International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2015): 78.96 | Impact Factor (2015): 6.391

An Optimized Approach for Processing Small Files in HDFS

Deepika

Department of CSE, MSRIT, Bangalore-54, India

Abstract: ***In Today's world cloud storage, has become an important part of the cloud computing system. Hadoop is an open-source software for computing huge number of data sets to facilitate storage, analyze,manage and access functionality in distributed systems across huge number of systems. Many of the user created data are of small files. HDFS is a distributed file system that manages the file processing across huge number of machines in distributed systems with minimum hardware requirement for computation. The performance of the HDFS degrades when it is handling the storage and access functionality of huge number of small files. This paper introduces the optimized strategies to handle small file processing in terms of storage and access efficiencies. Replication algorithms: HAR and sequenceFile, merging algorithms, replica placement algorithms, Structurally-Related Small Files (SSF)- File Merging and Prefetching Scheme (FMP) and SSF-FMP with three level prefetching-catching technology. The proposed strategies help in effective increase of access and storage efficiency of small files. Inclemently shorten the time spent for reading and writing of small files when requested by clients.***

Keywords: Cloud storage, HDFS, Merging, Replica placement, sequence File.

1. Introduction

Now days' cloud computing has become most important computation mechanism in the web pattern computation and more and more around the world. Cloud storage is the main part of cloud computing as it provides the data storage access of large data sets for end users whenever and wherever required in the distributed file system[1]. Data replication acts vital role in the cloud storage in terms of data availability, minimizing data access latency and load balance on several servers at the same time. Hence performance of the system increased exponentially [3].

Hadoop

Hadoop is an open-source software for computing huge data sets to facilitate storage, manage, analyze and access functionality in distributed systems across huge number of systems. The Hadoop mechanism started in Google, Facebook, Twitter etc. to store and process huge amount of data. Hadoop mainly contains the two parts they are: Map Reduce and Hadoop Distributed File System(HDFS) [2].

HDFS is a distributed file system that manages the file processing across huge number of machines in distributed systems with minimum hardware requirement for computation. HDFS mainly supports write once-read many type of workload on streaming data access and huge data sets. It provides the data block replication of data to save against hardware failures. HDFS is mainly a master/slave architecture and includes the mainly three components those are: Name Node, Data Node and Clients as show in bellow figure.

HDFS contains single Name Node that mainly responsible for managing namespace for file systems with respect block replacement mechanism and data Node. It stores the metadata or file namespace in DRAM for fast access and also keeps the copy of the file system namespace (FSIMange) on disk for data recovery. Any modification or updation to the file system namespace are stored in the EDitLog and timely merge with the FsImage so that stored copy of the file system namespace always be up to date. Name Node contains the files metadata, filedirectories, file content blocks which includes updation time,file length, block size, proprietorship, replication, access information. Data Nodes stores the blocks of data from the splitting mechanism of original file into smaller blocks of data with each blocking having the size of 64MB.Data Nodes are stores the file in terms of blocks of data and responsible for serving read write to/from clients. Data Nodes does the operations on blocks based on the instruction given by the Name Node. HDFS mainly offers replication for fault tolerant and saves against node failure. HDFS mainly defers from other file system in the following cases:

1) Fault-tolerant
2) Low cost hardware
3) Increased Throughput

MapReduce

MapReduce is another mechanism that supports the parallel generating and processing huge data sets. It mainly takes care of paralleling, scheduling of tasks and handles the failures automatically. Map function inMapReduce frame processes the key-value pair to generate intermediate key-value pairs.Reduce function in MapReduce frame integrates the intermediate values that associated with the same intermediate key [4].

2. Small File Problem in HDFS

HDFS is a distributed file system that manages the file processing across huge number of machines in distributed systems with minimum hardware requirement for computation. The capacity of the HDFS incrementally reduces, when it is handling the storage and access functionality of huge number of small files. Many current systems in energy, climatology, astronomy, biology, e-Business, e-Library, and e-Learning contain huge amounts of small files. Hence how to manage storage and access of small files and the replication becomes a critical issue in

HDFS. Storing and managing of huge number of small files mainly creates heavy burden on the Name Node itself. It Increases the memory consumption of Name Node and degrades the access efficiency of small files. The huge number of small files creates more impact on the performance of the metadata management in HDFS.HDFS appears as a bottleneck for handling metadata services for huge small files.

3. Optimized Solutions to Small Files

Data replication

Data replication is widely used in the cloud computing system to give the promise for data availability and increased system reliability. Computation node access the data for computation also be stored in the storage that may have some geological distance from the computational node. Hence, we can use the data replica strategy that stores the replica of data to other storage node that close to the computation node so that time to access the data from storage node can be reduced incrementally method of data replication helps in large distributed system can minimize the energy cost and expenses of the operating systems also handles the data lost during some situation in the distributed systems

.

General Solutions

1. HAR (Hadoop archives):

HAR combines the HDFS files into blocks of data. AR maintained both data files and metadata files. It may increase the performance on memory usage of Name Node. Creation of copy of the file when creation of archives leads to heavy load on the disk space [3].

2. Sequence File

Sequence File has the concept of Persistent data structure is used to store the binary key-value pairs of data files. The key value contains the filename and value become the content of the files. The bottle neck with Sequence File access is that, it needs to read whole Sequence File to lookup into particular key and the mentioned key can be deleted or updated. Hence access efficiency of sequence File is mostly affected [3].

The Merging Algorithm of Small Files

Merging algorithms has the concept forstoring of huge number of small files into large files. It reduces the length of the metadata ofhuge number of small files storage. Hence data access efficiency and latency can be improved and access time is saved accordingly. Here files are classified into four types they are: Read intensive, write intensive, read and write intensive and read and write sparse. The idea behind here is classifying all small files into four types and merge the files that belongs to specific categories and allocate files into blocks of data. Then merging of third and second category to fourth category and then merge files of first category to these blocks if some blocks are not otherwise merge all files belongs to first category. Basic idea is small files are merged into large files and allocated storage interims of blocks [3].

Replica Placement Algorithm

Based on the merging algorithms, replica placement algorithm will act accordingly. Some set of small files will be selected based on the size of the small files and placement logic will be activated at some interval of time t. Level mechanism maintained here, firstly merged files are stored in level 2, if user want to access these files these files will be moving upper levels i.e. level1 and level0.The data at level2 is from Data Nodes. The main objective of the replica placement algorithm is todetermine whether a block that stores small files should be placed in an Innode[3].

Structurally-Related Small Files (SSF)- File Merging and Prefetching Scheme (FMP)

Structurally-related files are the files that are dependent segments of a large file. These files have the characteristics that, they can be merged into fixed set, to be represented as large file. Set contains fixed file numbers and other files are not included in this fixed set. By considering the approach of structurally related files, FMP-SSF is introduced. FMP-SSF includes the following steps:

- The large files contain the Structurally-related small files are merged into single file and italso calledmerged file to reduce burden on the Name Node in terms of memory consumption.
- Prefetching and caching methods are user to improve the access efficiency of small files.

File merging will take place in HDFS clients, that merges large files contain the structurally-related small files into single merged file. Name Node manages the metadata of merged file and existence of original files will not be perceived in Name Node. So, huge small files merging methods reduces the complexity of metadata management by Name.Thus, memory consumption on Name Node reduces increasingly. Local index file contains the indexes for original files which defines length and offset of merged file. Hence for every merged file, index file is built to denote offset and length of native files in merged file. If, merged file contains multiple blocks, local index files for each block are pointed by the starting address of each block. Storage efficiency improved incrementally using SSF-FMP method by reducing memory consumption on Name Node with merging of huge number of small files into single merged file and improved access efficiency by using catching technique to access small files by reducing access time [4].

Structurally-Related Small Files (SSF)- File Merging and Prefetching Scheme (FMP) with three level prefetching-catching technology

SSF-FMP includes the following three level prefetching and catching technology

a) Metadata catching

b) Prefetching index

c) Prefetching correlated files

a) Metadata catching

In metadata catching, file mustbe bind to the merged file to get the merged file metadata from Name Node for reading and writing of small files when requested by clients. The cached metadata of the merged file helps in minimizing the connection to the Name Node to fetch the metadata and clients can directly access the cached metadata when the native files of the merged file requested by clients. Hence, we can save the accessing time for merged file metadata

from the Name Node when end user requests the same file repeatedly [5].

b) Prefetching index

Local index files contain the offset and length of each files belongs to the merged files and stored in different blocks. This helps the client to identify blocks that contains the requested file. Prefetching local index file from the Data Node helps to perform direct I/O operations when reading files that belongs to the same merged file. Hence reducing the I/O operation time [5].

c) Prefetching correlated file

Files of the merged file contains the files in logical order and have the right correlations. The prefetching correlated files triggered when the clients gets the requested file. Depending on the logical order of the files in merged file, prefetching of the correlated files in the same merged files takes place [5].

Optimized Scheme – SIFM

Optimized schema called Structured index file merging used to improve access and storage efficiency of small files in Hadoop distributed file system. The idea behind SIFM includes the following:

1) File correlations are included in merging files that reduces the access time and delay while reading small files.
2) Storing metadata of files in a structed distributed architecture is used to reduce the access time operations of requested files.
3) Access locality is applied in the inter block on Data Nodes. Catching and prefetching used to reduce the access time when simultaneous reading of huge number of small files.

As per the file merging algorithm, small files are merged and converted as a merged file. In optimized scheme, small files are merged to the large file. At the same time the metadata file creation and Structured index files are built for files of merged file thatloaded into the Name Node and merged files loaded into the Data Node. In other side catching and prefetching schema helps in metadata catching and file indexingto fetch small files. The requested files are searched in HDFS blocks based on the offset and length of small files when requested by clients.Optimized technique for fetching small files greatly helps in improving I/O performance and reduces the communication cost when reading huge number of small files [6].

4. Observations and Suggestions on Below Algorithms

By analyzing the table created above in Table 1, weakness in some of these strategies are pointed out along with some suggestions for improvement:

1) Merging algorithms and block replacement algorithm performs better than HAR and SequenceFile in terms of performance, access and storage efficiency [3].
2) Merging of structurally related small files enhances the access and storage efficiency by using catching technology to shorten the access the time than just merging of files [4].
3) Structured index file merging and prefetching using catching performs better than the merging algorithms and incrementally reduces the burden on Name Node in HDFS [6].
4) Merging model to merge massive small files with efficient index structure optimizes the performance of processing small files and efficiently reduces the CPU time taken to process small files which reduces the memory utilization of the Name Node [9].

Comparison of existing Strategies for small file processing

Sl. No.	*Title of Paper*	*Techniques Used*	*What isOptimized*	*Metrics for Evaluation*
1	Research Article - Optimized Data Replication for Small Files in Cloud Storage Systems [3]	• Merging algorithm • Block replica placement algorithm	• Effectively shorten the time spent reading and writing small files • Improve the access efficiencies of small files • Performs better than HAR and SequenceFile	• Time spent reading and writing small files • Access time of small files
2	An Efficient Approach for Storing and Accessing Small Files in HDFS [4]	• structurally-related small files merging and prefetching technique • Caching technique • Boundary Filling algorithm	• Good storage efficiency • Access efficiency of small files • Better performance by caching	• Storage efficiency • access time • Response Time • Performance
3	An optimized approach for storing and accessing small files on cloud storage [5]	• File grouping • Prefetching technique	• Storage efficiency of small files • Access efficiency of small files	• Response time • Storage efficiency • Access efficiency
4	Optimization Scheme for Small Files Storage Based on Hadoop Distributed File System [6]	• Structured Index File Merging-SIFM • Prefetching and caching strategy	• Better performance in storing and accessing of huge small files on HDFS	• Performance • Access time
5	An effective strategy for improving small file problem in distributed file system [7]	• Small files merge strategy • Prefetching files based the transition probability	• MDS workload is ffectively reduced • Request responsedelay.	• MDS workload • Access delay

6	Optimization strategy of Hadoop small file storage for big data in healthcare [8]	• File merging algorithm based on balance of data block	• Optimizethe volume distribution of the big file after merging • Effective reduce in number of data blocks • Reduce the memory overhead of major nodes of cluster • Reduce load to achieve high-efficiency operation of data processing	• Distribution of big file • Number of data blocks • Memory overhead on Name node
7	Improving the Performance of Processing for Small Files in Hadoop: A Case Study of Weather Data Analytics [9]	• Merge Model to merge massive small files • Efficient indexing mechanism	• Optimize performance of processing small files drastically up to 90.83% • Effectively reduces the memory utilization of the namenode • Less CPU time taken • Reduces the time taken • Efficiency of storing, managingand processing small file	• Performance • Data processing time • Memory utilization • CPU time • Storage efficiency

5. Conclusion

Conclusion and Future work:

Most of the scientific and other applications are cloud based. Cloud storage is the main part of cloud computing as it provides the data storage access of large data sets for end users whenever and wherever required in the distributed file system.HDFS is a distributed file system that manages the file processing across huge number of machines in distributed systems with minimum hardware requirement for computation. This paper introduces methods to handle the small files problem inters of accessing and storage of huge number of small files. Merging algorithm helps to mergesmall files to form a large or logical merged file and replica placement algorithm reduce theseeking time when reading or writing huge number of small files.

The merging and Replica placement proposed in existing system typically optimizes the storage efficiency and minimizes the seeking time when file requested by clients. In Future work using catching technique in accessing small files helps in improving access efficiency more than the existing strategies.

References

[1] T. Sivashakthi and N. Prabakaran, “A survey on storage techniquesin cloud computing,” International Journal of EmergingTechnology and Advanced Engineering, vol. 3, no. 12, pp. 125–128,2013.

[2] X. Liu, J. Han, Y. Zhong, C. Han, and X. He, “ImplementingWebGIS on hadoop: a case study of improving small file I/Operformance onHDFS,” in Proceedings of the IEEE InternationalConference on Cluster Computing and Workshops.

[3] Optimized Data Replication for Small Files in Cloud Storage Systems Volume 2016, Research Article. Article ID 4837894,2016

[4] An Efficient Approach for Storing and Accessing Small Files in HDFS Volume No.: II, Special Issue on IEEE Sponsored International Conference on Intelligent Systems and Control (ISCO’15).

[5] An optimized approach for storing and accessing small files on cloud storage, Journal of Network and Computer Applications,2012.

[6] Optimization Scheme for Small Files Storage Based on Hadoop Distributed File System.Yingchi Mao1, 2, Bicong Jia1, Wei Min1 and Jiulong Wang1 Vol.8, No.5 (2015), pp.241-254.International Journal of Database Theory and Application.

[7] An effective strategy for improving small file problem in distributed file systemTao Wang, Shihong Yao, Zhengquan Xu*, LianXiong, Xin Gu, XipingYang International Conference on Information Science and Control Engineering,2015.

[8] Optimization strategy of Hadoop small file storagefor big data in healthcareHui He1 · Zhonghui Du1 Weizhe Zhang1Allen Chen,2015.

[9] International Journal of Computer Science and Information Technologies, Vol. 5 (5), 2014, 6436-6439.Improving the Performance of Processing for Small Files in Hadoop: A Case Study of Weather Data Analytics.