ADVANCES in NATURAL and APPLIED SCIENCES

ISSN: 1995-0772
EISSN: 1998-1090
2016 April 10(4): pages 347-354

Published BY **AENSI Publication**
http://www.aensiweb.com/ANAS
Open Access Journal

An Improved Hadoop Performance Model With Automated Resource Provisioning And Scalable Data Chunk Similarity Compression Within Constrained Deadline

[1]D.M. Kalai Selvi and [2]Dr.P. Ezhumalai

[1]M,E(CSE),R.M.D Engineering college,kavaraipettai,chennai-601206 Tamil Nadu.
[2]HOD of CSE Department, R.M.D Engineering college, kavaraipettai,chennai-601206 Tamil Nadu.

Received 25 January 2016; Accepted 18 April 2016; Available 28 April 2016

Address For Correspondence:
Dr.P. Ezhumalai, M,E(CSE),R.M.D Engineering college,kavaraipettai,chennai-601206 Tamil Nadu.

ABSTRACT
Hadoop is an open source implementation of MapReduce framework which is used to increase the performance of parallel processing of big data. Mostly Hadoop users will lease resources required for their work from cloud service providers like Amazon but they don't know exact configuration and time period for their resource which leads to over or under provisioning of the resource and also exceed the deadline. In addition to this, processing big data in cloud is very difficult due to their high velocity and high volume and required cloud environment to achieve scalability, availability and flexibility and traditional compression techniques are not effective. Hence we proposed an improved Hadoop performance model in which user feedback system is used for automated resource provisioning and also helps in completing job within deadline. Also, efficient data processing can be done by scalable data chunk similarity compression by partitioning big data sets into chunks and calculate similarity model among chunk to compress it . Implementing all these algorithms in cloud will lead to scalability, flexibility, reliability.

KEYWORDS: Hadoop, MapReduce, Cloud service provider, Data chunk, Data compression, similarity model, Bigdata.

INTRODUCTION

Rapid evolution of raw data from all sources like sensors, blog spots, social networking sites, aviation, etc which explored from our surroundings. This leads to the phenomena called Big Data which moved IT solutions to the other dimensions. Big data has capability to handle both structured and unstructured data where 80% of the data are unstructured data and only 20% are structured data among the whole data sets. [1] Based on the challenges faced by Big Data it can be characterized by four dimensions: volume, velocity, variety, veracity. Cloud Service provider like Amazon EC2 enables user to configure the resources needed for their application, but in current form of EC2 cloud does not support Hadoop jobs with deadline constraint. So, it is the sole responsibility of the user to assign necessary resources needed to complete the job with specified deadline which is highly challenging task. It is necessary to focus on Hadoop performance modeling which is also a critical task. Hadoop jobs has multiple processing phases with three core phases – Map phase, shuffle phase, reduce phase such that first slot of the shuffle phase is processed in parallel with map phase called overlapping stage. Other slots of the shuffle phase are processed after the map phase is processed called non-overlapping stage. In the existing Hadoop performance model does not has the capability to automate the resource provisioning and complete the Hadoop job within the specified constraints. Furthermore, the number of reduce slots are strictly constant. Because of these four V's of big data, processing of big data is very difficult in traditional data processing methodologies. To overcome this issue, cloud computing provides [2] a pledged platform for processing big data with it efficiency, agility. Besides, cloud computing platform dynamically provides, configure, and reconfigure resource as required by the end user but lack in space and time. Current data processing techniques uses recursive algorithms and has multiple iterations which leads to computation problems such as parallel memory bottlenecks, deadlocks on data accessing ,algorithm inefficiency[3]. So we proposed an improved Hadoop performance model which will automatically configures resources required by end user and complete the job within deadline. Besides, the processing efficiency can be improved by data chunk similarity model compression techniques. The content of this paper is organized as follows. In section 2 we review all related work. Section 3 will be problem analysis and Section 4 will be our proposed work and finally section 5 & 6 will be our experimental results and conclusion.

Related Work:

Some techniques that are popularly used to upgrade the performance level of Hadoop framework and their challenges are discussed here. Furthermore various compression techniques have been followed to increase efficiency for processing data in cloud are also analyzed. [4] Starfish is MADDER (MAD-Magnetism, Agility, Depth) and Self tuning system for big data analytics on big data. The main goal of starfish is to enable Hadoop user and applications to get good performance automatically throughout the data lifecycle without manually understanding the need of the Hadoop user and manually allocate the needs by tuning various knobs. STARFISH collects detailed information of Hadoop job at a very fine granularity for automation of job estimation which is a burden. [5]The general problem in forming cluster is to determine resource and configure it according to Map Reduce Job in order to meet the desired constraints such as cost, deadline, execution time for given big data analytics referred as *cluster sizing problem.* In this paper, Elastisizer ***a*** system added on the top of Hadoop stack to automate the allocation of resources to meet the desired requirements of massive data analytics by the Elastisizer will provide reliable answers to cluster sizing queries in an automated fashion by using detailed information of Hadoop job profile. In Elastisizer need detailed information of job profile which increases the overhead of high execution time. [6] In this paper, the proposed Hadoop performance model considers both overlapping and non-overlapping stages. It uses scaling factor to automatically scale up or scale down the resource allocation with the specified timeframe. Though this Hadoop performance model increase the possibility of automatic resource allocation but it suffers simple linear regression. The HP model is restricted to a constant number of reduce tasks. [7]CRESP provides automatic estimation of job execution time and resource provisioning in an optimal manner. [8]In general, the multiple waves generate better performance than single wave of the reduce phase. In CRESP, the number of reduce slots should be equal to the number of reduce slots and it consider only single wave of the reduce phase. In [9], proposed data reduction techniques where they considered low rank and sparse tensor. Low rank tensors are synthesized as sums of outer products of sparse loading vectors and a special class of linear dimensionality-reducing transformations that reduce each mode individually using a random compression matrix. They also proved "oracle" properties i.e. able to identify uncompressed sparse loadings directly from the compressed tensor data. They suggest two-step recovery algorithms: 1. Fitting a low rank model in compressed domain. 2. Per-mode l_0 /l_1 decompression. But these reduction methods can't cope up with increasing rate of big data. In [3] uses spatio and temporal features of big data to compress them on cloud. It uses two main algorithms. First, spatio temporal compression algorithms to reduce the data size. Second, clustering algorithms is developed based on spatio similarity on data streams in cloud. But this compression technique work well only till spatio and temporal data correlation is valid. As the data rate increases, these techniques can't be exploited. In this paper, [10] presented an adaptive data gathering scheme by compressive sensing for wireless sensor networks. By introducing autoregressive (AR) model into the reconstruction of the sensed data, the local correlation in sensed data is exploited and thus local adaptive sparsity is achieved. The recovered data at the sink is evaluated by utilizing successive reconstructions, the relation between error and measurements. Then the number of measurements is adjusted according to the variation of the sensed data. Furthermore, a novel abnormal readings detection and identification mechanism based on combinational sparsity reconstruction is proposed. Up to about 8dB SNR gain can be achieved over conventional CS based method with moderate increase of complexity. [11]The big data stored on distributed sites in cloud can be queried using "Hydoop" and related concepts like "Hive, Hbase, and Zookeeper". In order to improve the efficiency and reliability of Hadoop Performance model by providing user friendly automated resource provisioning scheme with best data compression and effective job completion with the predefined constraints, an improved Hadoop performance model is proposed. Our proposed work will overcome all these issues.

Problem Analysis:

3.1 Modeling Hadoop Job Phases:

In Hadoop, any job which needed parallel processing and high performance will enter into the map reduce framework. Usually, the entered job is splitted into three core phases such as Map, shuffle and Reduce phase. In the existing system, fine granularity information about the job is collected [6] which is useful to complete the job within deadline. Map tasks are executed in map slots and reduce tasks are executed in reduce slots. Each slot runs a task at a time. Slots are assigned in terms of CPU and RAM. Map and reduce phase can be executed in single or multiple phases [12].

3.1.1 Map Phase:

The map phase accepts input dataset from Hadoop Distributed File system (HDFS) and which should be in key value pair format. The map phase initially split the input datasets into blocks (by default 64MB) where each block considered as map tasks and assign a map slot. After the map phase is processed, the intermediate key-value pair result will be generated which will be stored in buffer not in disk in order to reduce the complication due to replica. The total execution time for map phase will be eq (1)

$$T^{mtotal} = \frac{T^{mavg} * N^{m}}{N^{mslot}} \tag{1}$$

3.1.2 Shuffle Phase:

The process of transferring the map output in a sorted manner to the reduce phase in order to minimize the work of reducer is called shuffle. If $N^r \leq N^{slot}$, then the shuffle phase will be completed in single wave. The total execution time for shuffle phase will be eq (2) Otherwise, the shuffle phase will be completed in multiple waves, then it will be eq (3)

$$T^{stotal} = \frac{T^{savg} * N^{r}}{N^{rslot}} \tag{2}$$

$$T^{stotal} = \frac{(T^{w}_{1}{}^{avg} * N^{w}_{1}) + \ldots + (T^{w}_{n}{}^{avg} * N^{w}_{n})}{N^{rslot}} \tag{3}$$

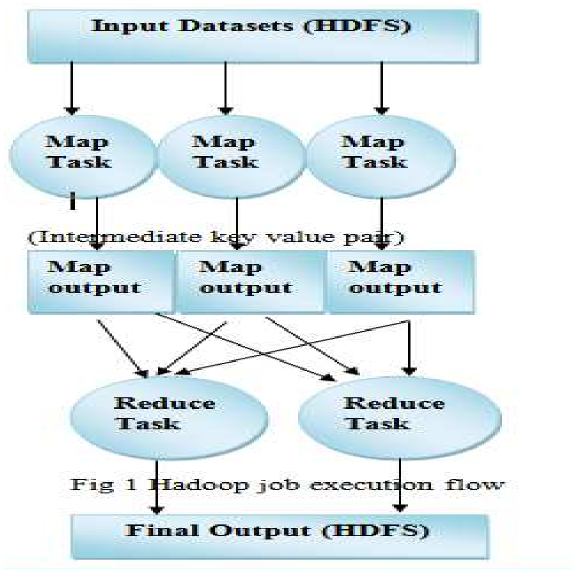


Fig. 1: Hadoop job execution flow

3.1.3 Reduce Phase:

Table 1.1: defines variables

Variables	Definition
T^{mtotal}	Total execution time for map task
T^{stotal}	Total execution time for shuffle task
T^{rtotal}	Total execution time for reduce task
T^{mavg}	Average execution time for map task
T^{savg}	Average execution time for shuffle task
T^{mavg}	Average execution time for reduce task
$T^{w}_{1}{}^{avg}$	Average execution time for shuffle task which will be completed during 1st wave

$T^{w}_{n}{}^{avg}$	Average execution time for shuffle task which will be completed during nth wave
N^{m}	Total number of map tasks
N^{r}	Total number of reduce tasks
N^{mslot}	Total number of configured map slots
N^{rslot}	Total number of configured reduce slots

The customized reduce function process intermediate map output and produces the final output .Usually, the final output after the reduce phase will be stored in HDFS. The Hadoop performance model supports both overlapping and non overlapping stages. The total execution time for reduce phase will be eq (4).

$$T^{rtotal} = \frac{T^{ravg} * N^{r}}{N^{rslot}} \tag{4}$$

3.2 Data Compression Algorithm:

The traditional data compression algorithm is explained using fig (2). In fig (2) totally, there are 13 nodes starting from node 0 to node12. Each node collects its own sensing data and stored in a traditional compression orders. All the sensed data from each node is transmitted back to node 0 in a predefined order. The drawbacks are 1.compression is centralized and not scalable. 2. It can't effectively handle huge volume of data.

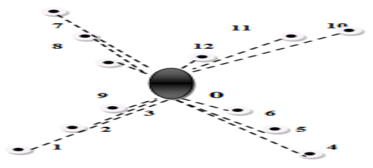


Fig. 2: Data compression algorithm

4. Proposed Work:

4.1 Improved Hadoop Performance Model:

4.1.1 Job Estimation Model:

In improved Hadoop performance model, we handle both overlapping and non overlapping stages. Generally when processing in multiple waves, the first wave of the shuffle phase starts immediately after completion of the first wave of the map phase and the first wave of the shuffle completed only after all waves of the map phase started which creates a long execution time for shuffle phase which can be overcome by HIVE. When an existing data infrastructure based on relational database wants to move in Hadoop which can be achieved using HIVE uses HQL(HIVE Query Language) similar to SQL(Structured Query language) use to fetch and process data from structured data infrastructure. The two most important concept in HIVE used to overcome the problem of latency by shuffle phase are bucketing and partitioning. When the input datasets are bucketed based on their column using hash function along with partitioning, the overhead of large datasets are reduced using command line interface or web user interface where bucketing and partitioning commands are queried in HQL based on user needs. Already the datasets are bucketed i.e. sorted based on columns the task of shuffling get reduced which improves performance. Furthermore, it helps to complete job within deadline. The estimated execution time for Hadoop jobs are finalized by user profile collects detailed information about Hadoop jobs such as data size, map tasks, reduce tasks and their duration. When a job processes an increasing size of an input dataset, the number of map tasks is proportionally increased while the number of reduce tasks is specified by a user in the configuration file. The number of reduce tasks can vary depending on user's configurations. When the number of reduce tasks is kept constant, the execution durations of both the shuffle tasks and the reduce tasks are linearly increased with the increasing size of the input dataset as considered in the HP model. This is because the volume of an intermediate data block equals to the total volume of the generated intermediate data divided by the number of reduce tasks. As a result, the volume of an intermediate data block is also linearly increased with the increasing size of the input dataset. However, when the number of reduce tasks varies, the execution durations of both the shuffle tasks and the reduce tasks are not linear to the increasing size of an input dataset. In the improved HP model, we considered varied number of reduce slots based on the user profile which has detailed information about past history input data size and their execution time and the number of map slots and reduce slots which helps to estimate the varied number of reduce slots.

4.2 Resource Provisioning Model:

Starfish and other Hadoop performance model supports manual or self tuning of the resources needed for their job execution. Resource provisioning plays a key role as a job can be completed within specified deadline say 't' only when the sufficient resources required for the job for their processing has been provided. In improved HP model supports automatic resource provisioning without manipulating knob parameters using user feedback system. In user feedback system, collects fine granularity information about the history of already processed job and by assigning weight or priority to each job based on previous logs so that high critical task can be provided resources immediately and automatically. Since even high critical task are provided resources actively which helps to complete Hadoop jobs within specified constraint.

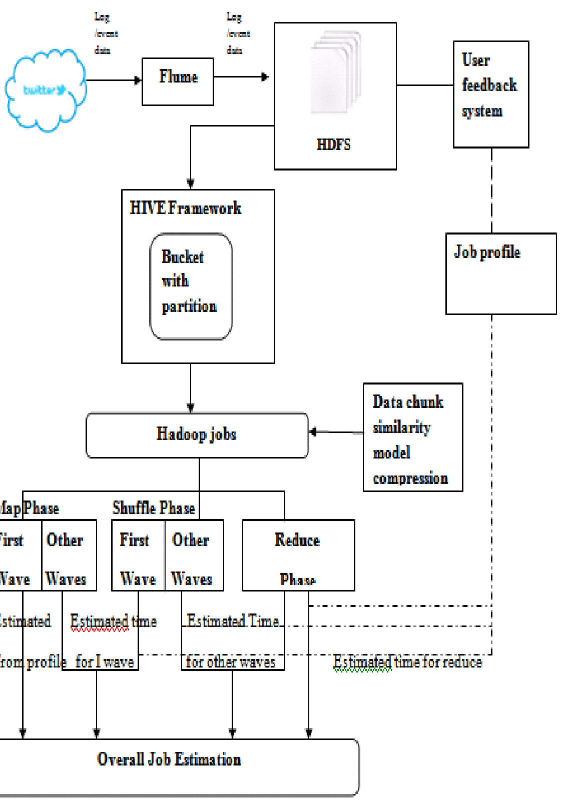


Fig. 3: Improved Hadoop performance model

4.2 Scalable Data Chunk Similarity Compression Algorithm:

As shown in Fig(4), big data sets are clustered into two or more data chunks such that similarity between two data chunks can be defined as distance(chunk1,chunk2)< threshold.

4.2.1 Similarity Model for Numeric Data:

Suppose there are two vectors $\vec{x}(x_{1,\dots}x_n)$ and $\vec{Y}(y_{1,\dots,}y_n)$ and similarity among numeric

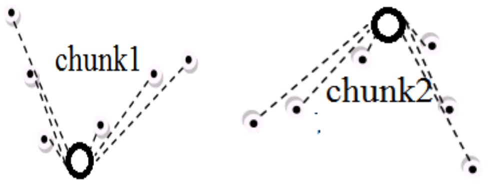


Fig. 4: Data chunks data can be defined by geometric approach. Geometric similarity describes the distance between two objects in geometric space. The similarity model for numeric data can be defined as in eq(5) & eq(6).

$$\mathrm{Sim}_{n1}(\vec{x},\vec{Y}) = \cos\theta \tag{5}$$

$$\mathrm{Sim}_{n2}(\vec{x},\vec{Y}) = \frac{\|\vec{x}\|}{\|\vec{Y}\|} \tag{6}$$

4.2.2 Similarity Model for Text Data:

Suppose there are a string pair $p(s_1,s_2)$ and a timestamp series $t= t_{1,}\ t_{2,\dots}t_{n.}$ We can define joint probability P of each pair by the state time stamp series as

$$P\,(p,t|\Theta\,) = \pi_{t1}\prod_{i=1}^{i=i+1}(\ \tau_{ti\ ti+1})\prod_{i=1}^{n}(\,O_{tipi}\,) \qquad (7)$$

$$\Theta \equiv (\pi,\ \{\ \tau_s\}_s,\{\{\ O_s\}_s\}\ ) \qquad (8)$$

Formula 8 is the parameters consisting of states of the initial, transition and output probabilities.

4.2.3 Data Chunk Generation:

Suppose there is a big data unit S={ $x_1, x_2, \dots x_n$}, then the similarity model for data vectors x_1 , x_2 can be calculated using formulas (5)(6)(7)(8) based on the type of data. Now we will generate a standard data chunk set S' as follows

t=i

$\{x_1\}$

$\{x_1, x_1+x_2\}$

…….

$\{x_1, x_1+x_i\}$

……

$\{x_1, x_1+x_2, x_1+x_2+x_3\}$

……..

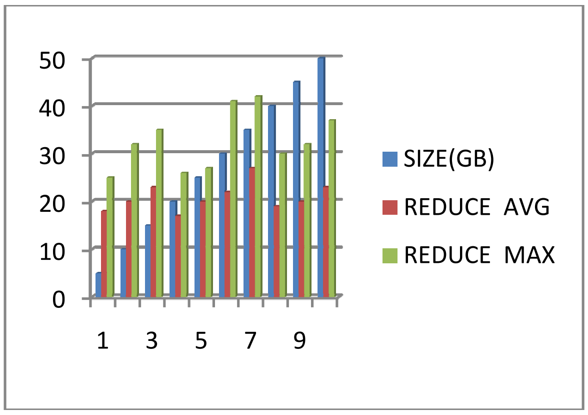
$\{x_1, x_1+x_2, x_1+x_2+x_i\}$

5. Performance Evaluation:

5.1 Experimental Setup:

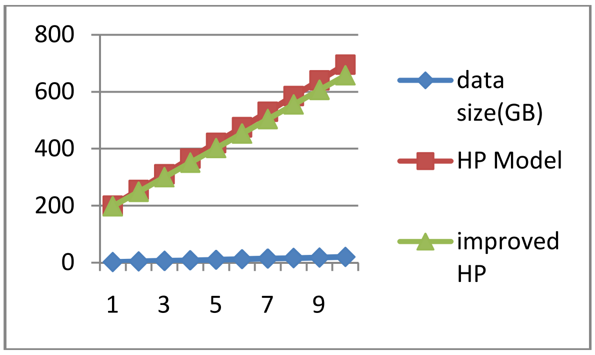
The experimental Hadoop cluster was setup on Amazon EC2 Cloud using 20 m1.large instances. The specifications of the m1.large. In this cluster, we used Hadoop-1.2.1 and configured one instance as Name Node and other 19 instances as Data Nodes. The Name Node was also used as a Data Node. The data block size of the HDFS was set to 64MB and the replication level of data block was set to 3. Each instance was configured with one map slot and one reduce slot. We run WordCount applications on Hadoop cluster and employed Starfish to collect the job profiles. For each application running on each cluster, we conducted 10 tests. For each test, we run 5 times and took the average durations of the phases.

Fig.4. the performance of the WordCount application with a varied number of reduce tasks

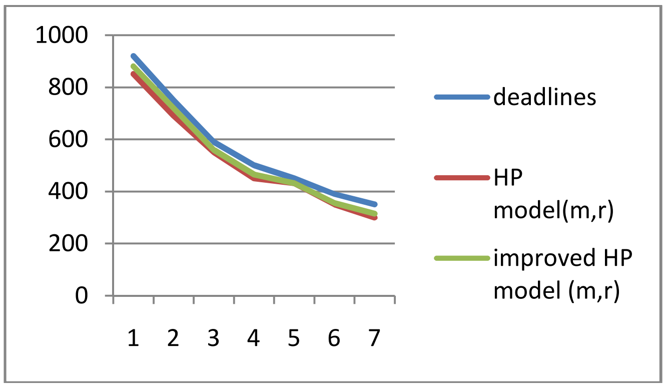


DATA SIZE(GB)	MAP TASKS	MAP TASK DURATION		SHUFFLE DURATION IN FIRST WAVE		SHUFFLE DURATION IN OTHER WAVE		REDUCE DURATION	
		AVG	MAX	AVG	MAX	AVG	MAX	AVG	MAX
5	80	12	23	69	73	20	22	18	25
10	160	12	24	139	143	26	29	20	32
15	240	12	23	212	215	38	44	23	35
20	320	13	23	274	278	34	39	17	26
25	400	13	25	346	350	41	47	20	27
30	480	11	24	408	411	47	57	22	41
35	560	11	27	486	489	59	71	27	42
40	640	14	24	545	549	45	52	19	30
45	720	12	23	625	629	50	58	20	32
50	800	11	24	693	696	55	65	23	37

The following graph describes the performance of improved HP model versus HP model for job estimation time



The following graph describes the estimated resource provisioning and compares resource provisioning performance HP model versus improved HP model



Conclusion:

Running a Map Reduce Hadoop job on a public cloud such as Amazon EC2 necessitates a performance model to estimate the job execution time and further to provision a certain amount of resources for the job to complete within a given deadline and increase the efficiency of storage using data chunk similarity compression. This paper has presented an improved HP model to achieve this goal taking into account multiple waves of the shuffle phase of a Hadoop job. The experimental results showed that the improved HP model outperforms both Starfish and the HP model in job execution estimation and resource provisioning. One future work would be to consider dynamic overhead of the VMs involved in running the user jobs to minimize resource over-provisioning.

REFERENCES

1. http://www.redbooks.ibm.com/.
2. Alamri, W.S., M.M. Ansari, M.S. Hassan, A. Hossain, Alelaiwi and M.A. Hossain, 2013. “A Survey on Sensor-Cloud: Architecture, Applications, and Approaches,” International Journal of Distributed Sensor Networks, pp: 1-18.
3. Yang, X., C. Zhang, J. Liu, Pei, K. Ramamohanarao and J. Chen, 2014. “A Spatiotemporal Compression based Approach for Efficient Big Data Processing on Cloud,” Journal of Computer and System Sciences (JCSS). 80: 1563-1583.
4. Herodotou, H., H. Lim, G. Luo, N. Borisov, L. Dong, F.B. Cetin and S. Babu, 2011. “Starfish: A Self-tuning System for Big Data Analytics,” in In CIDR, pp: 261-272.
5. Herodotou, H., F. Dong and S. Babu, 2011. “No One (Cluster) Size Fits All: Automatic Cluster Sizing for Data-intensive Analytics,” in Proceedings of the 2nd ACM Symposium on Cloud Computing (SOCC ’11), pp: 1-14.
6. Verma, L. Cherkasova and R.H. Campbell, 2011. “Resource provisioning framework for mapreduce jobs with performance goals,” in Proceedings of the 12th ACM/IFIP/USENIX international conference on Middleware, pp: 165-186.
7. Chen, K., J. Powers, S. Guo and F. Tian, 2014. “CRESP: Towards Optimal Resource Provisioning for MapReduce Computing in Public Clouds,” IEEE Transcation Parallel Distrib. Syst., 25(6): 1403-1412.
8. Herodotou, H., “Hadoop Performance Models,” 2011. [Online]. Available: http://www.cs.duke.edu/starfish/files/hadoop-models.pdf. [Accessed: 22-Oct-2013].
9. Sidiropoulos, N. and A. Kyrillidis, 2012. “Multi-Way Compressed Sensing for Sparse Low-Rank Tenors,”IEEE Signal Processing Letters, 19(11): 757-760.

10. Wang, J., S. Tang, B. Yin and X. Li, 2012. “Data Gathering in Wireless Sensor Networks Through Intelligent Compressive Sensing,” Proceedings IEEE INFOCOM, pp: 603-611.
11. Hadoop, http: // Hadoop. apache.org, accessed on November 20, 2015.
12. “Hadoop Definitive guide “ Tom White O’Reilly Press
13. Capriolo, E., D. Wampler and J. Rutherglen, 2012. "Programming Hive", O'Reilley.