Fully Structured and Semi-Structured Information in Semantic Wikis”, 2009. <www.ceur-ws.org/Vol-464/paper-14.pdf>

[2] Seth Grimes, “Unstructured data and 80% rule”, 2008. <www.breakthroughanalysis.com/2008/08/01/unstructured-data-and-the-80-percent-rule/>

[3] Leaf Petersen, Robert Harper, Karl Crary, Frank Pfenning,Carnegie Mellon University, “A Type Theory for Memory Allocation and Data Layout”, 2002. <www.repository.cmu.edu/cgi/viewcontent.cgi?article=1474&context=compsci>

[4] Oracle Application Server Concepts, “Scalability and High Availability”, 2005. <www.docs.oracle.com/cd/B14099_19/core.1012/b13994/avscalperf.htm>

[5] Sunguk Lee, “Shared-Nothing vs. Shared-Disk Cloud Database Architecture”, International Journal of Energy, Information and Communications. November 2011. <www.sersc.org/journals/IJEIC/vol2_Is4/15.pdf>

[6] Jeffrey Dean and Sanjay Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters”, 2004.

[7] Tyson Condie, Neil Conway, Peter Alvaro, Joseph M. Hellerstein (UC Berkeley). Khaled Elmeleegy, Russell Sears, (Yahoo! Research),“MapReduce Online”, 2010.

[8] Apache Hadoop , “HDFS Architecture Guide”, March 2013, <www.hadoop.apache.org/docs/stable/hdfs_design.html>

[9] Azza Abouzeid, , Kamil Bajda-Pawlikowski, Daniel Abadi, Avi Silberschatz, Alexander Rasin “HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads “, 2009