Effective Job Execution in Hadoop Over Authorized Deduplicated Data

Sachin Arun Thanekar*
Research Scholar, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, AP, India. E-mail: sachin.sangamner@gmail.com

K. Subrahmanyam
Professor, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, AP, India.

A.B. Bagwan
Professor, SPPU, Pune, India.

Received July 07, 2020; Accepted September 10, 2020
ISSN: 1735-188X
DOI: 10.14704/WEB/V17I2/WEB17043

Abstract

Existing Hadoop treats every job as an independent job and destroys metadata of preceding jobs. As every job is independent, again and again it has to read data from all Data Nodes. Moreover relationships between specific jobs are also not getting checked. Lack of Specific user identities creation and forming groups, managing user credentials are the weaknesses of HDFS. Due to which overall performance of Hadoop becomes very poor**.** So there is a need to improve the Hadoop performance by reusing metadata, better space management, better task execution by checking deduplication and securing data with access rights specification. In our proposed system, task deduplication technique is used. It checks the similarity between jobs by checking block ids. Job metadata and data locality details are stored on Name Node which results in better execution of job. Metadata of executed jobs is preserved. Thus by preserving job metadata re computations time can be saved. Experimental results show that there is an improvement in job execution time, reduced storage space. Thus, improves Hadoop performance.

Keywords

Hadoop, H2Hadoop, Deduplication, HDFS, Storage.

Introduction

In hadoop framework, the allocated task is executed with the help of multiple workers called as data nodes. The task is executed using map reduce work strategy. The

processing task is assigned to the name node of hadoop framework. The name node is responsible for assigning task called as job to the data nodes. This process is called as mapping. After receiving the task to the data node, data node processes the data and returns the execution result to the main name node. Main node collects the results from all workers and generates a final result copy. This process is called as reduce process. The repetitive, same job allotment takes equal execution time and memory as before using map reduce strategy on hadoop framework. The name node and/or data node do not preserve the history of processing [1],[2],[3],[4]. To overcome this limitation of existing hadoop framework a new system is proposed. The proposed system is extension to the existing map reduce workflow over hadoop framework. This extension improves system work execution efficiency in terms of time and memory.

As compared to native Hadoop, H2Hadoop results in reduced CPU time, less read operations. Previous jobs data is not maintained in Hadoop, Without the knowledge of earlier processing it simply allocates Data Nodes. So every job is independent which again reads data from all Data Nodes. Also no arrangement is there for checking relationships between different jobs. Hence there is a scope to improve Hadoop performance [12],[17],[19],[20].

Deduplication system is used to avoid data-duplication and random block distribution. Space management is the main focus of this task. Job execution is handles by MapReduce.

Job metadata and data locality details are stored on Name Node which results in better execution of job. Executed jobs related metadata is preserved in a separate data structure on NameNode. Thus jobs are executed efficiently with better space management [13],[16],[17],[23]. An authorization environment is required to restricts unauthorized Data access, job execution by creating user identities.

So the objectives of our work are,

- To make storage space more effective by using de-duplication.
- To design a system to improve job execution performance using metadata over HDFS.
- To create user identities and groups for authorized data access and job execution over hadoop.

Background Knowledge

1) Hadoop

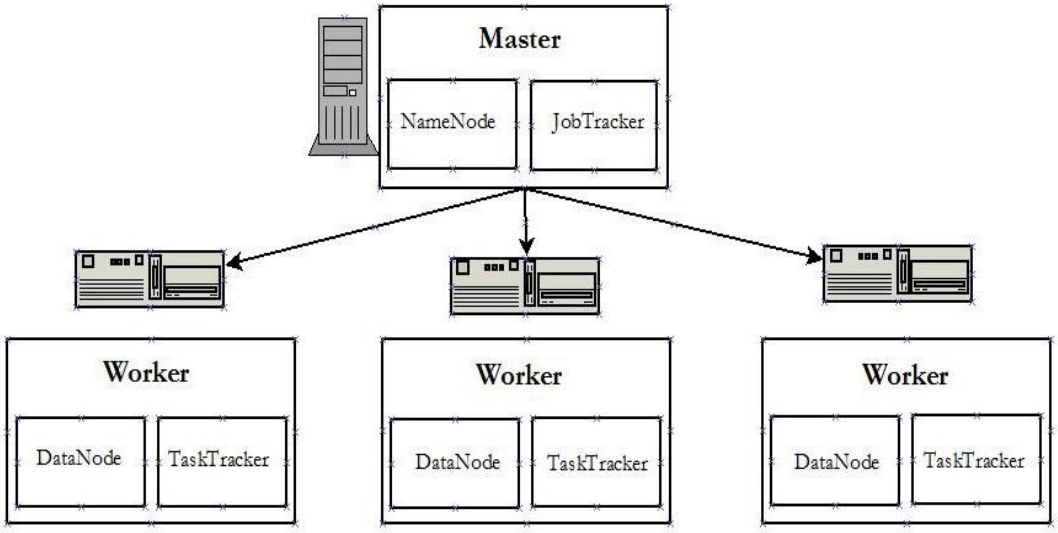

Fig. 1 Hadoop Daemons

For data storage Client requests NameNode and it replies with IP address of the DataNode. Raw data in converted into HDFS format and divided into different data blocks by Client. Then these blocks are stored on different DataNodes[5],[6],[20].

Job Tacker gets a mapreduce job with source file name from Client. Job tracker searches and send it to those task trackers which have that required data blocks. Then assigned task trackers finishes the required jobs execution and sends results to the Job Tracker through which Job Tracker client gets all the final results.

As shown in figure 2, every job is independent in Hadoop, it re execute the same job again even if the results are already calculated. The repetitive, same job allotment takes equal execution time and memory as before using map reduce strategy. The name node and/or data node do not preserve the history of processing. [7],[10],[14],[21],[22].

2) H2 Hadoop

In H2Hadoop, metadata of earlier finished jobs is stored on Namenode. This data is stored in dynamic data structure known as Common Jobs Blocks Table (CJBT) [5,6, 8,9,11]. The workflow is as shown in figure 3. While job is submitted for processing, first it is searched in CJBT whether similar file is processed previously or not [13,15,16,17,18]. Thus re computations, resources can be saved. Operation speed also

increases. CJBT preserves Jobs with common name, Jobs with common features, Block Names with Block Id.

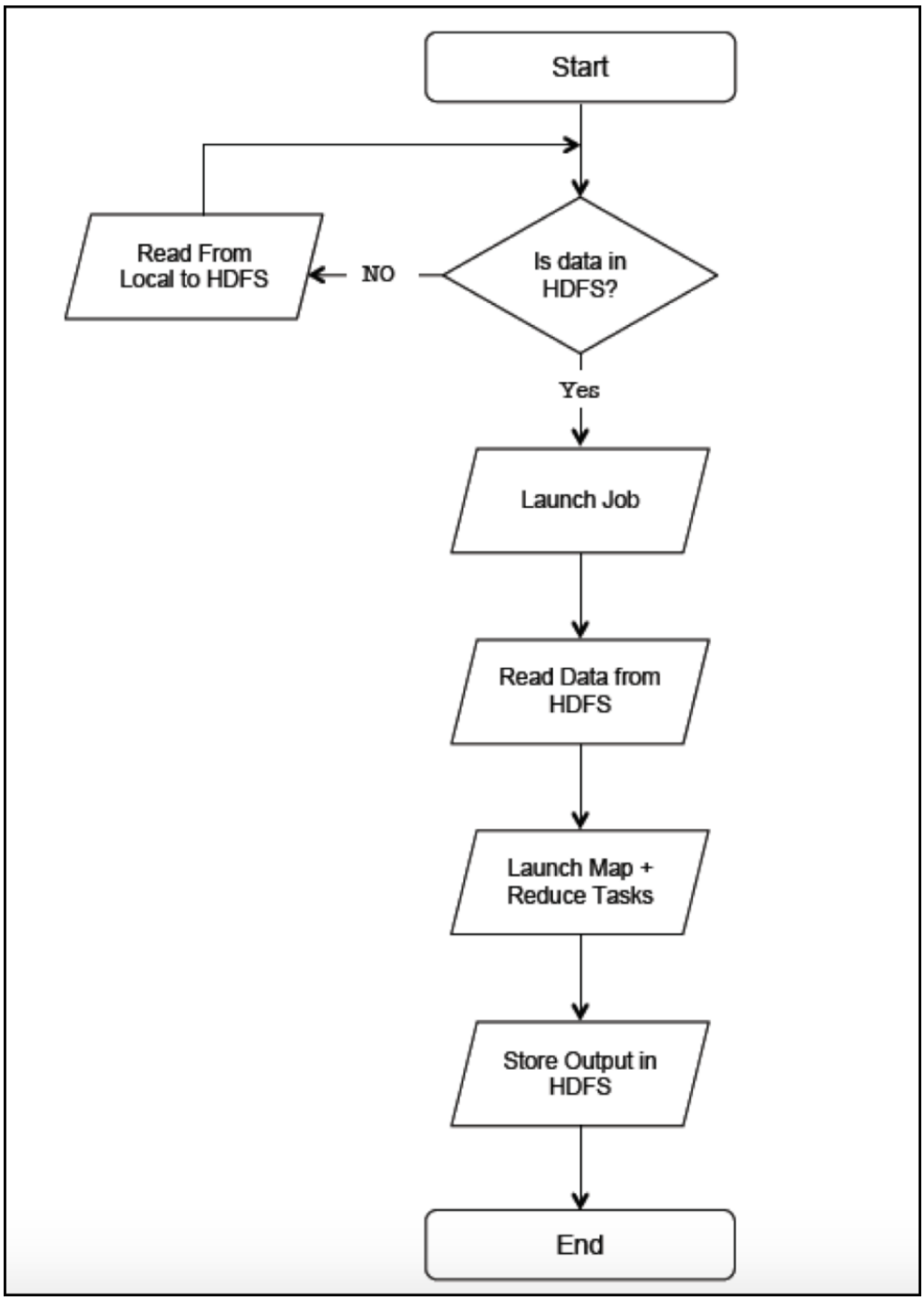

Fig. 2 Hadoop Workflow

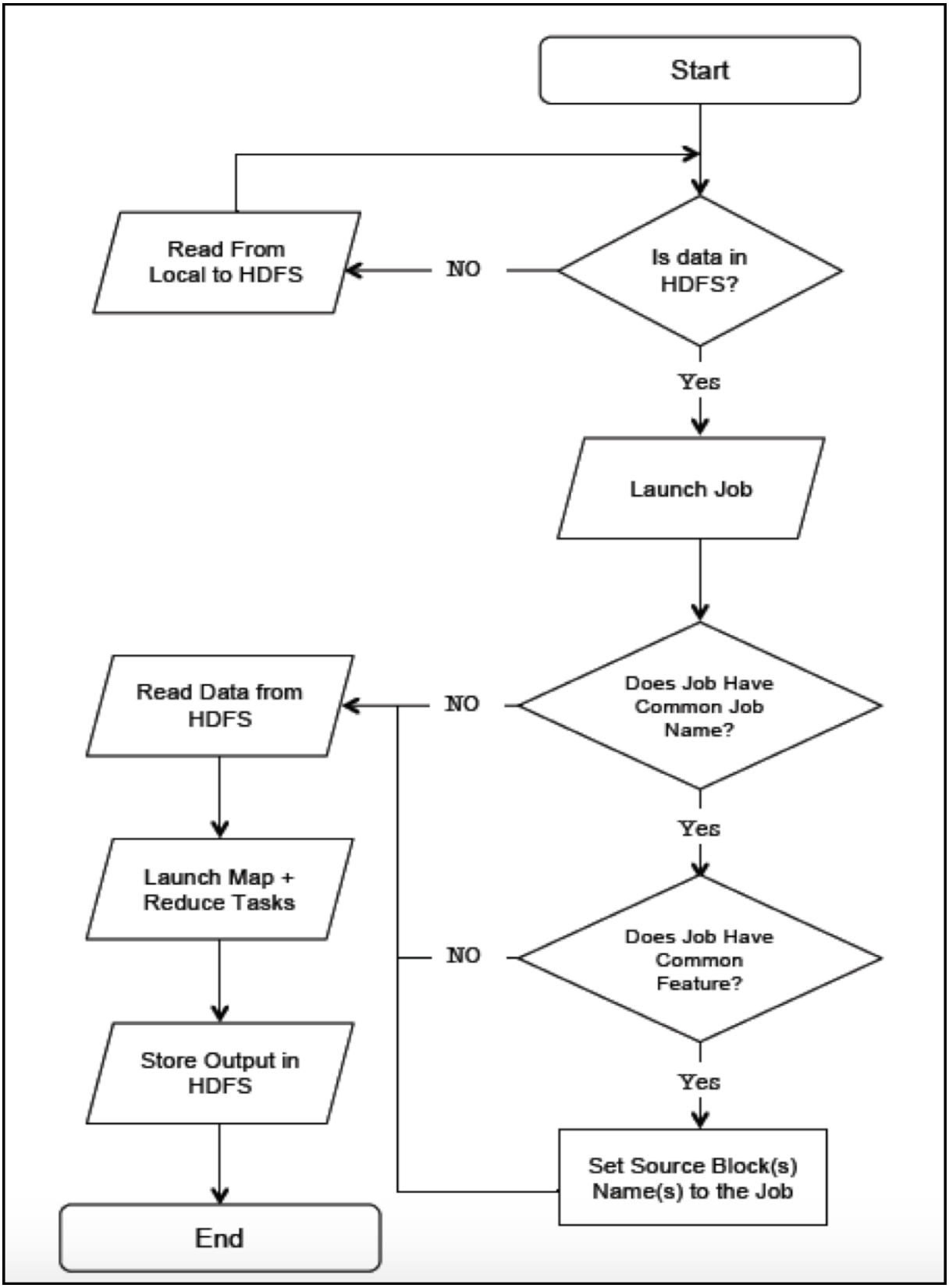

Fig. 3 H2Hadoop Workflow

Proposed Work

As shown in figure 4, the system has following three main functionalities:

- Data Uploading and Authorization
- Job Execution
- Task de-duplication checking similarities over data

1) Data Uploading and Authorization

In data uploading data deduplication checking and data authorization policies are constructed whereas in job execution, efficient task allotment and execution is done [24]. In the following section detailed description of these functionalities is given.

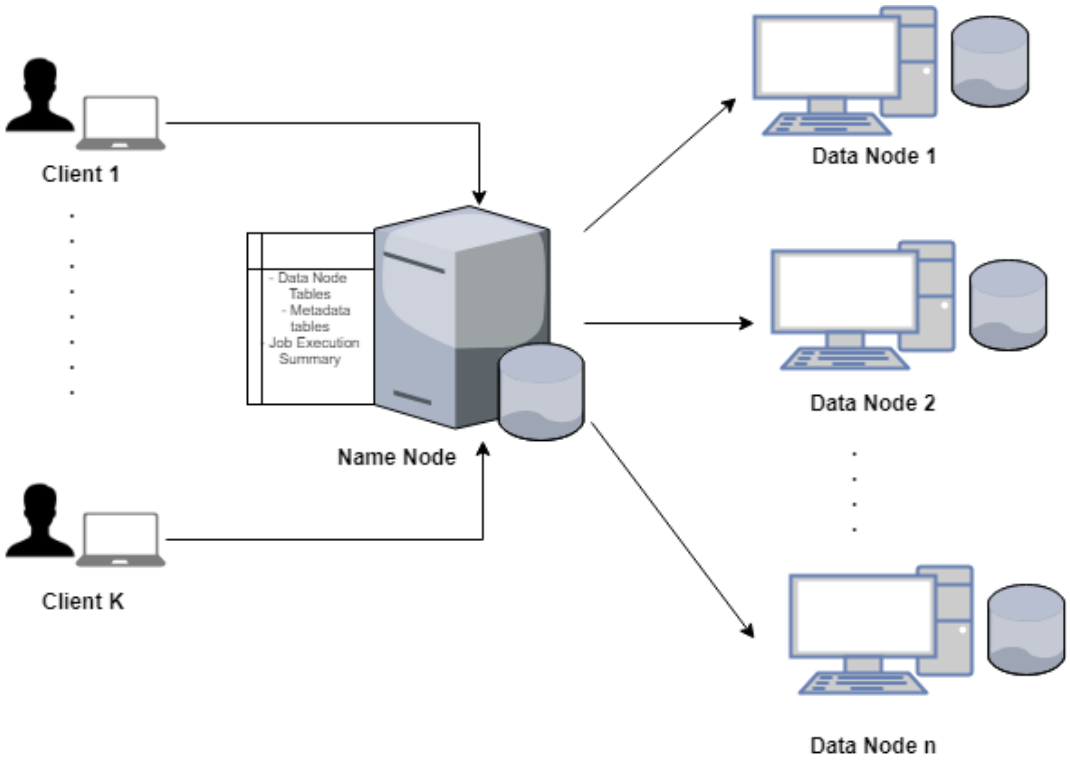

Fig. 4 Architecture of proposed system

File Level (Pre-filtering) Deduplication Algorithm

- Input: text file
- Output: divided data in the form of blocks or no output if match found
- Metadata table stores block details

When file is submitted,

Check attributes of the file
If these attributes matches with metadata table,
Then no need to process, skip the process
Else divide data into blocks;
Store new files attributes in metadata table;
Inform Data node to store data;
End

Block Level (Post-filtering) Deduplication Algorithm

- Input: text file divided in the form of blocks and their details
- Output: divided data in the form of blocks or no output if match found
- Metadata table stores block details with hash values of blocks

When receives output from file level Compute hash function
Check this with index table, if match found skip further process
Else inform DataNode to store new blocks;
End

2) Job Execution

File Level (Pre-filtering) Deduplication Algorithm

- Input: text file
- Output: divided data in the form of blocks or no output if match found
- Metadata table stores block details

When file is submitted,
Check attributes of the file
If these attributes matches with metadata table,
Then no need to process, skip the process
Else divide data into blocks;
Store new files attributes in metadata table;
Inform Data node to store data;
End

Block Level (Post-filtering) Deduplication Algorithm

- Input: text file divided in the form of blocks and their details
- Output: divided data in the form of blocks or no output if match found
- Metadata table stores block details with hash values of blocks

When receives output from file level Compute hash function
Check this with index table, if match found skip further process
Else inform DataNode to store new blocks;
End

As shown in figure 5, first the whole file identity is checked, if match found then process ends there and the previously calculated results are used. If match not found then file data is divided into blocks, hash values of these newly created blocks compared with the previously stored values in the metadata table. If match found then those blocks will not be stored again. Only new blocks which are not previously processed or stored will be stored and accordingly metadata table entries will be updated.

User Rights Specification

After uploading the file user can specify rights for file access. The rights includes read, write and execute permission. User can assign file permission to single user or can assign rights to the group of users. Based on the assigned permission user can be able to execute the job on the given files.

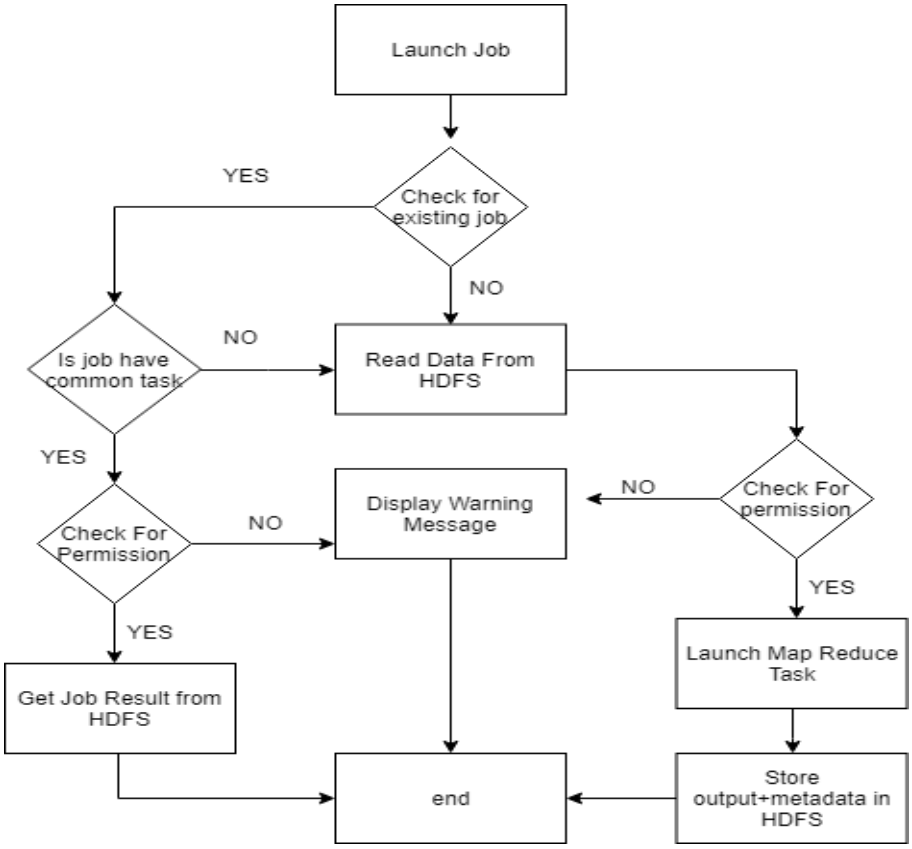

Fig. 5 Proposed System WorkFlow

3) Task De-duplication Checking Over Data

Here with the help of CJBT similarities between multiple jobs are found out with following attributes, Job name, User, Data node name, Dataset details. The similar blocks will not be processed again which are common in the two different tasks. Thus repreocessing of similar blocks is reduced and similar blocks will be processed and stored only once.

Result Analysis

Experiment Setup

We have used Ubuntu 16.04, Hadoop 2.6.0, Java 1.7.0, MapReduce Paradigm with One Name node and Three Data Nodes configuration. The nodes are having Intel I3 Processors, 2 GB RAMS, 500 GB HDDs.

Dataset Description

We have used 20_Newsgroup dataset. Different news topics are organized in 20 different newsgroups which includes similar, dissimilar or partial matching news.

Results

Table 1 File Level Deduplication (Block Size 100 KB)

File Size (MB)	Tag gen. Time(ms)	File Dup. Check Time (ms)	Block Dup. Check time(ms)	Upload Time (ms)	Total Processing Time (ms)
1	66	204	881	20384	21539
2	152	515	1477	36532	38718
3	160	20	2289	51403	53956
4	128	11	3167	68747	72073

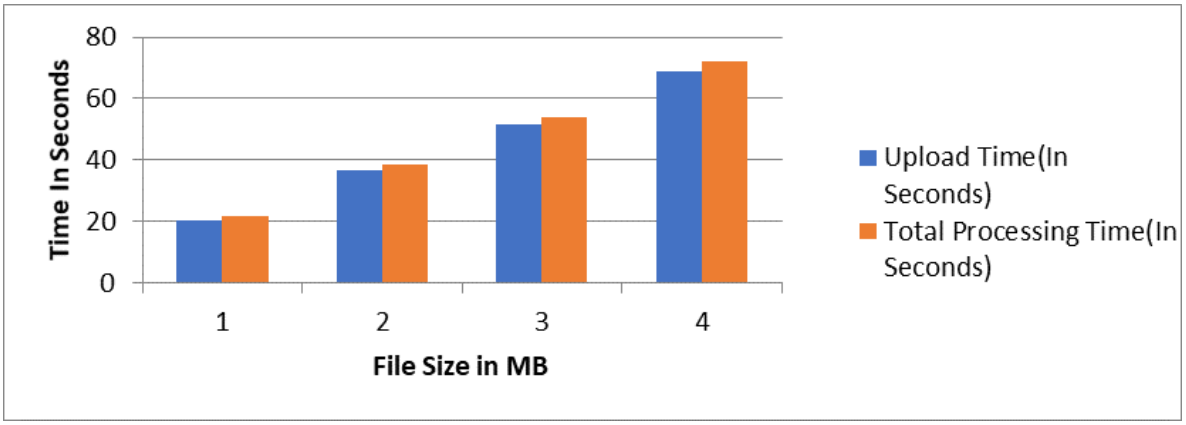

Fig. 6 File level Deduplication

From Table 1 and Fig.6 it is verified that the Tag generation time increases as file size increases.

File dedup check time is independent of file size. As file size increases number of blocks also increases and hence Block Dup. Check time also increases. Upload and total processing time also increases as file size increases.

Table 2 File Level Percentage Deduplication (File Size 4 MB)

Dup. Type	Tag gen. Time (ms)	File Dup. Check Time (ms)	Bolck Dup. Check time (ms)	Upload Time (ms)	Total Processing Time (ms)
0%	**128**	**11**	**3167**	**68747**	**72073**
25%	**108**	**8**	**2063**	**50114**	**52295**
50%	**67**	**8**	**1332**	**36300**	**34890**
75%	**92**	**35**	**789**	**21125**	**22045**
100%	**45**	**14**	**1**	**0**	**66**

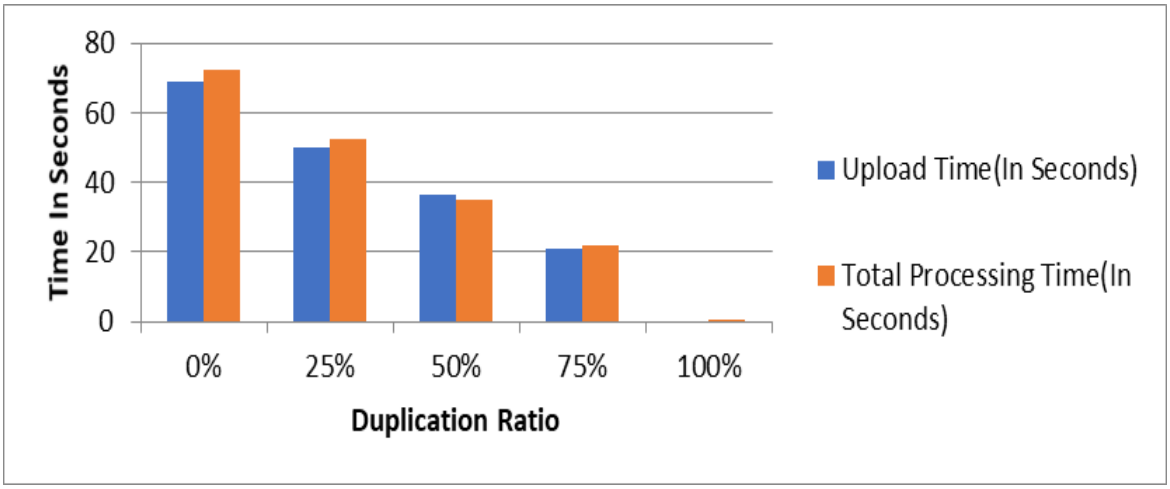

Fig. 7 File level Percentage Deduplication

Here we kept file size constant 4 MB. From Table 2 and Figure 7 it is verified that Duplication ratio increases Upload and total processing time decreases.

Table 3 File Level Variable Block Size (File Size 4 MB)

Block Size (kb)	Tag gen. Time(ms)	File Dup. Check Time(ms)	Block Dup. Check time(ms)	Upload Time(ms)	Total Processing Time(ms)
50	131	40	5457	140781	146414
100	128	11	3167	68747	72073
150	129	16	2040	55746	57934
200	134	34	1659	47229	49071

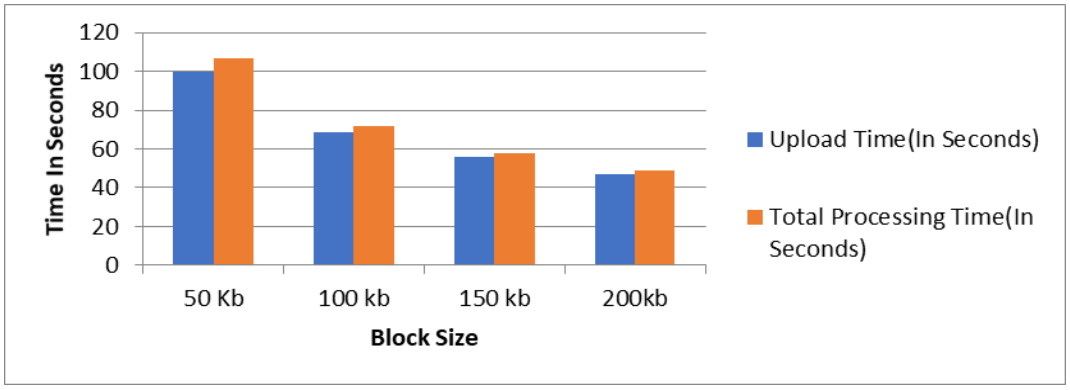

Fig. 8 File level Deduplication with variable block size

From Table 3 and Figure 8 it is verified that, as block size increases number of blocks decreases and hence processing time decreases. Moderate block size selection is important aspect.

Table 4 Job Level Deduplication (Block Size 100 KB)

File Size (MB)	Dedup check time (ms)	Job exe. Time(ms)	Total Processing Time(ms)
1	67	17712	17867
2	14	29474	29835
3	15	41131	41222
4	10	56047	56141

Fig. 9 Job Level Deduplication with 100 KB block size

From table 4 and figure 9 it is verified that, as file size increases number of blocks also increases and hence job execution time also increases. The job deduplication time is constant irrespective of file size.

Table 5 Job Level Percentagewise Deduplication (File size 4 MB)

Dedup type	Dedup check time(ms)	Job exe. Time(ms)	Total Processing Time(ms)
0%	10	56047	56141
25%	13	38630	38648
50%	38	25958	26006
75%	35	14834	14880
100%	57	0	70

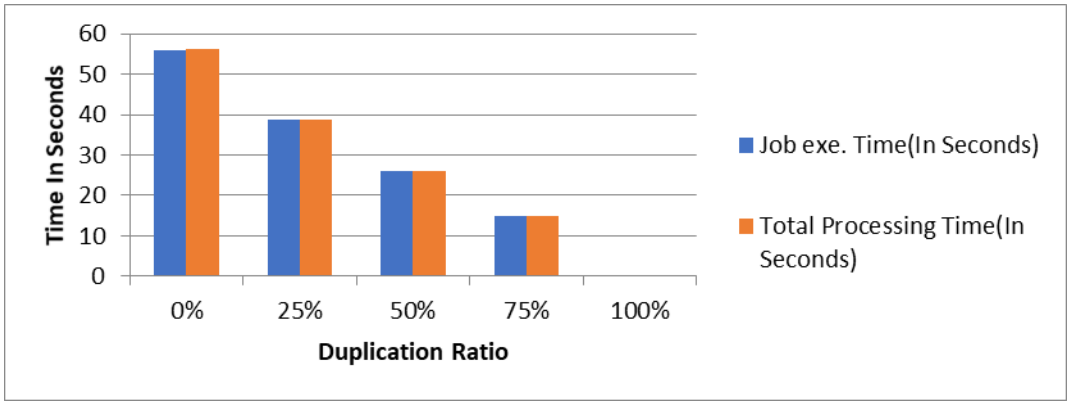

Fig. 10 Job Level Percentagewise Deduplication

From table 5 and figure 10 it is verified that Duplication ratio increases job execution time decreases.

Table 6 Job Level Variable Block Size (File size 4 MB)

Block Size(KB)	Dedup check time(ms)	Job exe. Time(ms)	Total Processing Time(ms)
50	55	106873	106935
100	10	56047	56141
150	57	36498	36564
200	128	31149	31600

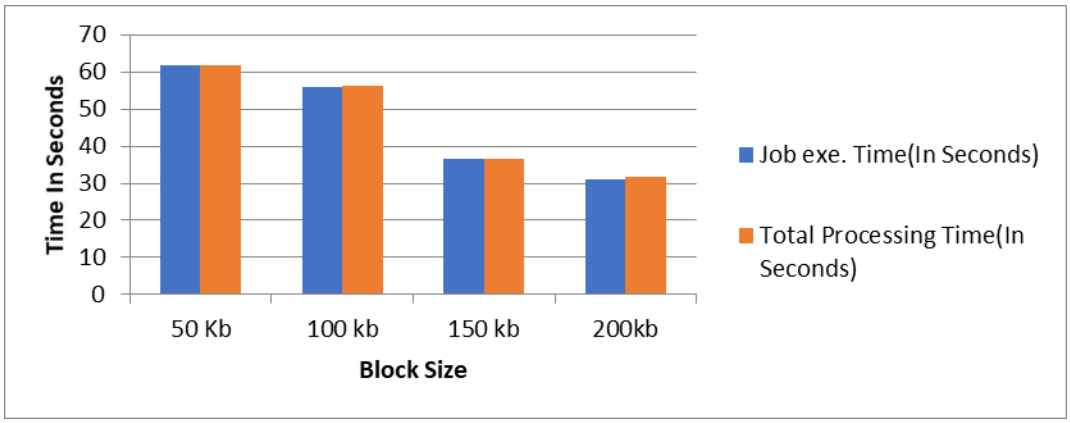

Fig. 11 Job Level Variable Block Size

From table 6 and figure 11 it is verified that as block size increases number of blocks decreases and hence processing time decreases. Moderate block size selection is important aspect.

Conclusion

As every job is independent in Hadoop, again and again it has to read data from all Data Nodes. Moreover relationships between specific jobs is also not getting checked. In our proposed system, task deduplication technique is used. It checks the similarity between jobs by checking block ids. Job metadata and data locality details are stored on Name Node which results in better execution of job. Metadata of executed jobs is preserved. Thus by preserving job metadata re computations time is saved. Experimental results show that there is an improvement in job execution time, reduced storage space. Thus, improved Hadoop performance.

References

Sachin, A.T., Subrahmanyam, K., & Bagwan, A.B. (2016). Big Data and MapReduce Challenges, Opportunities and Trends. *International Journal of Electrical and Computer Engineering (IJECE)*, *6*(6), 2911-2919.

Sachin, A.T., Subrahmanyam, K., & Bagwan, A.B. (2016). A Study on Digital Forensics in Hadoop. *International Journal of Contol theory and applications*, *9*(18), 8927-8933.

Sachin, A.T., Subrahmanyam, K., & Bagwan, A.B. (2017). Improving Hadoop Performance by Enhancing Name Node Capabilities. *Fronteiras: Journal of Social, Technological and Environmental Science*, *6*(2), 1-8.

Naik, N.S., Negi, A., & Sastry, V.N. (2015). Performance improvement of MapReduce framework in heterogeneous context using reinforcement learning. *Procedia Computer Science*, *50*, 169-175.

Alshammari, H., Lee, J., & Bajwa, H. (2016). H2hadoop: Improving hadoop performance using the metadata of related jobs. *IEEE Transactions on Cloud Computing*, *6*(4), 1031-1040.

Alshammari, H., Lee, J., & Bajwa, H. (2016). Evaluate H2Hadoop and Amazon EMR performances by processing MR jobs in text data sets. *In IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, 1-6.

Muhammad, I., Shujaat, H., Maqbool, A., Arsen, A., Muhammad, H.S., Byeong, H.K., & Sungyoung, L. (2015). Context-aware scheduling in MapReduce: a compact review. *Concurrency and Computation: Practice and Experience*, *27*(17), 5332–5349.

Saadon, A.G.B., & Mokhtar, H.M. (2017). iiHadoop: an asynchronous distributed framework for incremental iterative computations. *Journal of Big Data*, *4*(1), 24.

Arati, W.B., & Sanjay, T.S. (2014). Improved MapReduce Framework using High Utility Transactional Databases. *International Journal of Engineering Inventions*, *3*(12): 49-55.

Agarwal, S., & Khanam, Z. (2015). Map reduce: a survey paper on recent expansion. *International Journal of Advanced Computer Science and Applications*, *6*(8), 209-215.

Alshammari, H., Bajwa, H., & Lee, J. (2015). Enhancing performance of Hadoop and MapReduce for scientific data using NoSQL database. *In Long Island Systems, Applications and Technology*, 1-5.

Cunha, J., Silva, C., & Antunes, M. (2015). Health twitter big bata management with hadoop framework. *Procedia Computer Science*, *64*, 425-431.

Pravin, S., Bharat, P., Ajay, W., Mukesh, R., & Snehal, M. (2017). Enhanced Hadoop with Search and MapReduce Concurrency Optimization. *International Journal of Pure and Applied Mathematics*, *114*(12), 323-331.

Santhana Lakshmi, M., Sandhiya, D., Thasneem, A.N., & Sivashankari, S. (2017). Data Partitioning for Minimizing Transferred using MapReduce. *International Journal of Engineering Science and Computing*, *7*(4).

Swapnali, A.S., & Amol, B.R. (2017). A Survey on Performance and Security of Hadoop. *International Journal of Science and Research (IJSR)*, *6*(7), 51-54.

Sridevi, K., & Hema Latha, I. (2017). H2 Hadoop: 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*, *2*(6), 834-841.

Balaji, S.J., Radhika Raju, P., & Ananda Rao, A. (2018). Improving Performance of Map Reduce using DLAJS Algorithm. *International Journal of Computer Trends and Technology (IJCTT)*, *61*(1).

Kalyani, P., & Dakhode, V.V. (2017). Scalability Analysis and Improvement of Hadoop over H2Hadoop for Big Data Analysis. *IJSRD - International Journal for Scientific Research & Development*, *5*(4).

Xun, Y., Zhang, J., Qin, X., & Zhao, X. (2016). FiDoop-DP: Data partitioning in frequent itemset mining on hadoop clusters. *IEEE Transactions on parallel and distributed systems*, *28*(1), 101-114.

https://www.oreilly.com/ideas/processing-frameworks-for-hadoop

Mrudula, V., & Vimla, J. (2015). Distributed meta data management scheme in HDFS. *International Journal of Advanced Computer Science and Applications*, *6*(8).

Zhang, B. (2015). Self- configuration of the Number of concurrently Running Map Reduce Jobs in a Hadoop Cluster. *Independent Commissioner Against Corruption*, 149- 150.

Chang, R.S., Liao, C.S., Fan, K.Z., & Wu, C.M. (2014). Dynamic deduplication decision in a hadoop distributed file system. *International Journal of Distributed Sensor Networks*, *10*(4), 630380.

Sachin, A.T., Subrahmanyam, K., & Bagwan, A.B. (2019). Effective Utilization of Storage Space by Applying File Level and Block-Level Deduplication over HDFS. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, *8*(6S).

Authors

Sachin Arun Thanekar, received his B.E (Computer), M.E. (Computer). Degrees from Pune University. He is a Ph.D. scholar in CSE dept. of KLEF, Vaddeswaram, AP, India. His current interests include software testing, databases, big data and information security.

Dr.K. Subrahmanyam is a professor in CSE dept. of KLEF, Vaddeswaram, AP, India. His current interests include software engineering, software testing, big data, Cloud computing.

Dr.A.B. Bagwan is working as a Professor in Computer Engineering department of Siddhant College of Engineering, SPPU, Pune. His current interests include data Warehouse, data Mining, Algorithms and big data.