Available techniques in hadoop small file issue

M. B. Masadeh, M. S. Azmi, S. S. S. Ahmad
Faculty of Information and Communication Technology, Universiti Teknikal Malaysia Melaka, Malaysia

Article Info

Article history:

Received May 3, 2019
Revised Oct 27, 2019
Accepted Nov 20, 2019

Keywords:

Big data
Big data in small files
EHDFS
Hadoop
HAR
HDFS
Mapreduce
Namenode
nHAR
Sequance file

ABSTRACT

Hadoop is an optimal solution for big data processing and storing since being released in the late of 2006, hadoop data processing stands on master-slaves manner that's splits the large file job into several small files in order to process them separately, this technique was adopted instead of pushing one large file into a costly super machine to insights some useful information. Hadoop runs very good with large file of big data, but when it comes to big data in small files it could facing some problems in performance, processing slow down, data access delay, high latency and up to a completely cluster shutting down. In this paper we will high light on one of hadoop's limitations, that's affects the data processing performance, one of these limits called "big data in small files" accrued when a massive number of small files pushed into a hadoop cluster which will rides the cluster to shut down totally. This paper also high light on some native and proposed solutions for big data in small files, how do they work to reduce the negative effects on hadoop cluster, and add extra performance on storing and accessing mechanism.

Corresponding Author:

Mohammad Bahjat Al-Masadeh,
Department of Electrical and Computer Engineering,
Universiti Teknikal Malaysia Melaka,
Jalan Hang Tuah Jaya, 76100 Durian Tunggal, Melaka, Malaysia.
Email: mbmasadeh@uqu.edu.sa

1. INTRODUCTION

In the past, big data producing and storing was limited on the high-profile companies and organizations such as the old yahoo, IBM, Amazon…etc [1]. These companies used to generate customers tracking files, logs, receipts and up to photo, videos on the top of their business, but since the smartphones revolution era, IoT (internet of things) [2], online availability and high internet connection, big data production being massively and rapidly out of control which made a need of a tool that can absorbs all of these data in order to regulate and insight them, is hadoop [3-5].
Big data known as a massive amount of data that's cannot be processing with the standard data management techniques [6], Big data characteristics stands on 5V's are:

a. Volume: is shown how much capacity that's big data could be.
b. Varity: many resources that big data are coming from.
c. Velocity: big data streaming and high transferring.
d. Veracity: Even if the data is 100% correct, there will be a dirty and incomplete data.
e. Value: is the data valuable and can be useful? [7, 8]

The world-wide data are limited into three types, the first type called structured data, is the SQL data or the data in tables such as SQL server, My SQL, Oracle …etc, this data covers only 20% of the whole data in the world, the second type is unstructured data are multimedia data, internet of things data, machines and sensors data, the third type is semi-structured data, this one acting like hybrid between structured and unstructured data, both of semi-structured and unstructured data are 80% of total data in the world [9-11].

2. HADOOP ARCHITECTURE

Hadoop architecture stands on master-slave manner in data processing and storing, the main node in hadoop called “Namenode” [1], this one knows everything happening across the cluster such as data transactions, data storing ID, data replications, other Datanode health and availability and much more technical jobs [12], in other words, Namenode holds every metadata about the cluster, on the other hand, the rest of nodes called “Datanodes” that’s save the actual data in a smaller unites called data blocks, every data block is 64 MB size [13]. The saving mechanism force the income data to occupy 80% of the block size as the rest of the data block is used for metadata purposes. Hadoop has a good data availability technique that’s every file in the cluster must has three replicas distributed in some other datanode in order to keep the file accessible even if the Datanode somehow goes down [4, 14, 15].

3. BIG DATA IN SMALL FILES DEFINITION

A big data, can be more than 1 TB, or a 500 GB of multimedia files such as videos, or else can be 10 TB of logs files, however, big data is not only a bunch of big files, it can be a group of small files related together in making sense. Generally, files with volume within 5 MB are called small file [16], but some others said that’s files are less than data block size (64 MB) is called a small files [17]. The small files can be logs, images, documents, voice records …etc.

4. SMALL FILE ISSUE

Due to hadoop is an open source [18], it was adopted by several companies under different names such as: HortonWorks distribution by yahoo, Cloudera from Facebook, Google, Oracle and Yahoo, Facebook corona by Facebook and some others [19, 20]. That’s means hadoop is a novel successful tool to treats Big Data problem, thus, during these long years, there were a problem flout on the surface called “big data in small files”.Namenode, is the master node of hadoop, this node stores all the meta data about datanodes, and all metadata about the data in the datanodes, on the other hand, datanodes are used to store the actual data, and due to Hadoop’s architecture, every single file must be replicated into 3 copies in the cluster in order of availability purposes. In Datanode memory, every 64 MB of memory called data block, if the stored file is less than the data block, it considers as a small file [17], this data block can hold one file only whether the file is 1 MB or 64 MB, every data block is known to the Namenode by metadata, every metadata size is 120B, thus, if one Datanode has 1000000 data blocks, the total size of metadata that sent already to Namenode is 120*1000000=120 MB/Datanode, if the total nodes in one hadoop cluster is 10000 nodes, this leads to 120 MB*10000=1.2 TB of metadata, this massive number of metadata cannot be handled by the Namenode which could leads to a dramatical shut down, and if Namenode goes down, the secondary Namenode will take the lead for a while then shutting down [19], and finally the wholly hadoop cluster will be use-less [21, 22].

5. SOLUTIONS

Of course, many research papers proposed some exits for this problem, some of these researches are already in use while the rest still documented:

- **HAR file:** Hadoop Archive was the first novel exit for big data in small file issue, basically HAR file used to pack the small files into HDFS blocks that’s can treats directly with Namenode. HAR file is created by running a hadoop archive command, means a Mapreduce job starts packing all the huge number of small files to be a small number of HAR files, thus treating with ten of HAR files is much more efficient and smoother rather than treating with thousands of small size files [1], and beside of this, HAR file doesn’t alter the architecture of HDFS. HAR is an excellent exit to avoid Namenode metadata jam, HAR file gives an option to access the files directly inside it as well as the creation steps done in easy commands, but on the other side, as a shortage, HAR cannot be altered after being created, cannot add more files, or delete some unwanted files from it. The bumpiest shortage in HAR is every file inside it requires 2 index files (Master index, Index) to read [23], that’s mean reading a file from HDFS it self is much easier than reading from HAR file. Another limitation of HAR is the memory; HAR files puts extra pressure on the file system due to generate a copy of the original files and takes space as much as they need [23].
- **nHAR file:** new Hadoop Archive is a correction of HAR file, its almost the same story but with some differences in architecture, first different is nHAR file needs only one index file to read, the second different is nHAR can be edit, you can add more files to the archive after create it. (Mir 2017). As well as HAR, nHAR is used to read files without altering HDFS architecture, but on the contrary of it, nHAR use

only one index to access the files instead, which mean more reading performance and efficiency, however, the reading mechanism is done based on creating a hash table instead of master index approach [24].

Figure 1 shown nHAR working mechanisim that's every small file 1…n, every file will be hashing and given an index number right before being saved inside a large file called "Part File". Hash table in nHAR is using instead of the index file that's adopted in HAR, to generate a hash code, the file name and the number of index file are used, then merging index files and moving to the part file, once part file being full, and new one will generate and continue filling [25]. Similar to HAR, the actual file will be stored to the part file, so the same problem with HAR that's the actual file will add more data load to HDFS whole storage, even this, nHAR easier and simpler than HAR in adding a new file to an existing nHAR file. Both of HAR and nHAR techniques are preventing to avoid altering HDFS architecture, the rest of this paper is shown another small file reduction technique but with HDFS architecture upgrade.

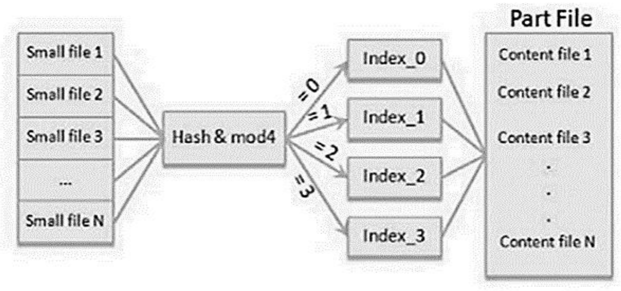

Figure 1. New archive technique [17, 24]

- **Improved HDFS:** This technique stands of two parts, the first one is client components which packing the small files into one big file, the second one is data node component which treating the resource management. Improved HDFS is an index based that's used to collect all small files of the same HDFS directory in one file, on the other side, the cash manager will resident in the Datanode.The packing and storing method are running based on the small files offset and length, every small file will store in one big file one by one, then determine the total sum of the small files in that big file and finally comparing the result with the data block size is 64 MB, thus, if the large file total length goes greater than the block size, then multiple block will used to save the file separately [24].

Figure 2 uncover the general file storing mechanism in HDFS that's only one file can be allocated in every data block, no matter if the file fits the block size or not, thus, every data block is connected to the Namenode via a metadata about it ID, Datanode location and some other info. As shown in Figure 2, once the file took place in the data block, the data block will be locked form inserting more files.

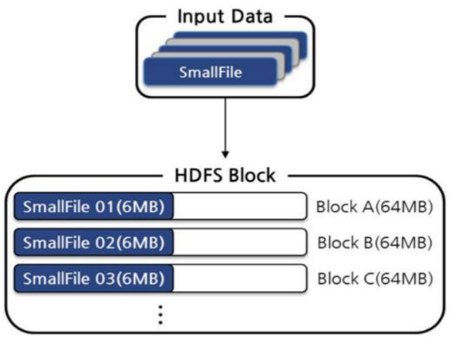

Figure 2. How the files being located in data blocks [26]

- **Sequence file:** Is another <key, value> technique for big data in small files, in a comparison with HAR, sequence file is faster in files access and file creation due to access a file inside a sequence file requires 1 index each. Sequence file can be update after being created, can add more small files depends on the block size. To create a sequence file, running a Mapreduce job is a must, but firstly, all the targeted small files must migrate to a respective block of Hadoop, then run a Mapreduce job to create the sequence file, however, Mapreduce job is required due to the sequence file architecture is based on <key, value>, which file name is the key, and file data is the value [27]. The Mapreduce job should maps all the small files till reaches the default block size and then pass it to reducer. The reducer will merge the files [28].

- **Map File:** Actually it's a two files, data file and index file, the data file consists all pairs <key,value>, and the key information are stored in index file, however, both of these two files are sequence files (Ahad and Biswas 2018), and acting like sequence file but map file doesn't looking for a entire file when search by key [23].
- **Extended HDFS**: EHDFS is another successful technique based on altering HDFS itself, however, all files are stored in a file called combine file in order to reduce the time in accessing files and reduce the load on Namenode, EHDFS stands on advance four indexing technique and prefetching of index information are file merging, file mapping, file prefetching and file extraction, the first file merging, file mapping and file prefetching are carried out by Namenode while the last one file extraction is carried out by Datanode [24]. As mentioned before, all of small files are stored in combine file, in the first stage (file merging), Namenode maintains a data only for this combine file, however, due to each data block can holds actual data size up to 80% of the block size and left the rest 20% for blocks metadata, there will be a hash table in the beginning of each block, this table holds an entry for each small file in the block in (offset, length), so based of this, the relation between block merging and combine file is the merged block is a part of the combine file. File mapping is the next stage of EHDFS, a request containing small file and combined file name is sent to the Namenode, for obtaining the location of the desired small file, in other words it's a process to match the small files names to the block of the combined file. File prefetching is just reducing the metadata usage in the Namenode. File extraction (carried by data block) used to extract the desired small file from the data block [24, 29].

6. CONCLUSION

There are a lot of big data in small files solutions, few of them use HDFS as is while the rest of others force to alter HDFS attitude, however, using HDFS as is would be a recomended choice to treats a pure hadoop, due to adding more customization will clear the small file issue, but on the other hand could add more complexity on hadoops work.

ACKNOWLEDGMENT

The authors thank the Ministry of Education for funding this study through the following grants: FRGS/1/2017/ICT02/FTMK-CACT/F00345. Gratitude is also due to Universiti Teknikal Malaysia Melaka and Faculty of Information Technology and Communication for providing excellent research facilities.

REFERENCES

[1] V. G. Korat and K. S. Pamu, "Reduction of Data at Namenode in HDFS using harballing Technique," vol. 1, no. 4, pp. 635–642, 2012.

[2] Y. Mo, "A Data Security Storage Method for IoT Under Hadoop Cloud Computing Platform," Int. J. Wirel. Inf. Networks, vol. 26, no. 3, pp. 152–157, 2019.

[3] G. Ramu, M. Soumya, A. Jayanthi, J. Somasekar, and K. K. Baseer, "Protecting big data mining association rules using fuzzy system," vol. 17, no. 6, pp. 3057–3065, 2019.

[4] M. Arslan, Z. Riaz, and S. Munawar, "Building Information Modeling (BIM) Enabled Facilities Management Using Hadoop Architecture," *2017 Portland International Conference on Management of Engineering and Technology (PICMET),* Portland, OR, pp. 1-7. 2017.

[5] T. Renner, J. Müller, L. Thamsen, and O. Kao, "Addressing Hadoop's Small File Problem With an Appendable Archive File Format," *Proceedings of the Computing Frontiers Conference,* pp. 367–372, 2017.

[6] E. G. Ularu, F. C. Puican, A. Apostu, M. Velicanu, and P. Student, "Perspectives on Big Data and Big Data Analytics," *International Journal of Production Economics, Elsevier,* vol. 191(C), pages 97-112, 2012.

[7] I. Ahmed, "A brief review: security issues in cloud computing and their solutions," *TELKOMNIKA (Telecommunication Comput. Electron. Control.),* vol. 17, no. 6, pp. 2812–2817, 2019.

[8] Pooja Sharma,Shimi S. L., S.Chatterji, "A Review On Obstacle Detection And Vision," *International journal of engineering sciences & research technology,* vol. 4, no. 1, pp. 2–5, 2015.

[9] M. April, I. No, B. P. Singh, and A. Agrawal, "Available Online at www.ijarcs.info Big Data : An Introduction," vol. 8, no. 3, pp. 21–23, 2017.

[10] G. Suciu, M. Anwar, I. Rogojanu, A. Pasat, and A. Stanoiu, "Big data technology for scientific applications," *11th ROLCG Conf. Grid, Cloud High-Performance Comput. Sci. - Proc.,* pp. 1–4, 2018.

[11] W. Fan, Z. Han, and R. Wang, "An Evaluation Model and Benchmark for Parallel Computing Frameworks," 2018.

[12] D. C. U. P. S. D. Y. Singh, “Hadoop: Understanding the Big Data Processing Method,” *Int. J. Sci. Res.*, vol. 4, no. 3, pp. 1620–1624, 2015.
[13] A. Mirarab, S. L. Mirtaheri, and S. A. Asghari, “Value creation with big data analytics for enterprises : a survey,” *TELKOMNIKA (Telecommunication Comput. Electron. Control.)*, vol. 17, no. 6, pp. 5–7, 2019.
[14] Z. Li, H. Shen, W. Ligon, and J. Denton, “An Exploration of Designing a Hybrid Scale-Up/Out Hadoop Architecture Based on Performance Measurements,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 2, pp. 386–400, 2017.
[15] P. P. Deshpande, “Hadoop Distributed FileSystem: Metadata Management,” *Int. Res. J. Eng. Technol.*, vol. 10, pp. 2395–56, 2017.
[16] H. He, Z. Du, W. Zhang, and A. Chen, “Optimization strategy of Hadoop small file storage for big data in healthcare,” *J. Supercomput.*, vol. 72, no. 10, pp. 3696–3707, 2016.
[17] M. A. Mir, “An Optimal Solution for small file problem in Hadoop,” *International Journal of Advanced Research in Computer Science*, vol. 8, no. 5, p. 5697, 2017.
[18] M. A. Ahad and R. Biswas, “Soft Computing in Data Science,” vol. 545. Springer Singapore, 2015.
[19] K. Aishwarya, A. Arvind Ram, M. C. Sreevatson, C. Babu, and B. Prabavathy, “Efficient prefetching technique for storage of heterogeneous small files in hadoop distributed file system federation,” *2013 5th Int. Conf. Adv. Comput. ICoAC 2013*, pp. 523–530, 2014.
[20] A. Erraissi, A. Belangour, and A. Tragha, “Digging into Hadoop-based Big Data Architectures,” *Int. J. Comput. Sci. Issues*, vol. 14, no. 6, p. 7, 2017.
[21] M. A. Ahad and R. Biswas, “Dynamic Merging based Small File Storage (DM-SFS) Architecture for Efficiently Storing Small Size Files in Hadoop,” *Procedia Comput. Sci.*, vol. 132, pp. 1626–1635, 2018.
[22] Y. Guan, Z. Ma, and L. Li, “HDFS Optimization Strategy Based On Hierarchical Storage of Hot and Cold Data,” *Procedia CIRP*, vol. 83, pp. 415–418, 2019.
[23] B. Gupta, R. Nath, G. Gopal, and K. K, “An Efficient Approach for Storing and Accessing Small Files with Big Data Technology,” *Int. J. Comput. Appl.*, vol. 146, no. 1, pp. 36–39, 2016.
[24] S. Bende and R. Shedge, “Dealing with Small Files Problem in Hadoop Distributed File System,” *Procedia Comput. Sci.*, vol. 79, pp. 1001–1012, 2016.
[25] S. Rangrao Kadam and V. Patil, “Review on Big Data Security in Hadoop,” *Int. Res. J. Eng. Technol.*, vol. 4, no. 1, pp. 1362–1365, 2017.
[26] C. Choi, C. Choi, J. Choi, and P. Kim, “Improved performance optimization for massive small files in cloud computing environment,” *Ann. Oper. Res.*, vol. 265, no. 2, pp. 305–317, 2018.
[27] R. Chethan, J. K. S. G. H. J, and P. M. C. N, “A Selective Approach for Storing Small Files in Respective Blocks of Hadoop,” *International Journal of Advanced Networking & Applications (IJANA)*, pp. 461–465.
[28] B. Sahare, A. Naik, and K. Patel, “Study of HADOOP,” Int. J. Comput. Sci. Trends Technol., vol. 2, no. 6, pp. 40–43, 2014.
[29] S. Dzinamarira, F. Dinu, and T. S. E. Ng, “Ignem: Upward migration of cold data in big data file systems,” *Proc. - Int. Conf. Distrib. Comput. Syst.*, vol. 2018-July, pp. 65–75, 2018.

BIOGRAPHIES OF AUTHORS

Mohammad Bahjat Masadeh, PhD candidate Data Minig/Big Data UTeM | Universiti Teknikal Malaysia Melaka, Master in information technology UUM | Universiti Utara Malaysia in 2008, 2009. He is a Sr .Net developer in info tech deanship Umm alQurah university, Mecca Sr Sharepoint developer in info tech deanship Umm alQurah university, Mecca, Wordpress blogger about hadoop administration, big data geek, Co-founder of wrshaa.com.

Mohd Sanusi Azmi received BSc., Msc and Ph.D from Universiti Kebangsaan Malaysia (UKM) in 2000, 2003 and 2013. He joined Department of Software Enginering, Universiti Teknikal Malaysia Melaka (UTeM) in 2003. Now, he is currently an Associate Professor at UTeM. He is the Malaysian pioneer researcher in identification and verification of digital images of Al-Quran Mushaf. He is also involved in Digital Jawi Paleography. He actively contributes in the feature extraction domain. He has proposed a novel technique based on geometry feature used in Digit and Arabic based handwritten documents

Sharifah Sakinah Syed Ahmad received B. Applied Science (Hons) Computer Modelling and M. Sc Mathematics from Universiti Sains Malaysia (USM). Her PhD is in Software Engineering and Intelligent System from University of Alberta, Canada.She is expert in Computer Aided Geometric Design (CAGD) and neural networks.