International Journal of Grid and Distributed Computing
Vol. 11, No. 2 (2018), pp.21-30
http://dx.doi.org/10.14257/ijgdc.2018.11.2.03

# Shape Recognition Based on MapReduce and In-Memory Processing on Distributed File System

Namkyun Baik[1], Dipankar Hazra[2] and Debnath Bhattacharyya[3]

*[1]Korea Advanced Agency of Convergence Technology,*
*Digital-ro 285, Guro-gu, Seoul, 08381, South Korea*
*[2]Computer Science and Engineering Department, OmDayal Group of Institutions,*
*Uluberia, Howrah-711316, West Bengal, India*
*[3]Computer Science and Engineering Department, Vignan's Institute of Information Technology, Visakhapatnam-530049, Andhra Pradesh, India*
*[1]dipankar1998@rediffmail.com, [2]debnathb@gmail.com,*
*[3]white-knight@naver.com*

**Abstract**

*Two novel approaches for centroid-radii based shape retrieval on distributed file system are proposed in this paper. Modified Centroid-Radii model is used for calculating the shape features of trained images. These shape features are stored into Hadoop Distributed File System (HDFS) instead of relational database, generally used for feature storage. HDFS can store large number of shapes that is not possible to be stored in a single machine. Modified Centroid-Radii Model is also used to calculate the shape feature of query image. In one approach MapReduce query is used for recognizing binary shape. In another approach Apache Spark is used. Shape feature of query shape is compared with the shape features stored in HDFS. In-memory processing of Apache Spark used to increase the speed of retrieval process. Spark based image retrieval is faster than MapReduce based image retrieval.*

***Keywords:*** *Shape Retrieval, Distributed File System, HDFS, MapReduce, In-Memory Processing, Spark, Image Big Data*

## 1. Introduction

Image retrieval is an active research area for long time. Content Based Image Retrieval (CBIR) recognizes image based on content of the query image. Content based features [1] are basically represented as multi dimensional array or vectors. There are mainly 4 types of features like texture, colour, shape and location. CBIR consists of two phases: training phase and testing phase. In training phase, a large number of images are taken as training images with known image type. Features of these images are calculated and stored in image database. Next phase is called testing phase. In testing phase, features of a test image with unknown image type are calculated in same way as training phase. Then features of the test image are compared with features of training images in the database. Distance measure like Euclidean distance or other is used as similarity metric. Image type of unknown test image is recognized this way.

Distributed File System manages the storage of data across machines, when dataset outgrows the storage capacity of single computer. The distributed file system of Hadoop is called Hadoop Distributed File System (HDFS). Over the years HDFS has become a key tool of managing big data [2, 3] analytics applications. These types of applications are defined by 3Vs- volume, velocity and variety. Volume refers to huge amount of data

Received (October 26, 2017), Review Result (January 15, 2018), Accepted (January 17, 2018)

generated in every second. These datasets can't be managed by conventional relational database system. Big data tools can store and analyze these. Velocity refers to the speed at which the data is produced, stored, analyzed and visualized. Now data is produced in real time and devices passes their data the moment it is produced with the availability of internet connected devices. Variety represents the data of different format. In the past, data was structured and easily fitted in relational database. But now data are of different formats like structured, semi-structured, unstructured and mixed. Data may also be incomplete, inconsistent, complex and variable. The wide variety of data requires different techniques to store data. There may be different sources of big data like video, audio, photo, weblog, books, research documents, customer data, Internet of Things, real-time sensor data *etc*. Now images and image sequences (videos) make up most of the unstructured big data. HDFS cluster stores huge amount of unstructured data. Regular file is spilt in smaller chunks called blocks and copies of the chunks are created depending on the replication factor of the file. The chunks are stored in different data nodes of the cluster.

MapReduce is a popular distributed computing framework [4]. Hadoop used MapReduce for processing data of HDFS. MapReduce implements parallel computing. It consists of two different phases, Map and Reduce. Each phase is capable of running on two different nodes or machines. Map phase transform data into key-value pair. Reduce phase aggregate Map phase result. All the key-value pairs are aggregated over keys.

But MapReduce is becoming legacy now. Apache Spark [5] is outperforming popular MapReduce distributed framework. It is distributed in-memory computing frameworks. Spark works in a simple way. It shares memory across the cluster and keeps all in memory as long as possible. In MapReduce jobs intermediate results are written into disk, where as in Spark intermediate results are written into memory. So the bottleneck of MapReduce job *i.e.*, disk I/O is not present in Spark and it is faster than MapReduce. Spark's approach to fault tolerance also requires less time and resource. MapReduce has to launch separate Java Virtual Machine (JVM) for handling fault tolerance. But Apache Spark uses Resilient Distributed dataset (RDD) for fault tolerance. Resilient Distributed Dataset (RDD) is the main concept of in-memory processing. RDD is an immutable distributed collection of data with two types of operations: transformation and action. Transformation yields another RDD, whereas action writes to returns value to the master or writes to a stable storage system. Entire RDDs formed by transformation is called RDD lineage. It is a logical execution plan of a system. It is a Directed Acyclic Graph (DAG) of the entire parent RDDs of final RDD. DAG helps to recover the lost data by computing it from previous stage. Spark is compatible with Hadoop, YARN, standalone cluster mode etc. Its APIs also have rich functionality and programming languages. MapReduce dealt with lower level programming. Spark exposes APIs at higher level and avoids much of the coding of MapReduce jobs.

Some of the common requirements of Distributed Computing Framework for better multimedia based web services are scalability, computational flexibility, capacity, updates, flexible pipelines, simplicity [6]. MapReduce uses HDFS to store feature vectors. Spark also uses HDFS. Hence they are scalable. MapReduce partially supports second and third requirements, whereas fails for other requirements. Spark satisfies all the requirements. Spark defined many operators for manipulating RDDs. Keeping RDDs in RAM increase the speed of the algorithm. It also allows chaining of operations on RDDs. Large high dimensional feature vectors can be used and updated by operation on RDD.

In proposed systems, shape features of images are stored in HDFS. Retrieval is done using either MapReduce or Spark. The paper is organized as follows. In the related works portion, different content based image retrieval systems were surveyed. System description section discussed the proposed system in detail. Experimental result shows that accuracy of centroid-radii model is very high for proposed system. In conclusion section, the worth of the proposed system is mentioned.

## 2. Related Works

Shape based image retrieval uses shape features for recognizing image. centroid-radii model proposed by K. L. Tan, *et al.*, [7] is one of the efficient methods for shape based image retrieval now. Lengths of the shape's radii from centroid to boundary are used to represent the shapes. So the shape vectors do not depend on location of the shape of the image, thus the representation is invariant to translation. The shape vector is divided by major axis length to give normalized shape vector and so the method becomes scaling invariant. But this method is not robust to rotation. Modified version of centroid-radii model [8] solved problem multiple major axis along with rotation invariance. Here shape vector is calculated from one major axis and one direction (anti-clockwise) and stored in the shape database, whereas all the major axis and both direction (clockwise and anti-clockwise) shape vectors have been considered for query phase. For storing shape vectors, one axis and one direction is chosen to reduce the volume of the shape database. But separate query vectors consisting of all major axis and both direction are used for retrieval, such that the process does not miss any of the matching shape.

Researchers are now addressing the retrieval of big data images. The trends are mainly high volume of multimedia data for indexing, multiple core computers and easier access of cloud. A method of recognition of binary shape on distributed file system [9] used Modified Centroid-Radii model for calculating features from binary shapes in training phase. Features are stored in HDFS. Modified centroid-radii model is also used for calculating features for query shape in testing phase. Features of the query shape are matched with features stored in HDFS file system using MapReduced retrieval. This method is highly efficient and can store huge amount of features because of HDFS file storage, but retrieval time is needed to be reduced. Some other methods are exist which use HDFS as storage. One method use relevance feedback mechanism [10] to retrieve images from distributed file system of Hadoop. Distributing the load in data nodes reduces response time. The first level retrieve image on single shape feature. The second level retrieves images on relevance feedback. Relevance feedback mechanism is implemented using particle swarm optimization. It reduces the drawback of optimization and fuzzy results in relevance feedback mechanism. Method for processing large number of small image files like face detection is proposed on HDFS [11]. SequenceFile of Hadoop file format converts many small image files into large file as input and stored in HDFS. The small files are in <key, value> format where key is the index information and value is file data. Small files do not preserve their format in SequenceFile and do not be accessed directly from SequenceFile. Pre-processing is also required. Another approach combines multiple input files of same node in InputSplit to increase performance by reducing amount of data to be transferred between nodes. Combination of images and then processing is proved to be more effective than SequenceFile method for face detection problem of large number of small files in HDFS. Each node has a Hadoop framework installed on a virtual machine. Management of HDFS become easier by cloning virtual machine. Method is also proposed for indexing and searching 100M images with MapReduce on top of Hadoop distributed framework [12]. Hadoop is useful for achieving scalability. For single machine, a lot of time is required not only for creating and using indexes, but also for copying data to HDFS. Hence this type of system works efficiently when bandwidth is enough for feeding grid/cloud with required data. A system incorporates local tetra pattern technique [13] for forming feature vector and MapReduce technique for searching image from Hadoop Distributed File System. Local tetra pattern technique calculates the relationship between centre pixel and its neighbour pixels by computing the grey level differences. The feature vectors are then stored in the HDFS. The feature vector of query image is compared with the feature vectors stored in HDFS using Euclidean distance. The text files used for storing feature vectors of images in HDFS are completely secured. MapReduce distributed processing on HBase table is fast and accurate and applied in remote sensing image retrieval [14] for images like house, lake, road, river *etc.*, MapReduce is adopted in both phase, *i.e.*, storage and retrieval. In the Map process of storage phase colour and texture features of remote sensing images are extracted. These features are stored into HBase table in Reduce process of storage phase. In the Map process of retrieval phase color and texture features of query image is extracted and compared with same values of training features in HBase table. Key-value pair for < similarity, imageid > is calculated. In the Reduce process of retrieval phase key-value pair is sorted and similar image is recognised. Hadoop Map Reduce processing framework is applied to process extremely large video files [15] of images of persons extracted by CCTV camera. Image frames are extracted from video files. Image frames are converted into Hadoop sequence file and then sequence file into array for storage into HDFS. MapReduce distributed processing model and HDFS storage model is also used for content based image retrieval of medical images [16]. MapReduce model extract visual features from medical images and stores into HBase table. When user sends a query image, it will be temporarily stored in HDFS. Map phase extract features from query image, store image features in HDFS. Similarity between query image in HDFS and training images in HBase is computed. Reduce phase collect and combines all the result from Map phase and store final recognized image in HDFS. Kullback-Leibler Divergence and Euclidean distance are used to calculate similarity between features.

Apache Hive, Pig, and Spark are other big data tools for storage and retrieval [17]. Apache Pig is a data warehouse tool to provide data summarization, query and analysis. The Hive gives SQL like interface to query data stored in various databases and file systems. Apache Pig is high level scripting language that is used for complex data transformation. Instead of writing code in Java for implementing Map Reduce, SQL like scripting language Pig Latin can be used for data processing on Hadoop clusters. Apache Spark is high speed engine for big data processing. Its fundamental data structure is Resilient Distributed Dataset (RDD). RDD is an immutable distributed collection of objects stored in memory. Spark holds intermediate result in memory in form of RDDs, which are more useful when same dataset is used for multiple times. It supports lazy evaluation of query. Execution of DAG graph starts after action is performed on Resilient Distributed dataset (RDD). Spark is more suitable for handling medical storage and retrieval than Apache Pig and Hive. Hadoop and Spark based image retrieval system [18] is completely scalable and reduces computational cost. Hadoop is used for indexing and Spark is used for retrieval. Performance is measured with respect to computational cost instead of retrieval rate. Performance can be evaluated for different number of images and it can be shown that the system is useful for both small and large number of images. Spark based system works more efficiently than only MapReduce based system. Improved Content Based Image Retrieval System (CBIR) using Spark [19] works efficiently for huge amount of images. Speeded Up Robust Feature (SURF) algorithm is used to extract features from images and convert images into java classes. Thus huge number of small files will be created. Image database is constructed using Apache Avro to combine these files. Avro’s file structure is modified to become suitable for random read. An Avro file is an element of RDD, hence to be persisted in memory and reused. Retrieval will be done from Avro files’ block area. The system spends little more time in constructing database but real time retrieval is faster and system gives better result for large amount data. Previous works of Spark based retrieval lack the structure of RDDs and their lineage. Hence the design of a complete system of Content Based Image Retrieval using in-memory processing is required and proposed.

## 3. Proposed Work

Initially shape vectors are computed for training images using Modified Centroid-Radii model and stored in the nodes of ‘Hadoop Distributed File System (HDFS)’. Either

MapReduce based query or Apache Spark based query is used image recognition. The steps are discussed in detail in following:

### 3.1. Shape Vector Computation by Modified Centroid-Radii Model

Centroid (Cx, Cy) of image B is calculated using following formula,

$$C_x = \sum_{y=1}^{n} \sum_{x=1}^{m} B.*M_x \Big/ \sum_{y=1}^{n} \sum_{x=1}^{m} B \tag{1}$$

$$C_y = \sum_{y=1}^{n} \sum_{x=1}^{m} B.*M_y \Big/ \sum_{y=1}^{n} \sum_{x=1}^{m} B \tag{2}$$

Where B=Image matrix with foreground elements=1 and background elements=0
[m, n]=Size of Image B

Mx=Matrix of size [m, n] and all elements of this matrix is set to its x-coordinate

My=Matrix of size [m, n] and all elements of this matrix is set to its y-coordinate

B.*Mx= Element by element multiplication of B and Mx

B.*My= Element by element multiplication of B and My

Distance from centroid (Cx, Cy) to boundary pixel (Px, Py) is called radius L, which is calculated using following Euclidean distance formula:

$$L = \sqrt{(P_x - C_x)^2 + (P_y - C_y)^2} \tag{3}$$

where (Px, Py) is the coordinate of boundary pixel. Maximum value radius is called major axis L1. There may be more than one major axis for the shape. Major axis length will be starting point of calculation. $S_{IC}$ is the initial clockwise shape vector. It is started from major axis L1 and consists of n radius separated by equal distance from each other in clockwise direction. Ln is n-th radius from major axis in clockwise direction.

$$S_{IC} = [L1\ \ L2\ \ ......Ln] \tag{4}$$

Initial clockwise shape vector is divided by major axis length L1 to give normalize clockwise shape vector $S_{NC}$.

$$S_{NC} = [L1/L1, L2/L1 ..........Ln/L1] \tag{5}$$

Normalization solves scaling invariance problem. Similarly normalize anti-clockwise shape vector $S_{NA}$ will be calculated starting from major axis L1, but in anti clockwise direction. $Ln^A$ is n-th radius from major axis in anti-clockwise direction.

$$S_{NA} = [L1/L1, L2^A/L1..........Ln^A/L1] \tag{6}$$

These shape vectors are calculated for all major axis for all training images.

### 3.2. Storing Shape Vectors in HDFS

All the shape vectors of training images are stored in files of Hadoop Distributed File System (HDFS). The design HDFS based on two types of nodes, a namenode and multiple datanodes. Namenode manages all the metadata needed to store and retrieve the actual data from datanodes. At the time of copying data into HDFS, it is automatically sliced and then stored on different datanodes. Each slice is a different part of the total data set. But user sees this as original file. Namenode/datanode is a master/slave architecture where master (namenode) manages opening, closing, renaming files and directories and the slaves (datanodes) are responsible for serving write, read requests from the file system to clients. There is also secondary namenode that performs checkpoints of namenode file system's state but is not a failover node.

### 3.3. Recognition of Query Shape

Two methods have been proposed for recognition of query shape. One is MapReduce based retrieval and another is retrieval by in-memory processing.

**3.3.1. MapReduce based Shape Retrieval:** Assume $Im_1$, $Im_2$,........., $Im_M$ are M images of training image data set of proposed system. Input file will be split and a Mapper will be assigned to each of the split depending on user defined split size. If split size is not specified, number of Mappers depends on the number of HDFS blocks. In the Mapping phase, similarity or distances between query image and training images are calculated to find the similar most training image of query image. Here {distance, image type} is formed as key-value pair, where distance $d_t$ is calculated using following Euclidean Distance Formula:

$$d_t = \sqrt{(L_{t1} - L_{q1})^2 + (L_{t2} - L_{q2})^2 + .............+ (L_{tn} - L_{qn})^2} \qquad (7)$$

where $L_t = \left[L_{t1} L_{t2} .....................L_{tn}\right]$ is feature set of training image $Im_t$.

and $L_q = \left[L_{q1} L_{q2} .....................L_{qn}\right]$ is feature set of query image $Im_q$.

$d_{min}$ is the minimum distance between query image and a training image, where as $Im_{min}$ is the corresponding training image. Similarly, $d_{2min}$ is the second most minimum distance between query image and a training image and $Im_{2min}$ the corresponding training image, so on. $d_{max}$ is the maximum distance between query image and a training image, where as $Im_{max}$ is the corresponding training image. The Mapper output (intermediate data) is stored in local file system (not in HDFS) which is cleaned up after every Hadoop job completes. The outputs from Mappers are shuffled and sorted in Reducers. Output files contain {distance, image type} sorted in ascending order of distances for input file such that similar most image can be easily identified. Figure 1 displays steps for the MapReduce query for shape recognition.

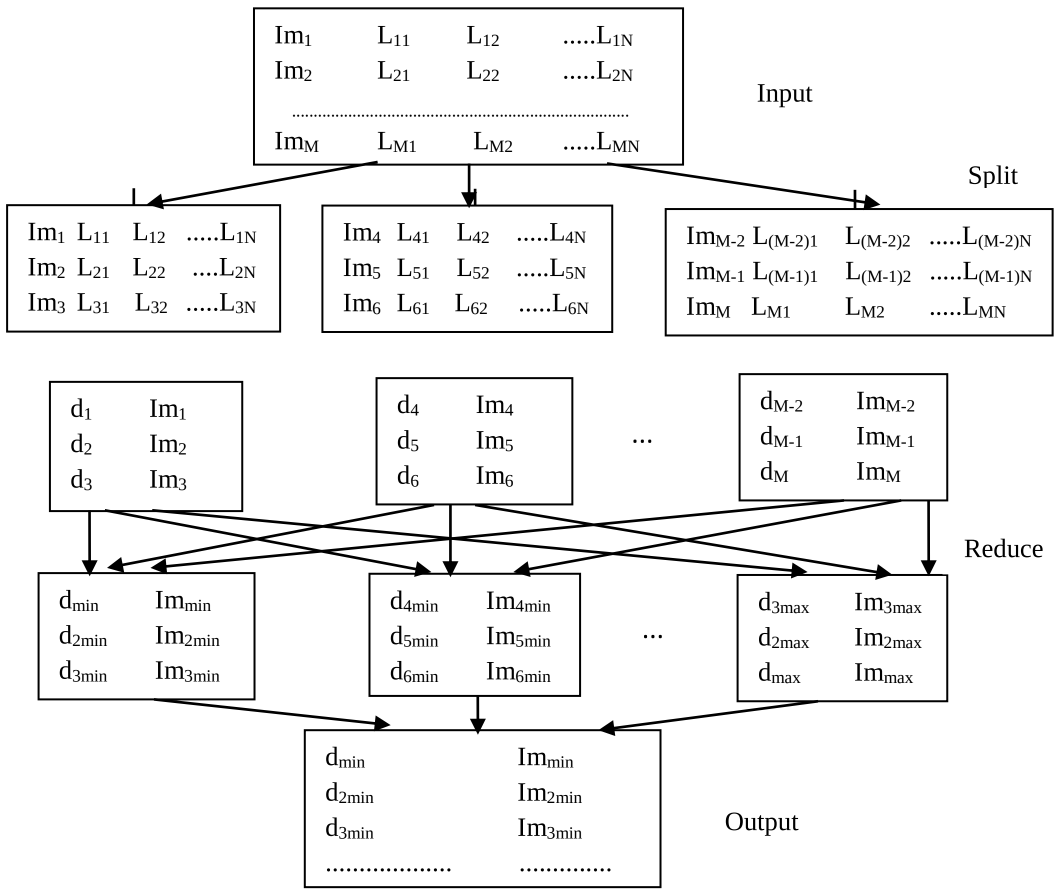

**Figure 1. Diagram of MapReduce Query for Shape Retrieval**

**3.3.2. In-memory Processing for Shape Retrieval:** In-memory processing of Apache Spark is another alternative for recognition of binary shape. Figure 2 shows the Directed Acyclic Graph (DAG) for of shape retrieval using Spark. The DAG consists of the different RDDs formed during query phase and their logical sequences.

Lines will be read from HDFS files to memory and RDD is created. In initial transformation, similarity or distances between query image and training images are calculated to find the similar most training image of query image and a new RDD is created with {Image type, distance}, where distance d is calculated using Euclidean distance formula of Equation (7).

In the next transformation, a new RDD is created by comparing the distances of previous RDD with threshold distance. The images with distance less than threshold distance are the neighbours of query image. In the action step, the neighbouring images are being reduced with respect to image type for retrieving the type of the query image.

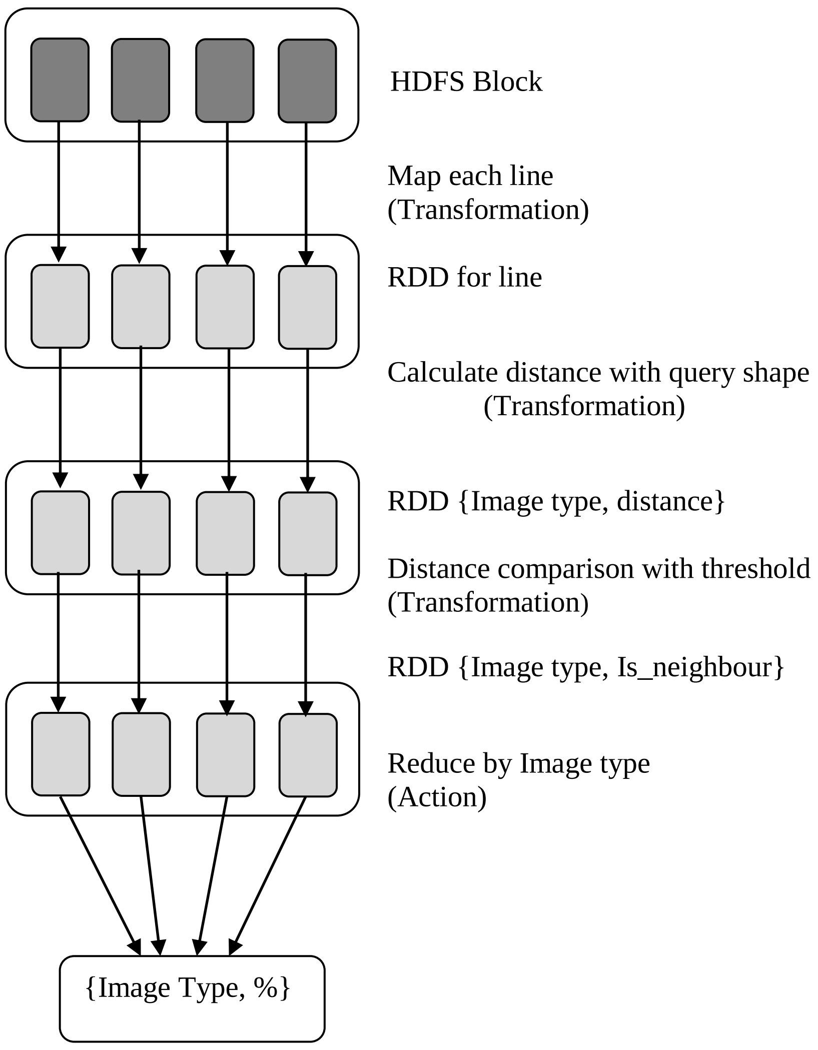

**Figure 2. DAG of RDD for Shape Retrieval**

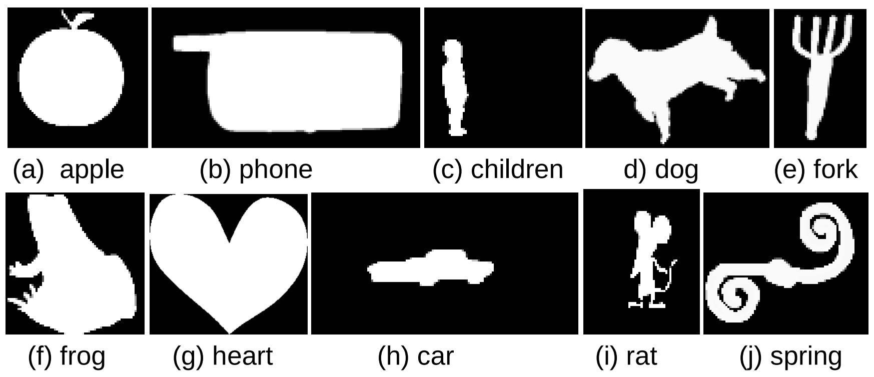

**Figure 3. Sample Training Shapes Stored in HDFS**

**Table 1. Retrieval Accuracy of Different Shapes**

| Image name | Apple | Phone | Children | Dog | Fork | Frog | Heart | Car | Rat | Spring |
|---|---|---|---|---|---|---|---|---|---|---|
| Accuracy rate | 100% | 90% | 100% | 100% | 100% | 100% | 100% | 95% | 100% | 100% |

## 4. Experimental Result

The downloaded binary images are part of MPEG7 database [20]. Training dataset consists of the 10 type of shapes: apple, phone, children, dog, fork, frog, heart, car, rat and spring. Figure 3 shows some of the training shapes used in the experiment.

Retrieval accuracy of different image shapes is given in Table1. Result shows that retrieval accuracy of shape recognition is quite high for all the images using modified centroid-radii model.

## 5. Conclusion

In the proposed method HDFS is used for storing image features for shape based image retrieval. HDFS can store large volume of image features across many inexpensive servers that operate in parallel. It can scale to process large amount of training features. It was not possible when RDBMS was used as storage. It is also cost effective and faster than RDBMS. Training features are replicated to other nodes in the cluster. So another copy of features is available to use in case of failure of a node. Either MapReduce or Apache Spark is used in retrieval phase. MapReduce writes intermediate results in local disk, but Spark writes intermediate results in distributed memory. In-memory processing of Spark helps to increase the speed of proposed system. Since shape based retrieval is giving satisfactorily result, other image retrieval mechanism like texture, colour, spatial organization based can also be stored in HDFS platform and retrieved by In-Memory processing using Spark.

## References

[1] R. Parekh, "Principles of Multimedia", McGraw Hill Education Private Limited, India, **(2014)**.

[2] T. White, "Hadoop: The Definitive Guide, 4th Edition", O'Reilly, **(2015)**.

[3] S. Mukherjee and R. Shaw, "Big Data – Concepts, Applications, Challenges and Future Scope", International Journal of Advanced Research in Computer and Communication Engineering, vol. 5, no. 2, **(2016)**.

[4] D. Eadline, "Hadoop2 Quick-Start Guide, 1st Edition", Pearson India Education Services Pvt. Ltd, India, **(2016)**.

[5] S. Gulati and S. Kumar, "Apache Spark 2.X for Java Developers", Packt Publishing Ltd, UK, **(2017)**.

[6] G. P. Gudmundsson, B. P. Jonsson, L. Amsaleg and M. J. Franklin, "Towards Engineering a Web-Scale Multimedia Service: A Case Study Using Spark", Proceedings of the ACM Multimedia Systems conference, Taipei, Taiwan, **(2017)**.

[7] K. L. Tan, B. C. Ooi and L. F. Thiang, "Retrieving Similar Shapes Effectively and Efficiently", Multimedia Tools and Applications, **(2003)**, pp. 111-134.

[8] D. Hazra, "A Modified Centroid-Radii Model for Improved Shape Recognition", Proceedings of the National Conference on Electrical, Electronics and Computer Engineering, Kolkata, India, **(2011)** November 4-5.

[9] D. Hazra and D. Bhattacharyya, "Map Reduced Shape Retrieval using Modified Centroid-Radii Model on Distributed File System", Advance Science and Technology Letters, vol. 147, **(2017)**, pp. 42-47.

[10] B. Darsana and G. Jagajothi, "Distributed Retrieval of Images using Particle Swarm Optimization and Hadoop", International Journal of Computer Applications, vol. 71, no. 8, **(2013)**.

[11] I. Demir and A. Sayar, "Hadoop Optimization for Massive Image Processing: Case Study Face Detection", International Journal Computers, Communications & Control, **(2014)**.

[12] D. Moise, D. Shestakov, G. Gudmundsson and L. Amsaleg, "Indexing and Searching 100M images with Map-Reduce', Proceedings of the ACM International Conference on Multimedia Retrieval, Dallas, United states, **(2013)**.

[13] S. P. Dravyakar, S. B. Mane and P. K. Sinha, "Private Content Based Multimedia Information Retrieval using MapReduce", International Journal of Computer Science Engineering and Technology, vol. 4, no. 4, **(2014)**.
[14] Z. Meng, "Remote Sensing Image Retrieval Algorithm based on MapReduce and Characteristic Information", International Journal of Simulation, Systems, Science and Technology, vol. 17, no. 3, **(2016)**.
[15] K. AshaRani and S. Subhalakhshmi, "Image Recognition System in Hadoop", International Journal of Advanced Research in Computer Science and Management Studies, vol. 3, no. 6, **(2015)**.
[16] S. Jai-Andaloussi, A. Elabdouli, A. Chaffai, N. Madrane and A. Sekkaki, "Medical Content Based Image Retrieval by using the Hadoop Framework", Proceedings of the International Conference on Telecommunications, **(2013)**.
[17] P. Haripriya and R. Porkodi, "An Efficient Storage and Retrieval of DICOM objects using Big Data Technologies", International Journal of Advanced Research in Computer Science, vol. 8, no. 3, **(2017)**.
[18] L. Costantini and R. Nicolussi, "Performance Evaluation of a Novel Hadoop and Spark Based System of Image Retrieval for Huge Collections", Advances in Multimedia, vol. 2015, no. 629783, **(2015)**.
[19] Y. Duan, L. Jiang, X. Lin and X. Chen, "An Improved Content based Image Retrieval System On Apache Spark", Proceedings of the International Conference on Mechatronics, Control and Automation Engineering, Atlantis Press, **(2016)**.
[20] MPEG7 CE Shape-1 Part B Database available: http://www.imageprocessingplace.com/ root_files_V3/ image_database.htm, **(2017)**.