Effective Utilization of Storage Space by Applying File Level and Block-Level Deduplication over HDFS

Sachin Arun Thanekar, Kodukula Subrahmanyam, AliAkbar Bagwan

Abstract: ***Hadoop framework is very efficient and easy to handle huge records storage as well as its processing. Hadoop makes use of massive commodity hardware clusters to save and process massive data in an allotted fashion. Open Source, Massive information handling capabilities and faster processing abilities made it very popular. Existing Hadoop Framework destroys metadata of preceding jobs, it actually allocates Data Nodes via ignoring what it has processed earlier and hence for each new process it reads data from all Data Nodes. There isn't any provision made for checking relationships between similar data blocks. Thus it weakens the Hadoop overall performance. The uploaded big data files are partitioned in to number of blocks and are distributed over node clusters. To avoid random block distribution and data-duplication, deduplication system is used. Such deduplication system focuses on space management and only keeps track of data files on Hadoop Distributed File System (HDFS). Such system do not participate in efficient job execution in map reduce environment. For efficient execution of job, data locality information and job metadata is stored. Time required for job execution can be decreased for next execution of same job by preserving job metadata. A combined environment produce efficient job execution results with efficient space management.***

Keywords: HDFS, Hadoop, MapReduce, BigData, H2hadoop.

I. INTRODUCTION

At present the rate of electronic data increase is quickened. This huge data is majorly stored on databases and in distributed fashion across the globe. This data might be structured, semi-structured or unstructured. Extensive usage of social media sites, e-commerce sites, blogs, different databases etc. are resulting in more and more data. Various organizations are using this data for their business needs. Based on the analysis of current data statistics they are foreseeing future trends of business.Storage, Extraction and Analysis of such a huge data is very challenging. It requires lot of storage, computational power and resources. These challenges make legacy databases inefficient and insufficient for processing techniques. Even after replacing the data servers with high processing servers many organizations are not able to solve this huge data problem beyond certain level.

As Hadoop is designed with scale-out approach by using commodity hardware rather than traditional scale-in approach, data storage and maintenance became very cost effective as compared to different storage mechanisms.Performing efficient large-scale distributed computation is not that much easy. It requires parallel processing and distribution parts of the main problem to multiple machines. As Hadoop uses commodity hardware, the data is distributed and for its efficient parallel processing Map Reduce technique is introduced.

A single machine may have a few gigabytes of memory. But if the data size is in the extent of many terabytes, then thousands of machines with powerful RAMs may be required. Even then also it will be impossible for that machine to process or address all of the data.Huge amount of Intermediate data is also generated while performing a large-scale computation. It also takes some space to store. Again give rise to memory shortage problems. The overflow data requires other nodes to store data. A large-scale distributed system must consider this problem and should be able to manage the resources efficiently. Also such system must spend more time on actual core computation [1,2,3,4].
In H2Hadoop technique, if the data block isn't present in HDFS then only Map Reduce operations are performed and results are stored in HDFS. If the data block to be uploaded is already processed previously and results are already there in HDFS, then the same results are used again. For this the Name Node is retaining extra information about previous processing [12,17,19,20]. It outcomes in much less number computations, resources saving and quicker processing.

II. HADOOP

Hadoop daemons interactions are shown in Figure 1. In order to store the data Client requests NameNode. In response NameNode sends IP address of the DataNode. Client formats raw data in HDFS format and divides them into different data blocks. These data blocks are stored on different DataNodes [5,6,20].

Revised Manuscript Received on December 22, 2018.
Sachin Arun Thanekar, PhD Scholar, Department of Computer Science & Engineering, KLEF, Vaddeswaram, Guntur (A.P.), India.
Kodukula Subrahmanyam, Professor, Department of Computer Science & Engineering, KLEF, Vaddeswaram, Guntur (A.P.), India.
AliAkbar Bagwan, Professor, Department of Computer Engineering, RSCOE, Tathwade, Pune (Maharashtra), India

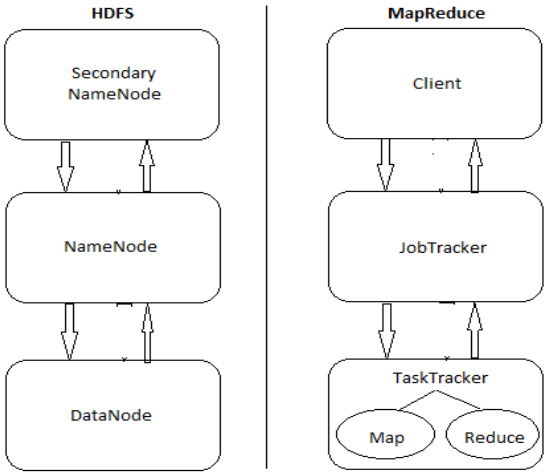

Fig. 1 Hadoop daemons interaction

By default hadoop maintains three replicas of same data. The first copy is always stored on processing DataNode itself, second copy is preferably stored on another DataNode in the same rack and third copy is preferably stored on some another DataNode in some different rack. Client sends a mapreduce job with source file name to the Job Tracker. Job tracker sends it to those task trackers which have that required data blocks. Each task tracker finishes the required execution and sends results to the Job Tracker again. Then through Job Tracker client gets all the final results.

By default hadoop treats every job as an independent job even if client has another job which requires same data blocks. Thus even if results are already there in different DataNodes still all the computational steps are repeated by considering it as a new job. There is no provision made for checking the similarities of data blocks and what is already processed.Moreover metadata of previous jobs is also not maintained. For every request it simply reads data from all DataNodes. It results in weak Hadoop performance.

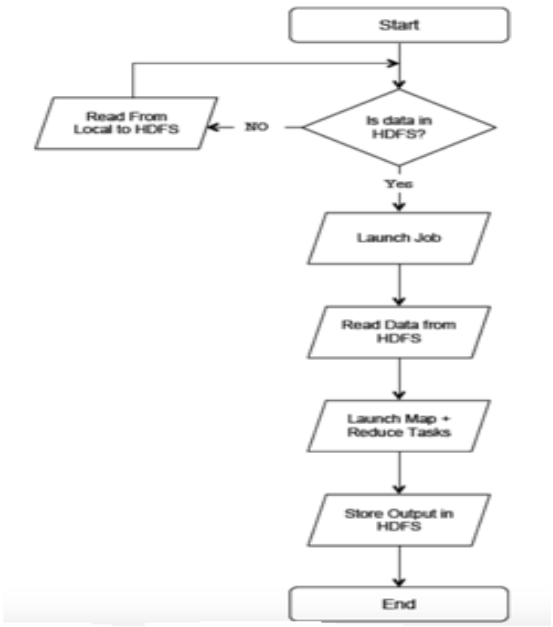

Fig. 2 Hadoop workflow[5]

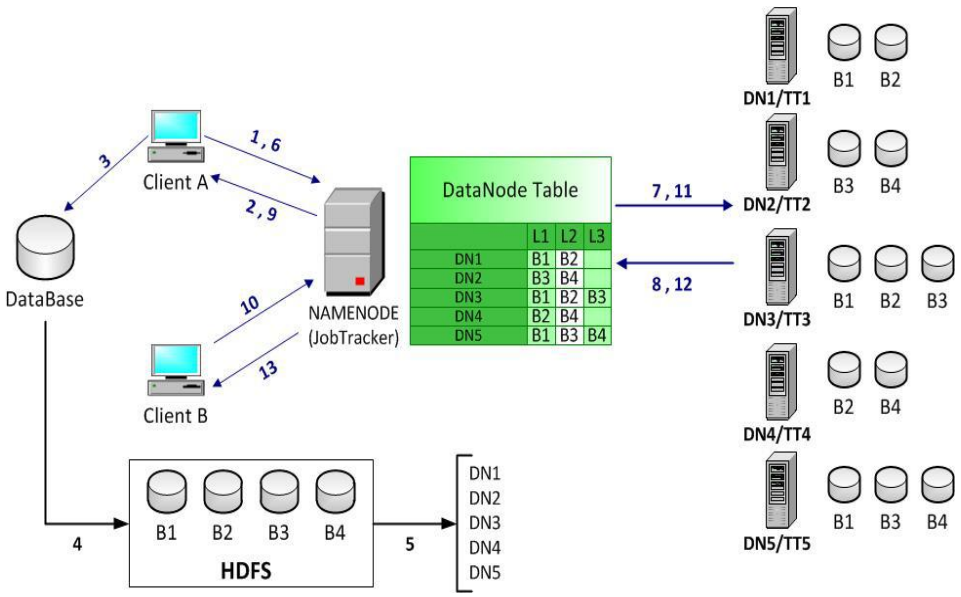

Fig. 3 Hadoop Map Reduce workflow[5]

As shown in figure 3 when client A stores a file then all the mapreduce computations are done and results are stored on DataNodes. When client B want to store same file again then also hadoop treats it as a completely new file again and repeats all mapreduce computations. Since every task is trated as an independent task results are not shared hence hadoop is spending time on same computations again for same data which is already processed earlier [7,10,14,21,22].

III. H2HADOOP

H2Hadoop solved the problem of task redundancy. NameNode is storing Common Jobs Blocks Table [CJBT] which saves data about finished jobs. [5,6, 8,9,11,15]. The workflow is as shown in figures 4 and 5,

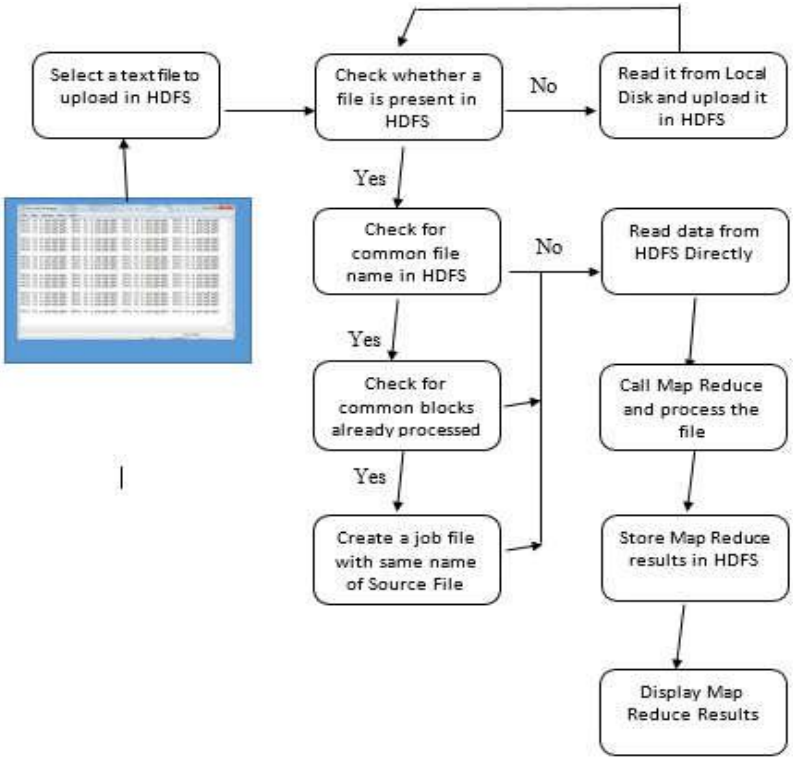

Fig. 4 H2Hadoop workflow

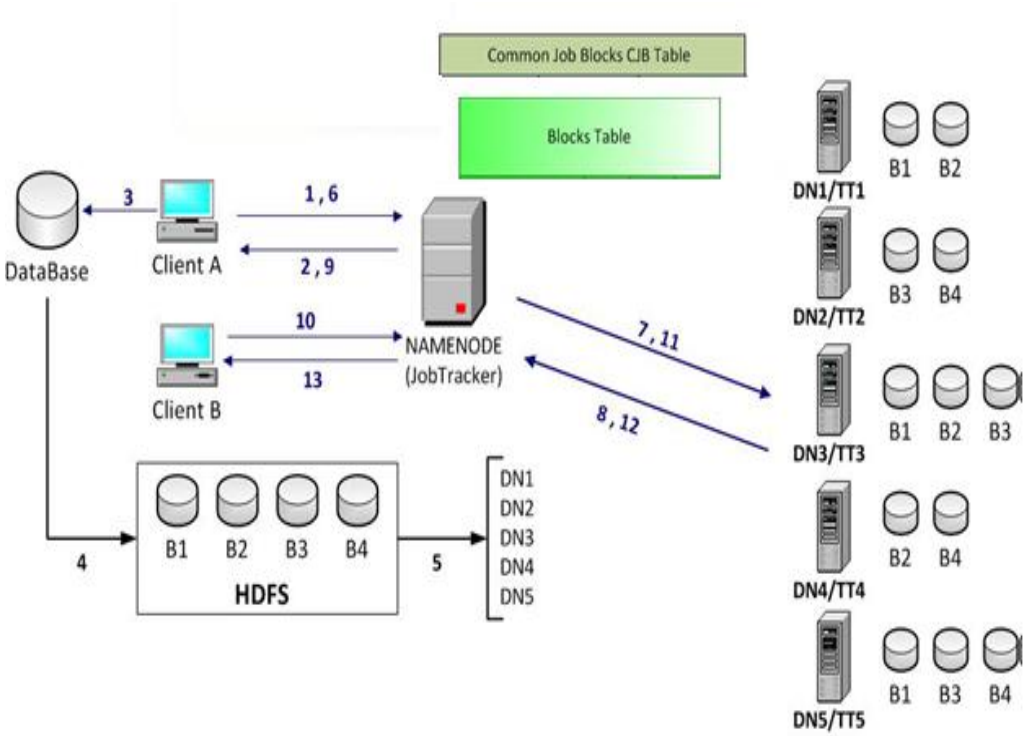

Fig. 5 H2Hadoop MapReduce workflow[5]

Thus from CJBT [13,16,17,18] only we can check whether similar file is already processed, blocks are fully/partially processed or not. It saves time for recomputations, resources. Also increases the speed of operations when some match is found. CJBT is dynamic in nature. In CJBT we are checking following things,

Jobs with common name, Jobs with common features, Block Names with Block Id

IV. PROPOSED SYSTEM

The uploaded big data files are partitioned in to number of blocks and are distributed over node clusters. To avoid random block distribution and data-duplication, deduplication system is used. Such deduplication system focuses on space management and only keeps track of data files on HDFS. Such system do not participate in efficient job execution in map reduce environment.

In the era of big data we all are facing storage problems of huge data generated and that to in very very short time. HDFS is also experiencing the same. In order to store this huge increasing data efficient storage management is the need. To improve the storage efficiency a dynamic deduplication decision algorithm is a solution [23]. Using personal devices or any other servers, users can upload data in HDFS. A two-tier deduplication technique is used. i.e. pre-filter and post-filter. Pre-filter checks file level deduplication and post- filter checks block level deduplication.

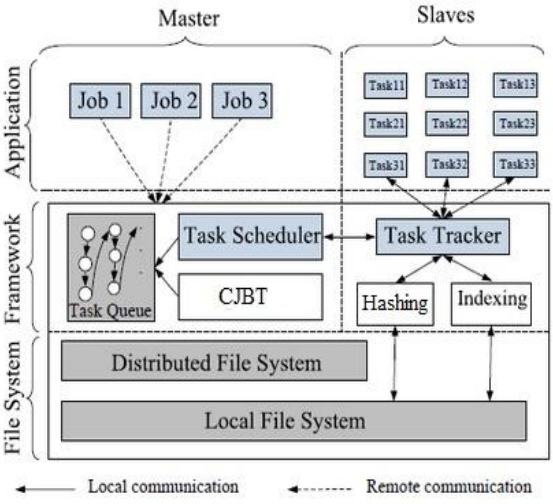

Fig. 6 Proposed system framework

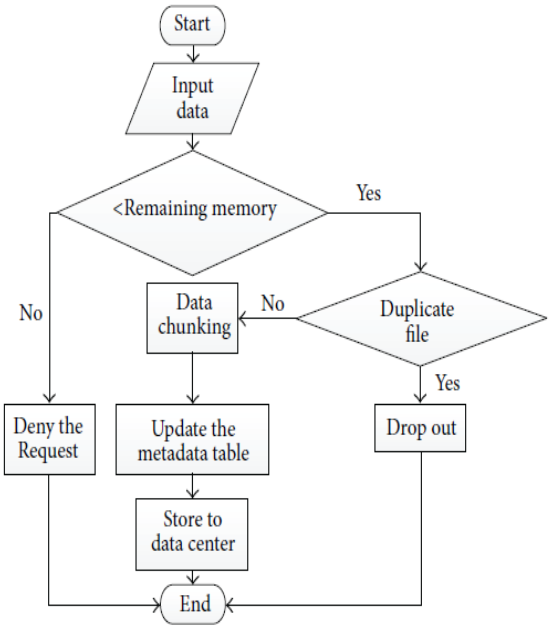

Fig. 7 Pre-filtering deduplication [23]

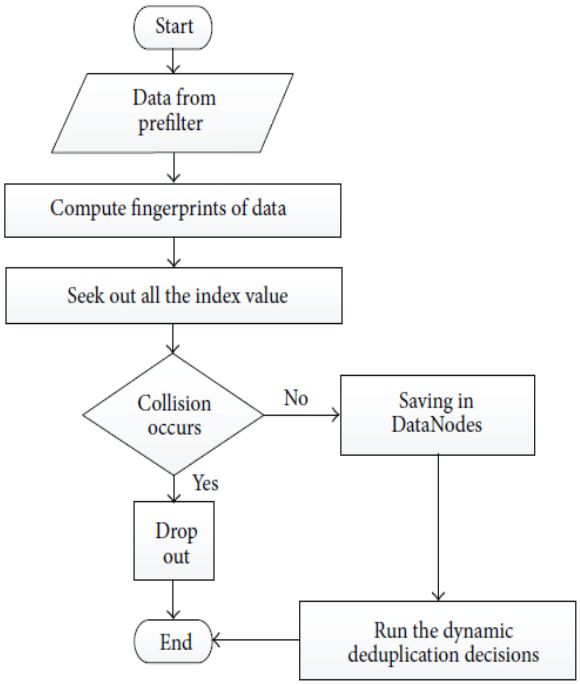

Fig. 8 Post-filtering deduplication [23]

For efficient execution of job, data locality information and job metadata is stored. Time required for job execution can be decreased for next execution of same job by preserving job metadata. The job execution environment does not keep track of uploaded data. It simply preserves metadata of executed jobs.

A combined environment produce efficient job execution results with efficient space management[13,16,17]. The data node executes the allocated job over data and preserves the job and data execution history. The name node checks whether data is already processed with same job or not. If yes, it checks the data node history and get the results from data node without re-executing the same task. The duplication work of result generation is avoided called as task de-duplication.To execute this task, system preserves table data structure. It includes: Job name, User, Access specification, Data node name, Dataset details, Result location. Name node initially work on De-duplication checking. If De-duplication found name node only collects the results from respective data nodes.

V. RESULTS

We implemented the system using Hadoop and H2hadoop framework. When the file is submitted for processing first it checks file level duplication using hash id. If hash id is already there in the preprocessing table then previously calculated results are shown. If hash id is not matching then blocks are created and block ids are checked. If match found that block is not processed again. Thus saves time for recompuations for same data. The comparative results are as shown in figure 9 and figure 10.

Table. 1 Hadoop Results

File Name	File Size (KB)	MapReduce Computations Time required (In seconds)	Time required to Get Results on Screen (In Seconds)
sachin.txt	53	18	19
tested.txt	60	23	25
t.txt	70	28	30
shruti.txt	90	30	33
srushti.txt	100	36	40
shravan.txt	120	40	43

Table. 2 H2Hadoop Results

File Name	File Size (KB)	Time required to Get Results on Screen in Hadoop (In Seconds)	Time Required to Get results on screen in H2Hadoop (In seconds)
sachin.txt	53	19	04
tested.txt	60	25	05
t.txt	70	30	05
shruti.txt	90	33	04
srushti.txt	100	40	04
shravan.txt	120	43	05

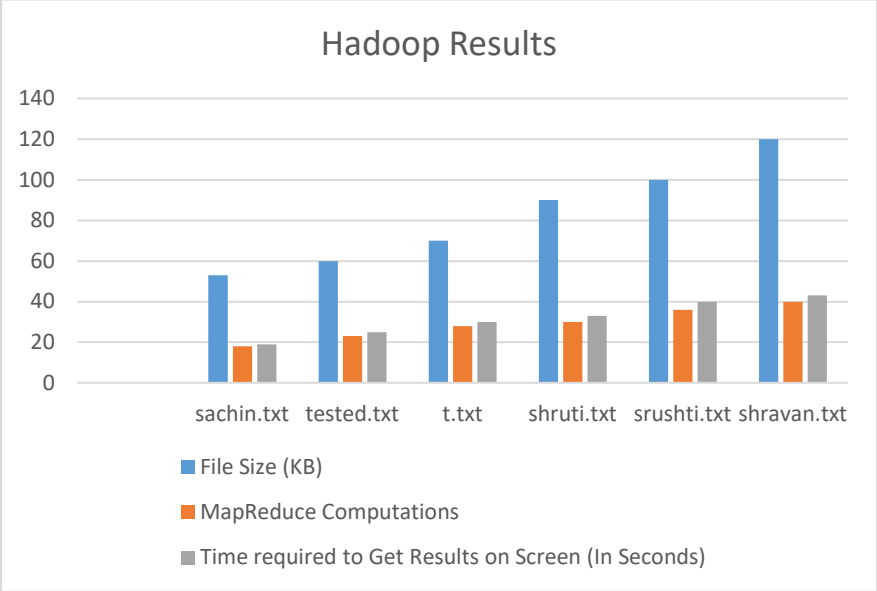

Fig. 9 Hadoop results

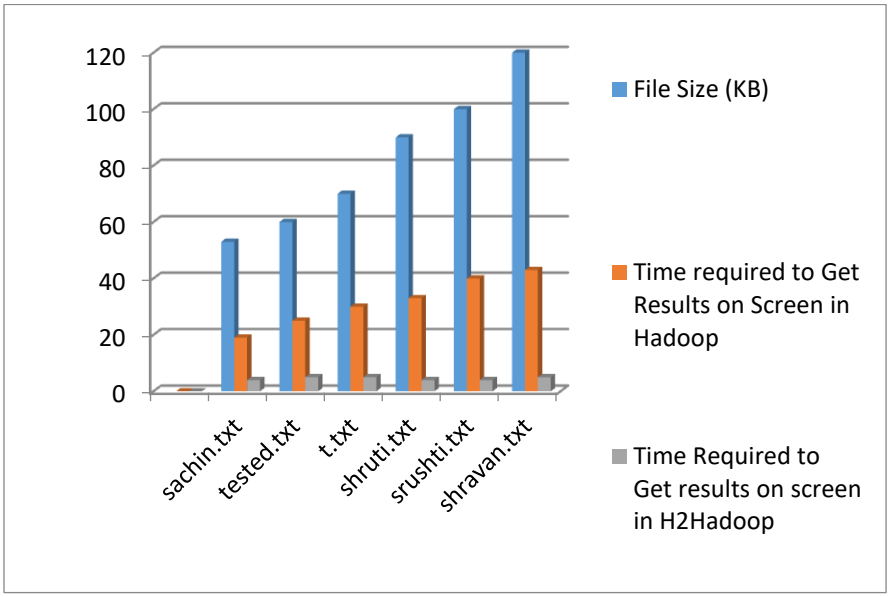

Fig. 10 H2Hadoop results

Table. 3 Deduplication Results

File Name	File Size (KB)	File Level Duplication Time (sec)	Block Level Duplication Time (sec)
sac.txt	525	164	10829
sup.txt	1030	183	20295
sru.txt	1550	219	29990
shr.txt	2010	289	45830
shv.txt	2520	361	61089
swa.txt	3070	473	79673

As shown in the tables 1 and 2, when the file is submitted for the first time the computations are done as per the normal Hadoop framework work. When same file is submitted again then actual H2hadoop work starts. So for the first time computations it takes same time in normal Hadoop and h2hadoop. The above results are just the comparative results of Hadoop and H2hadoop. The more detailed results are as shown in Table 3. It is very clear that when duplication ratio increased the time required for uploading the file in HDFS is also decreased.

VI. CONCLUSION

The uploaded big data files are partitioned in to number of blocks and are distributed over node clusters. To avoid random block distribution and data-duplication, deduplication system is used. For efficient execution of job, data locality information and job metadata is stored. Time required for job execution is decreased for next execution of same job by preserving job metadata. A combined environment produced efficient job execution results with efficient space management.

REFERENCES

1. Sachin Arun Thanekar, K. Subrahmanyam, A. B. Bagwan, "Big Data and MapReduce Challenges, Opportunities and Trends", *International Journal of Electrical and Computer Engineering (IJECE)* Vol. 6, No. 6, pp. 2911~2919, December 2016.
2. SachinArunThanekar, K. Subrahmanyam, A. B. Bagwan, "A Study on Digital Forensics in Hadoop", International Journal of Contol theory and applications, 9(18), pp. 8927-8933, 2016.
3. SachinArunThanekar, K. Subrahmanyam, A. B. Bagwan, "Improving Hadoop Performance by Enhancing Name Node Capabilities", Fronteiras: Journal of Social, Technological and Environmental Science, v.6, n.2, may.-august. 2017, p. 1-8.
4. Nenavath Srinivas Naik, Atul Negi, VN Sastry, "Performance Improvement of MapReduce Framework in Heterogeneous Context using Reinforcement Learning", Elsevier ISBCC'15. 2015.
5. H. Alshammari; J. Lee; H. Bajwa, "H2Hadoop: Improving Hadoop Performance using the Metadata of Related Jobs," in IEEE Transactions on Cloud Computing ,vol.PP, no.99, pp.1-1
6. H. Alshammari, J. Lee and H. Bajwa, "Evaluate H2Hadoop and Amazon EMR performances by processing MR jobs in text data sets," 2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT), Farmingdale, NY, 2016, pp. 1-6.
7. Muhammad Idris, Shujaat Hussain, Maqbool Ali, ArsenAbdulali, Muhammad Hameed Siddiqi, Byeong Ho Kang and Sungyoung Lee, "Context-aware scheduling in MapReduce: a compact review", Concurrency and Computation: Practice and Experience Volume 27, Issue 17, pages 5332–5349, Dec. 2015
8. Afaf G. Bin Saadon and Hoda M. O. Mokhtar, "iiHadoop: an asynchronous distributed framework for incremental iterative Computations", Journal of Big Data, Springer, July 2017.
9. Arati W. Borkar , Dr. Sanjay T. Singh , "Improved MapReduce Framework using High Utility Transactional Databases", International Journal of Engineering Inventions Volume 3, Issue 12 (July 2014) PP: 49-55.
10. Shafali Agarwal, ZebaKhanam, " MapReduce: A Survey Paper on Recent Expansion", International Journal of Advanced Computer Science and Applications, Vol. 6, No. 8, 2015
11. H. Alshammari H. Bajwa L. Jeongkyu "Enhancing performance of Hadoop and MapReduce for scientific data using NoSQL database", Systems Applications and Technology Conference (LISAT) 2015.
12. ão Cunha, Catarina Silva, MárioAntunes, "Health Twitter Big Bata Management with Hadoop Framework ", Procedia Computer Science 64 (2015) 425 – 431.
13. Pravin Sanap,BharatPatare, Ajay Waghmare, MukeshRathod, SnehalMulay, "Enhanced Hadoop with Search and MapReduce Concurrency Optimization", International Journal of Pure and Applied Mathematics, Volume 114 No. 12 2017, 323-331
14. M. Santhana Lakshmi, D. Sandhiya, A. N. Thasneem, S. Sivashankari, "Data Partitioning for Minimizing Transferred using MapReduce", International Journal of Engineering Science and Computing, Volume 7 Issue No.4 , April 2017
15. Swapnali A. Salunkhe, Amol B. Rajmane, "A Survey on Performance and Security of Hadoop", International Journal of Science and Research (IJSR) , Volume 6 Issue 7, July 2017
16. K. Sridevi, Dr. I HemaLatha, " H2Hadoop: Metadata Centric BigData Analytics on Related Jobs Data Using Hadoop Pseudo Distributed Environment", International Journal of Scientific Research in Computer Science, Engineering and Information technology, Volume 2, Issue 6, 2017
17. Balaji Siva Jyothi, Dr. P. Radhika Raju, Dr.A.Ananda Rao, "Improving Performance of Map Reduce using DLAJS Algorithm", International Journal of Computer Trends and Technology (IJCTT) – Volume 61 Number 1 - July 2018
18. Ms. KalyaniPatil, V.V. Dakhode, " Scalability Analysis and Improvement of Hadoop over H2Hadoop for Big Data Analysis", IJSRD - International Journal for Scientific Research & Development Vol. 5, Issue 04, 2017
19. YalingXun, Jifu Zhang, Xiao Qin, Xujun Zhao, "FiDoop-DP: Data Partitioning in Frequent Itemset Mining on Hadoop Clusters", IEEE Transactions on Parallel and Distributed Systems, 2016.
20. https://www.oreilly.com/ideas/processing-frameworks-for-hadoop
21. MrudulaVarade, VimlaJethani, "Distributed meta data management scheme in HDFS", International Journal of Advanced Computer Science and Applications, 2015; 6(8).
22. B. Zhang, et al., "Self- configuration of the Number of concurrently Running Map Reduce Jobs in a Hadoop Cluster," ICAC 2015, pp.149-150, 2015
23. Ruay-Shiung Chang, Chih-Shan Liao, Kuo-Zheng Fan, and Chia-MingWu, "Dynamic Deduplication Decision in a HDFS", International Journal of Distributed Sensor Networks, 2014.