Hadoop Performance Tuning based on Parameter Optimization

Wei Wang[1,a*], Yong Shi[1,b], and Xin Liu[1,c], Yihong Feng[1,d], and Ning Tao[1,e]

[1] BIM Computing Research Center, Shenyang jianzhu University, Shengyang, 110168, China

[a]519836646@qq.com, [b]674365154@qq.com, [c]6620274@qq.com,

[d]1009523744 @qq.com, [e]23010150@qq.com

* The Corresponding Author

Keywords: Hadoop; TeraSort; Parameter optimization; Performance tuning

Abstract. In order to better verify that Hadoop performance can be improved through optimization of parameters, we can use the following test methods: benchmarking, stability testing, high availability testing, scalability testing, and security testing. In this paper, the benchmark test method is used to verify the optimization of parameters and to optimize the performance of Hadoop. This article mainly focuses on the 17 parameters in Tab.1. The optimization results are shown in Tab.3. The optimization of the parameters was verified by the execution time of the TeraSort algorithm in the benchmark test. During the experiment, the CPU and memory utilization rate, disk IO and network IO throughput and other indicators were collected. Fig.1-3 fully illustrates the comparison between Hadoop and TeraSort algorithm after parameter default value and parameter adjustment. The experimental results show that after the Hadoop parameters are adjusted and optimized, the Hadoop performance tuning is achieved under certain conditions.

The Introduction

The big data software platform is mainly composed of distributed file systems (such as HDFS), distributed computing systems (such as MapReduce), NoSQL databases (such as HBase), distributed data warehouses (such as Hive), and distributed databases (MPP)[1,2].

Table 1 Hadoop parameter list

No	category	parameter name	default
1	YARN	yarn.nodemanager.resource.memory-mb	5G
2	YARN	yarn.nodemanager.resource.cpu-vcores	8
3	MapRedcue	mapreduce.job.reduces	1
4	HDFS	dfs.blocksize	128M
5	MapRedcue	mapreduce.map.memory.mb	1G
6	MapRedcue	mapreduce.map.java.opts	756M
7	MapRedcue	mapreduce.task.io.sort.mb	200M
8	YARN	yarn.scheduler.maximum-allocation-mb	2G
9	MapRedcue	mapreduce.map.output.compress	FALSE
10	MapRedcue	mapreduce.map.output.compress.codec	org.apache.hadoop.io.compress.DefaultCodec
11	MapRedcue	mapreduce.reduce.memory.mb	1G
12	MapRedcue	mapreduce.reduce.java.opts	756m
13	MapRedcue	mapreduce.reduce.shuffle.input.buffer.percent	0.7
14	MapRedcue	mapreduce.output.fileoutputformat.compress	FALSE
15	MapRedcue	mapreduce.output.fileoutputformat.compress.codec	org.apache.hadoop.io.compress.DefaultCodec
16	MapRedcue	mapreduce.job.reduce.slowstart.completedmaps	0.05
17	MapRedcue	mapreduce.job.reduces	25

Provides storage, management and computing capabilities for big data. The big data software platform mainly includes open source Hadoop, Spark, etc., and is generally deployed on a general hardware platform.

Because Hadoop itself contains many parameters, the relationship between parameters is also more complicated. After simple experimental verification, this paper mainly optimizes the 17 parameters in Tab.1, and the parameter optimization is to follow the order, superimposed way to achieve Hadoop performance tuning[3].

Research on Experiment Design and TeraSort Algorithm

Experiment Design. In order to better verify that Hadoop performance can be improved through optimization of parameters, we can use the following test methods: benchmarking, stability testing, high availability testing, scalability testing, and security testing. In this paper, the benchmark test method is used to verify the optimization of parameters and to optimize the performance of Hadoop.

Benchmarking is an activity that measures and evaluates software performance metrics. At a time, benchmarks can be used to establish a known level of performance (referred to as the baseline), and a benchmark test can be performed after the hardware and software environment of the system has changed to determine the effect of those changes on performance.

There are a variety of test methods for benchmarking. such as: TestDFSIO, TeraSort, WordCount, MRBench, NNBench, simulator HDFS daily import/export, DistCP and so on. According to Hadoop's Map Reduce process, this paper mainly uses the TeraSort algorithm to verify that the 17 parameters in Tab.1 are adjusted, and the Hadoop performance is affected and optimized under specific conditions. The Hadoop test environment is based on the pattern of 1 Name Node and 4 Date Node.

TeraSort Algorithm. TeraSort's work principle requires the following steps. Firstly, sampling. Secondly, map marked the data record (identify the reduce number belongs). Then reduce local sort[4]. At the end, output sequentially.

1TB ordering is commonly used to measure the data processing capabilities of a distributed data processing framework. TeraSort is a sorting job in Hadoop. In 2008, Hadoop won the first place in the 1TB sorting benchmark assessment, which took 209 seconds. TeraSort cleverly uses Hadoop's MapReduce mechanism to achieve the purpose of the Sort, and the perfect combination with the Hadoop mechanism may be an important reason for its excellent sorting results. And because of this, we can use TeraSort to test Hadoop on the cluster, which will have high test utilization value.

Terasort's feature is hybrid, its data type is text, and the data source required for the test process is Hadoop's own TeraGen generation. TeraSort is just a small tool and may be insignificant compared to the production application. But a gadget, if it can be excavated, will have great value behind it. Especially for testing, if there is more knowledge of background knowledge, a gadget can be converted into many convenient and valuable test cases. And if you can infer something from a gadget, it can also provide value for testing elsewhere.

Table 2 Experimental procedure

step	content
1	hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar teragen -Dmapred.map.tasks=512 5368709120 /terasort/input1G
2	Hadoop fs –du –h /terasort
3	echo 3 > /proc/sys/vm/drop_caches
4	hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar terasort -Dmapred.reduce.tasks=95 /terasort/input1G /terasort/output
5	hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar teravalidate /terasort/output /terasort/01-validate
6	hadoop fs -ls /terasort/output
7	hadoop fs -cat /terasort/01-validate/part-r-00000

According to the complex relationship between them, the 17 parameters in Tab.1 are divided into 7 groups of parameter groups for adjustment. From the first group to the seventh group, the adjustment parameter values are accumulated in sequence, see Tab.3 for details. Verified by the TeraSort algorithm the correctness of parameter adjustment. After each adjustment of the parameters, we need to execute the TeraSort algorithm on Hadoop. The execution steps of the TeraSort algorithm are detailed in Tab.2. During the experiment, the execution time of the statistical algorithm is referenced. The PAT (Performance Analysis Tool) tool was used to monitor the CPU, memory, disk IO and network IO of the algorithm during the execution of the algorithm.

The single file size in the experimental data is 1G, the number of files is 512, and the total file size is 512G.

Test and Analysis

In Tab.3, we can understand that there are mutual influences between parameters, so we try to divide them into 7 groups for testing. The test found that as the parameters are adjusted, the execution time of the TeraSort algorithm is decreasing, which fully explains the parameters. With the optimization of parameters, the Hadoop performance is optimized.

Table 3 TeraSort algorithm execution time comparison

No	parameter name	default	optimization	Time
I	yarn.nodemanager.resource.memory-mb	5G	100G	1535 s
	yarn.nodemanager.resource.cpu-vcores	8	25	
	mapreduce.job.reduces	1	25	
II	dfs.blocksize	128M	1024M	1233 s
III	mapreduce.map.memory.mb	1G	4G	1126 s
	mapreduce.map.java.opts	756M	3G	
	mapreduce.task.io.sort.mb	200M	2047M	
	yarn.scheduler.maximum-allocation-mb	2G	4G	
IV	mapreduce.map.output.compress	FALSE	TRUE	990s
	mapreduce.map.output.compress.codec	org.apache.hadoop.io.compress.DefaultCodec	Lz4Codec	
V	mapreduce.reduce.memory.mb	1G	4G	973s
	mapreduce.reduce.java.opts	756m	3G	
	mapreduce.reduce.shuffle.input.buffer.percent	0.7	0.9	
VI	mapreduce.output.fileoutputformat.compress	FALSE	TRUE	967s
	mapreduce.output.fileoutputformat.compress.codec	org.apache.hadoop.io.compress.DefaultCodec	Lz4Codec	
VII	mapreduce.job.reduce.slowstart.completedmaps	0.05	0.8	871s
	mapreduce.job.reduces	25	95	

According to Tab.3, we can understand that with the adjustment of the default value of 17 parameters, the execution time of the algorithm is obviously shortened. In the case that 17 parameters are all at the default value, the execution time of the TeraSort algorithm is 21776 seconds.

Fig.1 shows the comparison between the default value of the parameters and the execution time of the TeraSort algorithm after parameter adjustment. After the parameters are optimized, the execution time of the algorithm is greatly improved.

Fig.2 shows clearly parameters default values and adjusted during the execution of the TeraSort algorithm, Hadoop compares the cpu and memory usage rate of the server, the occupancy rate of the cpu is obviously increased, and the memory usage rate is also increased.

Fig.3 clearly shows the implementation of the TeraSort algorithm after parameter default value and parameter adjustment. Hadoop significantly increased the server's DiskIO and NetworkIO throughput.

They are verified that they can, to some extent, improve the performance of Hadoop.

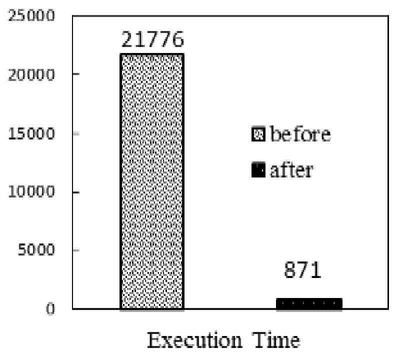

Figure 1. Execution time

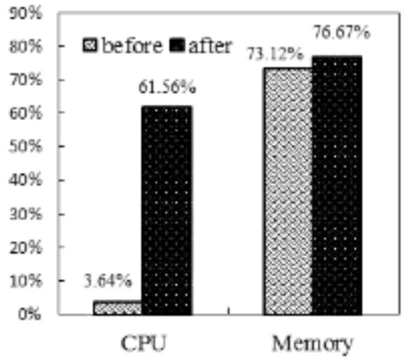

Figure 2. Occupancy

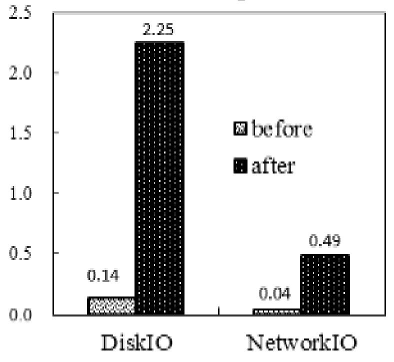

Figure 3. Throughput

Summary

This paper mainly validates the parameters of Hadoop through the running time of the TeraSort algorithm in the benchmark test, and then optimizes the performance of Hadoop. Hadoop performance tuning involves not only the performance tuning of Hadoop itself, but also the tuning of systems, such as lower-level hardware, operating systems, and Java virtual machines. In order to make better use of Hadoop, we should optimize the Hadoop parameters according to the actual application scenarios and requirements so that the Hadoop performance can be optimized under certain conditions.

Reference

[1] Saravanan S, Karthick K E, Balaji A, et al. Performance Comparison of Apache Spark and Hadoop Based Large Scale Content Based Recommender System[M]// Intelligent Systems Technologies and Applications. 2018:66-73.

[2] Ashlesha S, R. M. A Review of Hadoop Ecosystem for BigData[J]. International Journal of Computer Applications, 2018, 180(14):35-40.

[3] Trivedi M, Nambiar R. Lessons Learned: Performance Tuning for Hadoop Systems[M]// Performance Evaluation and Benchmarking. Traditional - Big Data - Interest of Things. 2017.

[4] Pahl C. Performance and Energy Optimization on Terasort Algorithm by Task Self-Resizing[J]. Information Technology & Control, 2014, 44(1):30-40.