Journal of Computer Science

Technical Reports

Effects of Design Factors of HDFS on I/O Performance

Han-Gyoo Kim

Department of Computer Engineering, Hongik University Seoul, Korea

Article history
Received: 11-01-2018
Revised: 28-02-2018
Accepted: 13-03-2018

Tel: 82-2-320-1469
Email: hgkim@hongik.edu

Abstract: Four major design factors of HDFS, the block size, the number of data nodes, the number of client processes and replication factor are investigated to find out the effects on the I/O performance of HDFS by performing experiments in a real physical HDFS infrastructure consisting of 64 Hadoop data nodes of Intel i9 based blades. The block size is observed to be optimal when it equals to about 1Gb or 128MB that is the amount of the data the hard disk drive device can effectively input and output for 1 second in most of today's off-the-shelf computers. Sophisticated allocation strategy is required to determine the number of mappers and reducers as the number of data nodes increase because the overall performance is influenced in complicated manner by the number of raw data blocks of the job to be processed, the processing time of the blocks for each node and the overhead of shuffling. Experiments shows that Hadoop distributes the work properly that the number of clients does not have a significant impact as the number of clients increases. There is little delay in copying the replica because replication is done in pipelined manner although the network is overloaded.

Keywords: Network Integrated Storage, Big Data, Cloud Storage, Scalable Storage, Huge Scale I/O

Introduction

Optimization of HDFS Data Processing

Hadoop as Operating System for Big Data Processing

The Hadoop Distributed File System (HDFS) and other software systems that make up the Hadoop Ecosystem are becoming increasingly valuable as an operating system for processing Big Data (OPDi, 2017) (Big Data, 2016) (Chaudhari *et al*., 2018). The input/output performance of the HDFS and the performance of the MapReduce layer implemented on top of the HDFS should be guaranteed to realize high performance application that reflects large data characteristics with a large size and a relatively short life cycle. A lot of researches have been reported to explore the I/O performance of HDFS (Park, 2016; Shankar and Lin, 2017; Dev and Patgiri, 2014; Clemente-Castillo *et al*., 2018), but few experimentation studies have been reported on the optimal design factors of HDFS system such as the block size and minimum number of data nodes required. In this study, we have studied the factors that determine the input/output performance of HDFS by performing the actual performance experiments. To find out the optimum design factors of the Hadoop cluster.

Distributed File System – HDFS

Although there is performance advantage through parallel processing of HDFS, a representative distributed file system supporting the Apache open source project Hadoop (Apache, 2016), efficient data distribution control is the most important design element of distributed file system. It must cope with hardware failure flexibly and ensure adequate performance through proper resource management.

Features of HDFS

HDFS divides the file into blocks and the default size of the HDFS block is 128MB. A node storing an HDFS block is called a data node and usually a Hadoop cluster is composed of several tens to thousands of data nodes. HDFS distributes the blocks constituting a file to multiple data nodes in order to process the file in parallel. The replicas of each block are copied to other data nodes to provide the reliability. This feature of HDFS is suitable for batch operations requiring high data throughput (HDFS, 2017).

MapReduce

HDFS is a solution that divides a file into blocks of the same size and stores them in data nodes distributed over the network. MapReduce is a framework to process in parallel

Science Publications

© 2018 Han-Gyoo Kim. This open access article is distributed under a Creative Commons Attribution (CC-BY) 3.0 license.

large amount of data stored over the data nodes. The Hadoop ecosystem 1.0 has failed to attract many analysts and users familiar with relational database analysis systems to Hadoop. To solve these problems, Hadoop ecosystem 2.0 adopted YARN (Yet Another Resource Negotiator) to improve scalability and data throughput (YARN, 2017).

I/O Operations in Hadoop

Writing and Reading Hadoop Blocks

Figure 1 (HDFS, 2017) shows the process of storing data blocks in HDFS. When a client of a Java Virtual Machine (JVM) requests data storage, the data stream class asks the name node for a list of data nodes to store the blocks. The data stream class receives the returned data node list from the name node and sends the blocks to the data nodes in the list. HDFS prevents data loss by duplicating the requested data block and storing it in other data nodes in preparation for hardware failure of the data node. When a replica is sent from the client to the first data node in streaming mode, the first data node receives the data stream and replicates the block and transfers it to the second data node. After the block is stored, each data node informs the client and the name node that it has successfully stored the data.

Hadoop I/O Operations in MapReduce

MapReduce, the data processing framework of Hadoop, is a process that causes a lot of data storage device I/O and the network I/O. The efficient block distribution and shuffling process of Hadoop system determines big data processing performance (Anjos *et al*., 2014). The shuffle/sort process sends and receives the mapping results to the reducers through the network connecting the data nodes that stores the intermediate data blocks of the <key, value> records. The shuffling/sorting process entails network I/O whose amount of time is comparable to data block I/O time involved.

Formula for Data Processing Time of a Hadoop Job

Hadoop processing involves hashing for mappers and shuffling and sorting for reducers as well as inputting and outputting of HDFS data blocks. The following formula elicits simple but solid model to estimate the job completion time for a Hadoop task:

$$\begin{aligned} &Elapsed\,Time = Map\,/\,HashTime + Reduce\,/\,Sort\,Time \\ &= (Block\,Input + Hashing + Output) \\ &+(Shuffle + Sorting + Block\,Output) \end{aligned}$$

Notice that individual factors in the formula are intertwined such that improving a factor in the formula often accompanies degenerating other factor. For example, reducing the time for inputting the blocks usually increases the network traffic for shuffling among the nodes thus leading to the increase of shuffle time.

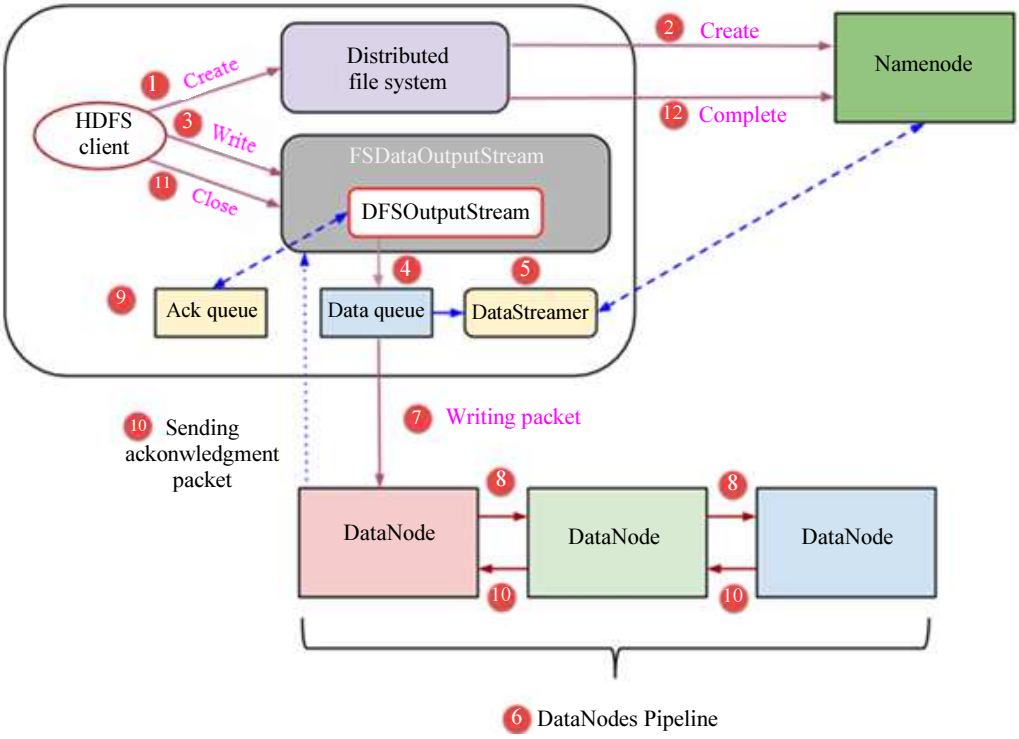

Fig. 1: HDFS block write

HDFS Design Factors that affects the I/O Performance of Hadoop Systems

Factors that influence HDSF I/O performance include the size of the block, the number of data nodes, the number of clients and how many blocks to replicate.

Block Size

Blocks are the units for reading and writing in HDFS. A file is divided into blocks, which are usually stored in blocks of 64 MB or 128 MB. The smaller the block size, the greater the block seek time. Conversely, if the size is too large, the degree of parallelism of the data is lowered and the efficiency of distributed processing is lowered. Therefore, the size of the HDFS block should be set to be optimal. Currently the default block size is set to 128MB.

Number of Data Nodes

If the number of nodes is increased, sufficient performance improvement can be expected due to the increase degree of distributed data processing in parallel. However, each data node continuously communicates with the name node, which causes overhead to increase the network load. HDFS manages the nodes on a rack-by-rack basis using the Rack Awareness function.

Number of Clients

Increase in the number of clients increases the load on the network, leading to a decrease in performance. However, if the node having the block is busy, the load is distributed to other nodes having the duplicate copies by YARN to balance the overall job processing.

Number of Replicas

Because replication is performed in pipelined manner, that is, the primary data node sends the replica to the secondary node at the same time it stores the block to itself, replication does not impose additional time to complete. Therefore, there is no delay in copying. However, the network load is increased. Hadoop's replication strategy is that if the client is a data node, it writes a block to itself and copies the replica to the nodes in other rack. If the client is not a data node, it is stored in one of the data nodes in the client's rack and the replica is stored in the other rack.

Hadoop I/O Performance Experiments

Experiments in this study were performed on a HDFS cluster with 64 Hadoop data nodes connected by 1Gbps Ethernet switch. Each data node is equipped with Intel i9 10 core 3.3GHz process, 8GB DRAM and SATA II HDD's.

Block Size and Performance

MapReduce jobs can impact overall performance depending on the number of mapper and reducer processes. Since the data is divided into a larger number of blocks of same size, more mapper processes make it possible to work faster. Therefore, the performance of MapReduce can be improved by increasing the number of mapper processes by reducing the block size. However, this is different from writing performance measurements because it is viewed from the perspective of MapReduce. From a writing point of view, SATA II disks used in Hadoop cluster nodes in the current experiment generally have a sustained data rate of about 130MB/s. It is concluded that reading and writing 100MB at a time is the most efficient if the average sustained data rate of disk is about 100MB/s. It also confirms the result is consistent to the conclusion from other papers (Tang, 2016; Shi *et al*., 2015; Mirza and Nagori, 2017; Ye *et al*., 2016; Ouyang *et al*., 2017). Experimental results of this paper showed the performance was best when the block size is of 64 ~ 128MB as given in Fig. 2.

Number of Data Nodes and Performance

Figure 3 shows the variation of I/O performance as the number of data nodes increases. The nodes are connected by 1Gbps Ethernet. In the experiment, all data nodes are installed in a single Ethernet switch to minimize the performance degradation due to the bandwidth limitation of the network.

The graph in Fig. 3 shows a relatively constant input/output speed of about 70 to 80MB/s regardless of the number of data nodes. Once the mapping process is completed, the resulting blocks are stored on the hard disk drives and the stored intermediate blocks are shuffled. In this case, blocks are transferred and received between the data nodes shuffling through the network. In other words, in addition to simply inputting and outputting data blocks, a considerable amount of network input/output occurs between data nodes. Therefore, depending on the bandwidth of the network connecting the data nodes, the overall input/output performance is greatly affected.

Notice in Fig. 3 that the per node average block write speed is relatively constant until each data node is overloaded by the number of blocks to write. It can be seen from the result that the processing speed is not lowered when each node writes up to a certain number of blocks, but when the number of blocks is increased, the processing speed is lowered. That is, there exists a minimum number of nodes that can process the total number of blocks without slowing down.

We also performed the same sets of experiments using low bandwidth network switch in order to investigate the effect of the network bandwidth. With the network switch of which bandwidth is 100Mbps that is one tenth of the 1Gbps switch used in the experiments shown in Fig. 3, we have the result that the write performance decreases as the number of nodes increases by more than 20% due to the low bandwidth of the network switch connecting the data nodes. We speculate that the overhead of communication between the name node and the data nodes due to exchanges of heartbeat and various metadata increases as the number of data nodes increases.

File Size : 1GB						
Block Size	Test1	Test2	Test3	Test4	Test5	Average
4MB	19	27	20	19	19	20.8
8MB	15	17	17	17	17	16.6
16MB	13	16	13	12	14	13.4
32MB	11	11	13	14	14	12.4
64MB	11	12	14	14	12	12.6
128MB	13	12	13	15	15	13.6
256MB	15	15	15	15	13	14.6
512MB	17	17	18	15	15	16.4
1024MB	18	18	18	18	18	18.0

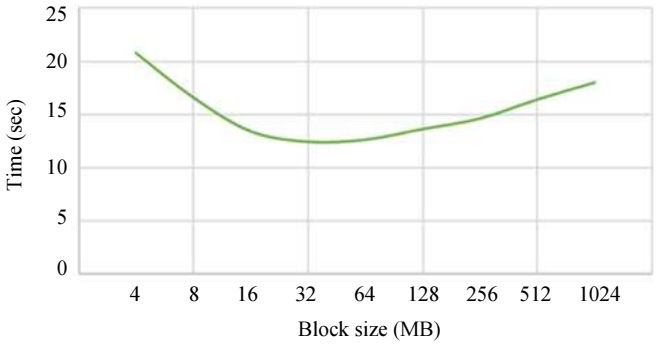

Fig. 2: Write performance by block size

Replication Factor : 3							
	Number of Blocks						
No. of Nodes	9	18	27	36	45	54	63
9	69.4	70.1	68.9	74.7	76.5	73.2	66.6
18	67.2	64.7	77.5	78.3	72.5	77.9	78.8
27	67.1	67.8	75.1	83.1	79.5	83.5	79.4

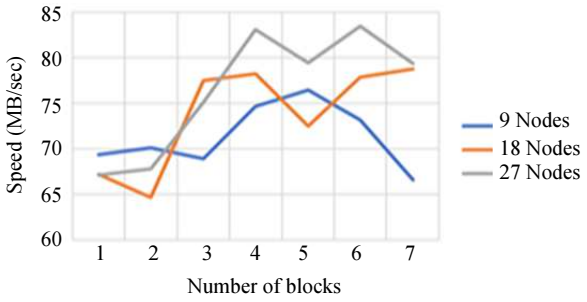

Fig. 3: Write performance by number of data nodes

Block Size : 128MB File Size : 1GB		
No. Clients	6 Nodes	12 Nodes
1	88.3	85.3
2	47.6	68.3
3	35.3	45.9
4	30.8	40.9
5	25.5	37.4
6	22.9	33.2
7	18.1	28.7
8	16.3	28.1
9	13.7	28.2
10	12.3	24.8
11	12.9	23.5
12	10.9	20.9
13	9.3	18.9
14	9.8	19.3
15	8.7	17.4

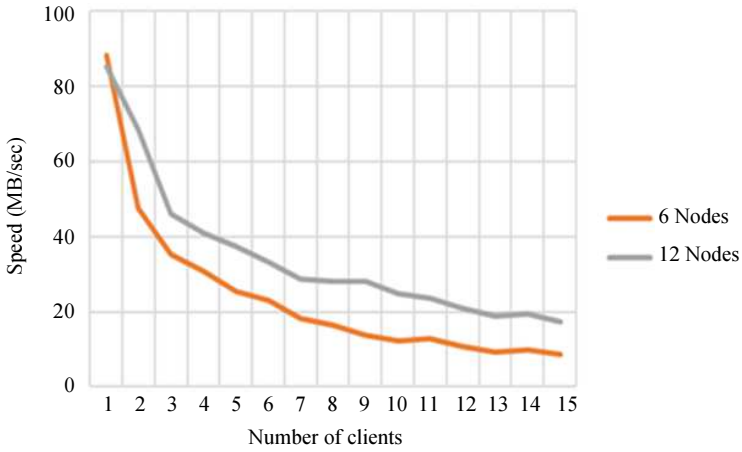

Fig. 4: Write performance by number of clients

Number of Clients and Performance

Simultaneous writing experiments were conducted to investigate the HDFS load that occurs when multiple clients are writing simultaneously. The experiment was performed by incrementing the client to 6 data nodes with replication of 3 and the block size of 128MB.

Figure 4 shows that as the client grows, the transmission time increases. The interesting fact is that the transfer time does not increase in proportion to the number of clients. There is a variation depending on the network environment, but it seems to maintain a constant speed even when a certain amount of load is applied. It is considered that this is due to the write policy of the block. Since two clients replicate three blocks to six data nodes, a total of six blocks are created when 128MB is transmitted. Hadoop avoids the concentration of blocks on specific data nodes through a balancer, but does not divide into exact number of nodes. This is because the Hadoop balancer avoids busy data nodes by judging the workload of the data nodes, but it does not reflect the network flow in its decision. If we increase the number of nodes enough, it is expected that we can reduce the load to some extent.

Figure 4 also shows the result of experiment in the same environment by increasing the number of nodes to 12. The total transmission time is also increased, but the transmission time is increased at a lower rate than when the number of nodes is six. The performance gain is up to about 150% instead of 200% even though we doubled the number of nods from 6 to 12.

Discussion and Conclusion

Our experimentation study contributes to find a set of important design parameters, block size, network bandwidth and number of clients, for Hadoop clusters of which number of nodes ranges up to several thousands.

Block Size

Experimental results show that the block size from 64MB to 128MB has the most efficient I/O performance. This seems to be related to the sustained data rate of 1Gbps of the SATA II hard disk drive used in the experiment. In other words, the size of the block showing optimal performance seems to be closely related to the effective bandwidth that the device storing the block can input/output. If high-speed SATA drives or SSD storage devices are used, it is estimated that higher input/output performance will be obtained when the block is larger than 128 MB.

Network Bandwidth Bottleneck

The result of this experiment shows that the bandwidth of the network connecting the data nodes determines the performance of the entire Hadoop cluster. In other words, the network bandwidth seems to have more influence on the overall average performance than the size of the high-speed storage device or the mounted DRAM. This means that there is a limit to increase the number of data nodes that can be mounted in a unit rack considering the performance price ratio. The observation seems reasonable because the shuffling process of MapReduce Hadoop framework induces a lot of network input and output, the bandwidth of the network device connecting the data nodes influences greatly the performance of the whole Hadoop application.

Number of Client Processes

As the number of client processes increases, the time required to perform the whole tasks also increases, but the time does not increase proportionally. This means that the resource management and job assignment functions of YARN are operating properly. As a matter of course, we can observe that increasing the number of nodes performing the operation proportionally alleviates the increase in the task completion time.

Findings and Contribution of Our Study

Firstly, we find that Hadoop block size should increase in proportion to the effective data transfer bandwidth of the storage device that contain the block for high I/O performance. Current default block size of Hadoop which is set to 128MB is an optimal size for the storage devices that have 1Gbps od sustained data transfer rate which is typical for SATA II type of HDD connected to PCI bus. This means that if a new storage device is to have 4Gbps of sustained data rate, the block size would have to increase to 125MB times 4, i.e., 500MB to have optimal I/O performance.

Secondly, we find that the most important governing factor for the overall Hadoop cluster performance is the network bandwidth of the cluster. That is, we must invest to provide faster network to the cluster than to faster storage devices.

Further Works

HDFS has a set of design parameters that affect performance and it is far from trivial to determine the optimal values of such factors due to the nature of the complex distributed system configurations as well as the characteristics of individual applications of Big Data processing. Especially, since the influence of the network bandwidth and the effective data rate of the storage device is greater, more extensive experimentation study should be performed to find out the effects of network bandwidth and the sustained speed of storage device on the overall Hadoop systems. We are currently preparing larger scale Hadoop cluster with 10G and 40Gbps Ethernet topology and with faster storage devices including SATA III SSD (Mao *et al*., 2018) and NVMe (Bhimani *et al*., 2018) storage devices and will perform experiments how the design factors including block size of HDFS and the number of data nodes are affected under various cluster environments.

Acknowledgement

The research is financed by 2015 Hongik University Research Fund.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that he has read and approved the manuscript and there are no ethical issues involved.

References

Anjos, J., B.R. Filho, J.F. Barros and U. Matte, 2014. Genetic mapping of diseases through big data techniques. Proceedings of the International Conference on Enterprise Information Systems, (EIS' 14) Barcelona, Spain. DOI: 10.5220/0005365402790286

Apache, H., 2016. Apache hadoop. https://en.wikipedia.org/wiki/Apache_Hadoop

Bhimani, J., Z. Yang, N. Mi, J. Yang and Q. Xu *et al*., 2018. Docker container scheduler for I/O intensive applications running on NVMe SSDs. IEEE Trans. Multi-Scale Comput. Syst. DOI: 10.1109/TMSCS.2018.2801281

Big Data Forum, 2016. http://www.big-dataforum.com

Chaudhari, R., G.S. Aujla, N. Kumar and J.J.P.C. Rodrigues, 2018. Optimized big data management across multi-cloud data centers: Software-defined-network-based analysis. IEEE Commun. Magazine, 56: 118-126. DOI: 10.1109/MCOM.2018.1700211

Clemente-Castillo, J.F., B. Nicolae, R. Mayo and J.C. Fernandez, 2018. Performance model of mapreduce iterative applications for hybrid cloud bursting. IEEE Trans. Parallel Distributed Syst. DOI: 10.1109/TPDS.2018.2802932

Dev, D. and R. Patgiri, 2014. Performance evaluation of HDFS in big data management. Proceedings of the International Conference on High Performance Computing and Applications, Dec. 22-24, IEEE Xplore Press, Bhubaneswar. DOI: 10.1109/ICHPCA.2014.7045330

HDFS, 2017. Architecture guid. https://www.guru99.com/learn-hdfs-a-beginners-guide.html

Mao, B., S. Wu and L. Duan, 2018. Improving the SSD performance by exploiting request characteristics and internal parallelism. IEEE Trans. Computer-Aided Design Integrated Circuits Syst., 37: 472-484. DOI: 10.1109/TCAD.2017.2697961

Mirza, M. and M. Nagori, 2017. Optimizing task assignment in hadoop using an efficient job size-based scheduler. Proceedings of the International Conference on Intelligent Computing and Control Systems, Jun. 15-16, IEEE Xplore Press, Madurai, India, pp: 1287-1292. DOI: 10.1109/ICCONS.2017.8250676

OPDi, 2017. https://www.odpi.org/

Ouyang, X., H. Zhou, S. Clement, P. Townend and J. Xu, 2017. Mitigate data skew caused stragglers through ImKP partition in MapReduce. Proceedings of the IEEE 36th International Performance Computing and Communications Conference, Dec. 10-12 San Diego.

Park, J., 2016. Improving the performance of HDFS by reducing I/O using adaptable I/O system. Proceedings of the International Conference on Electrical, Electronics and Optimization Techniques, Mar. 3-5, IEEE Xplore Press, Chennai, India, pp: 3139-3144. DOI: 10.1109/ICEEOT.2016.7755280

Shankar, V. and R. Lin, 2017. Performance study of CEPH storage with intel cache acceleration software: Decoupling hadoop MapReduce and HDFS over Ceph storage. Proceedings of the IEEE 4th International Conference on Cyber Security and Cloud Computing (SCC' 7), New York, USA, pp: 10-13.

Shi, J., Y. Qiu, U.F. Minhas, L. Jiao and C. Wang *et al*., 2015. Clash of the titans: MapReduce Vs. spark for large scale data analytics. 41st International Conference on Very Large Data Bases, Kohala Coast, Hawaii, pp: 2110-2121. DOI: 10.14778/2831360.2831365

Tang, X., 2016. Evaluating HDFS I/o performance on virtualized systems. University of Wisconsin-Madison.

YARN, 2017. https://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-site/YARN.html

Ye, M., J. Wang, J. Yin and X. Zhang, 2016. Accelerating I/O performance of SVM on HDFS. Proceedings of the IEEE International Conference on Cluster Computing, Sept. 12-16, IEEE Xplore Press, Taipei, Taiwan, pp: 132-133. DOI: 10.1109/CLUSTER.2016.71