The Design of the Distributed File System in a Small File

Yamei Zhang, Yanjun Fu

School of Intelligence Science and Information Engineering
Xi’an Peihua Universty, Xi'an, China
312413420@qq.com，931108963@qq.com

Keywords: The Internet of Things; MSF; HDFS; Efficient storage; Sensor cluster

Abstract. Through a new storage architecture of Internet of Things (IOT) big data research, retrieval and analysis of large data put forward the optimization design of innovation, design and build the mass small file storage system based on HDFS SensorFS, solve the fundamental problem in storage, efficient for large data storage and management of application support, provide new way for systematic approach.

1 Introduction

With the continuous development of sensor and network technology, the IOT has become an important direction for research and industry. As shown in fig 1, in the IOT a large number of sensors detect physical objects and send sample data files based on physical objects to the data center. It is generally believed that the number of sensors in the Internet of things has a huge number of characteristics[1].

The Hasoop file System, HDFS, is an open source implementation of Google Hie System, and becomes the standard model for the industry to study cloud computing and cloud storage, cloud application and cloud service reference. However, the existing HDFS architecture for small files stored there write low throughput, the master node memory overhead big main problems, as shown in Fig. 1 ,the under different file size 7 Data Node cases HDFS write throughput system limit value, when the file size is 64 MB, HDFS write throughput rate close to the limit of disk throughput, but when the file size is less than 1 MB, with the decrease of the file size, write throughput rate significantly decreased, when the file size is 10 KB, write throughput will only be about 15 MB/S, obviously cannot meet the Internet of things small and medium-sized file write throughput requirements[2][3].

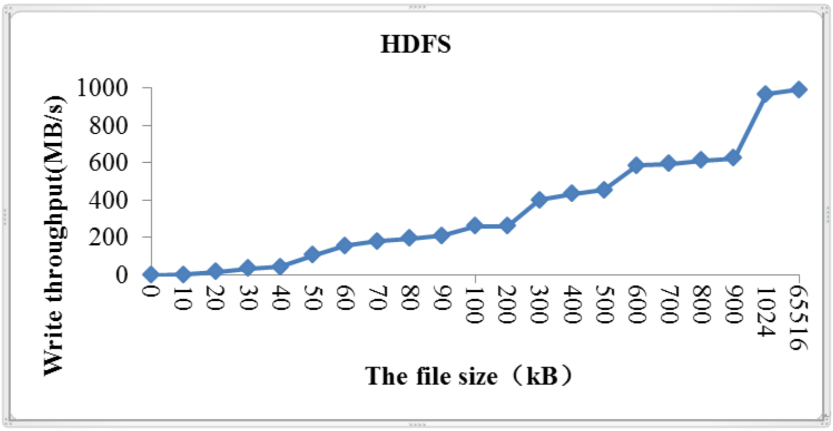

Figure 1. HDFS writes granularity in different files

2 Overview of Relevant Research at Home and Abroad

Existing technology and the Internet of things big data distributed storage and the demand of real-time computing, there are huge gap between relational database performance shortfalls in achieving high concurrency, speaking, reading and writing, and relational database is a large amount of data, such as hundreds of millions of the data, the query performance is not so efficient[4]; Second, the horizontal scalability is poor and cannot be deployed to a collection of more than 1000 nodes. In order to solve the shortage of the relational database, different company launched its own database design, such as Google designed the Big Table, it is a sparse, distributed and persistent multidimensional sequence Map. Cascade update the basic idea and the basic idea of parallel update at space overhead optimization index has some research, mainly includes two kinds of strategies: reducing internal node size, reduce the internal nodes routing overhead space[5]. Section is another way of thinking for internal space to each according to his need, general distributed file system including: Google GFS, Hadoop HDFS and so on, the system adopts the Master - Slave architecture, the Master node is responsible for maintaining the file system namespace, any changes to the file system namespace or attributes will be recording the Name ode.

3 Distributed File System for Massive Small Files (MSF)

In order to solve the above problems of HDFS, the following optimization design is made: (1) Design and build the mass small file storage system based on HDFS SensorFS, through design and build distributed memory on HDFS file system "cache" and "merge", optimize the small file write throughput. (2) Design a distributed cache system, used to write on the massive small file cache operation, after file cached in memory, file will be according to the different query, accordingly file merging, merged into a logical file, written to the underlying HDFS[6]. Compared to HDFS, the parallel writing caching mechanism can effectively improve the writing of large volumes of small files. Design parallel file merge operation, file organization through parallel log structure, and parallel write merge operation, can effectively improve the file writing. In addition, the distributed write cache structure allows all nodes to participate in the file cache and merge process, speeding up the merge of files. (3) The sensor clustering algorithm is proposed and designed. Object-oriented design points of the query sensor relevance evaluation model, and design the fast clustering algorithm of sensor, compared with the traditional hierarchical clustering, greatly optimize the document clustering need time overhead and space overhead, gathered by sensors file at the same time, improve the performance of the object query.

4 Client Aggregation Strategy for MSF

Massive small file client oriented aggregation strategy basic idea is: the client for the pretreatment of the small files according to the design of the aggregation strategy after the client files gathered, The research content mainly focuses on different application scenarios to design different document aggregation strategies. For GIS files, small files are merged into large file storage according to the geographical location of the files, and the grid index is set up for the access of small document. Wait for the person based on the large number of PPT files, mp3 files, documents and other related strong characteristics of small files for metadata analysis, and the file clustering; For XML files, analyze the correlation between different XML documents by analyzing the contents of XML files, and merge them. The user specified in the small files stored on a fast, and in the memory of the Data Node to store these small file metadata information, to the HDFS small file I/O performance optimization.

5 Centralized DRAM Caching Strategy for MSF

In order to better to send from different HDFS client files provide rapid aggregation service, some of the work are also put forward the open up additional memory buffer to write data center's file buffer and merge, its basic idea is: use of a centralized memory buffer layer, namely a single server memory, write requests of massive small file cache and merging[7]. Yan and others devised HMFS, the use of a centralized memory buffer cache HDFS write please E, and set up an extra cache K used in file merging, responsible for the small file merging became large files in HDFS, to improve the HDFS write throughput[8].

6 SensorFS System Design

6.1SensorFS Design Scheme

This paper proposes a suitable for the IOT big data environment of mass small file efficient storage system SensorFS. In order to improve the writing performance of large amount of small files, two new designs are adopted:(1) Write cache: First put numerous small file write cached in memory, and then through clustering writing method improve the massive small file write throughput rate, and reduce the communication cost between nodes. (2)Cluster writing: In the future, small files from different sensors will be clustered, and the files after clustering will be merged into large files and then written to persistent storage. Fig.2 shows the SensorFS system architecture design. Above SensorFS in HDFS underlying storage increase a Distributed Memory File System (DMFS)[9]. DMFS mainly works on two tasks: writing cache and sensor clustering. The write cache provides high-performance write caching services for the underlying HDFS to optimize write throughput rates. The sensor clustering is designed to divide the sensor into multiple clusters, which can be merged into large files according to the clustering results, and finally the large files are written to the HDFS for persistent storage. By writing cache and clustering, you can avoid the unkind of HDFS written by MSF.

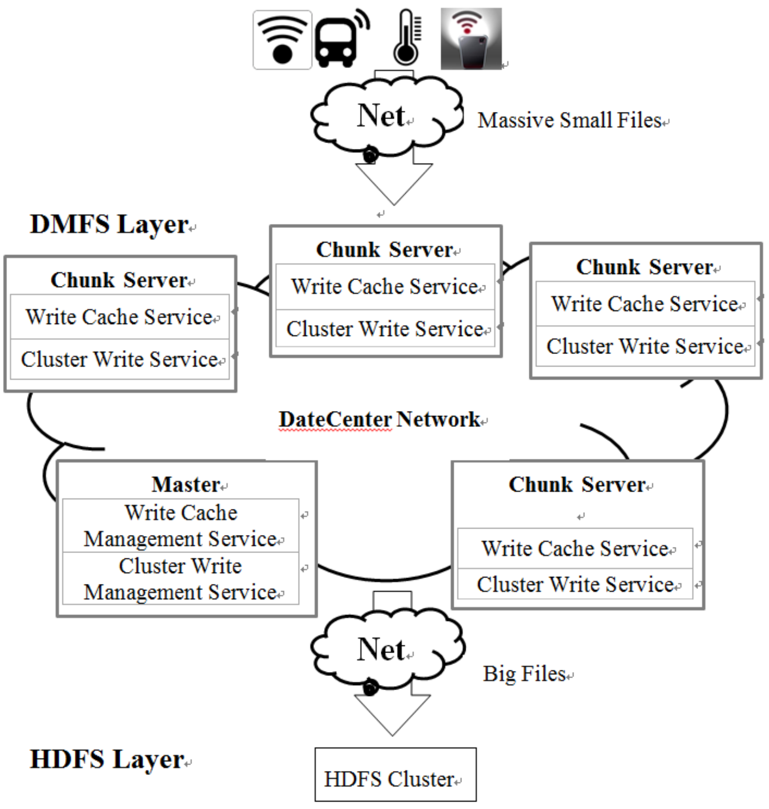

Figure 2. System architecture design for SensorFS

System deployment of DMFS to "top", which makes the DMFS design does not need to source the HDFS modification, and easier to manage, the risks from the HDFS version changes. DMFS can be installed on the master node, storage node, or any other server on HDFS, and connected to HDFS via the network. SemwFS USES DMFS for writing cache to get a high write cache throughput and file merge rate. In short, the distributed writing cache of SensorFS can significantly improve the writing throughput. You can effectively reduce the time overhead required for object queries in the Internet of things.

6.2 Sensor Clustering Based on Dependency

Sensor clustering is a kind of sensor hierarchical clustering algorithm based on dependence. Considering the iot with enormous sensor as the carrier, and the physical object is detected by sensors as the main body in the information, the Internet of things in the data access method is mainly based on the retrieval and access to a physical object, therefore, the corresponding physical objects sensor according to the existing storage related relationship, not only can improve the efficiency of the file is written to, but also to the query operation in the future. Different sensors will detect the same physical object file merging into one or more large files, can significantly reduce the files needed for data retrieval IO number, so as to improve the system in the face of the object based on query performance.

First, the connection between sensors is defined as the dependency between sensors. Then, the dependency relationship is transformed into a graph structure, and a dependency representation structure based on dependency graph is presented. Finally, the sensor is hierarchical clustering based on the dependency graph. Hierarchical clustering algorithm based on dependency graph in the search for the correlation of point pairs, only need to compute a dependency graph exists in the edge of the distance between the point of, at the same time cluster of condensation process, after the merger of the two most relevant cluster, only need to update the new combined with some points of the distance between the clusters, which can reduce the time overhead of clustering.

7 Memory Write Rate versus Experiment

At present commonly used open source memory storage system including Redist distributed memory database, a distributed cache system Memcached, Tachyon distributed memory file system, distributed cloud storage RAM Cloud Spark and distributed memory computing framework. We DMFS、Redis、Memcached, Tachyon, RAM Cloud and Spark the write throughput tests respectively, of which the writer DMFS writer threads in number is set to the CPU thread, Redis using YCSB set operation 10 tests, Memcached using memslap multithreaded set operation 10 tests, Redis, Memcached through consistent hashing scheduling in the client cluster deployment we tested under 5 storage nodes DMFS, Redis, Memcached, the limit of the Tachyon write throughput, Experimental results show that the DMFS when faced with a small file write throughput performance, on average, 60% higher than the Redis, mainly because of DMFS adopted multi-thread concurrent write technology to improve the system throughput, and Redis thread structure for the single thread, DMFS than Memcached performance improvement by 407%, mainly because Memcached in serious lock conflicts brought by the write operation, DMFS than Spark and Tachyon increased respectively625% and 182%. DMFS compared with RAMCIoud performance improved 31%, mainly because RAMCIoud use a dispatch thread to handle all of the network communication, then request is passed to the network by means of RPC requests the worker threads, has brought the loss on the write performance. To sum up, DMFS is suitable for writing cache of large amount of small files.

8 Conclusions

This paper presents a distributed write caching mechanism and document clustering based on sensor clustering, and designed a distributed cache system, used to write on the massive small file cache operation, effectively improve mass small file write throughput. Design parallel file merging, through parallel structure of log file organization, and parallel write merge operation, can effectively improve the file write throughput of the system in view of the high throughput mass small file write caching services and small file in memory consolidation services at a high speed.

References

[1] Guang-ming CHEN. Teaching Mode of Internet of Things in Colleges. Advanced Science and Industry Research Center. IEEE Internet Computing, vol. 5, issue 9:25-31, 2017.
[2] Zou Shuilong. Exploration on Personnel Training Mode of Internet of Things under the Universities Transformation and Development Background. Research Institute of Management Science and Industrial Engineering. Proceedings of 3rd International Conference on Computer Science and Electronic Technology International Society, 2017:100-113.
[3] A．Zaslavsky, C.Perera, D. Georgakopoulos, Sensing as a service and big data, arXiv preprint ArXiv: 2013,1301.:59-64.
[4] Duan shengze. Design and implementation of big data service platform based on Hadoop based cable production [D]. University of electronic science and technology, 2017:21-30.
[5] Li Chunhui. Application of big data in agricultural Internet of things [D]. Qiqihar university, 2015: 21-25.
[6] Lei Lu. A Design of Wireless Sensor Network for the Internet of Things[A]. Research Institute of Management Science and Industrial Engineering. Proceedings of 5th International Conference on Mechatronics, Materials, Chemistry and Computer Engineering (ICMMCCE 2017)[C]. Research Institute of Management Science and Industrial Engineering:93-98.
[7] Nongjian, Lu Insheng, Lu guangquan. A large data storage model based on the experimental platform of campus Internet of things [J]. Journal of wuzhou university, 2014:1-6.
[8] Liu Fuxin. Research and implementation of a common processing platform for IoT data based on cloud computing [D]. Central China normal university, 2015:11-18.
[9] Zhang Lanting. Social value and strategic choice of big data[D]. CPC central party school, 2014:19-23
[10] Xiaoyu Lu. Expressway ET Interconnection based on Internet of Things Communication Middleware. IEEE Internet Computing, vol. 3, issue 6:35-46, 2017.