Advanced Materials Research
ISSN: 1662-8985, Vols. 926-930, pp 1988-1992
doi:10.4028/www.scientific.net/AMR.926-930.1988

Online: 2014-05-23

Principle Study and Usage of Hadoop Benchmark

Cong Zheng[1,a], Liqing Zhou[1,b]

[1]Guilin University of Technology Library, Guangxi, China

[a]zhengcong35@qq.com, [b]303550889@qq.com

Keywords: Hadoop, Cloud Computing, Benchmark, DFSIO

Abstract: Hadoop is the most popular Cloud Computing platform with source code. Its maturity and completeness of ecosystem attract a large number of technical personnel and companies. With the development of a variety of assistive technologies, it is more and more easily to build Hadoop platform. But how to judge a Hadoop platform is built successfully and whether robust enough to be able to put into actual production also requires a series of benchmark tests. This article will introduce the principles and usage methods of Hadoop platform benchmark tools such as Slive, DFSIO and TeraSort. Then compare their results and make a comparison.

Introduction

Hadoop is the most popular open source cloud computing platform. Its maturity and completeness of the ecosystem attract a large number of technical personnel and companies. With the development of a variety of assistive technologies, it is more and more easily to build Hadoop platform. But how to judge a Hadoop platform is built successfully or not and whether robust enough to be able to put into actual production also requires a series of benchmark tests. Benchmark is used to measure performance using a specific indicator (cost per unit of measure, productivity per unit of measure, cycle time of x per unit of measure or defects per unit of measure) resulting in a metric of performance that is then compared to others[1]. A newly built Hadoop platform can be comprehensive tested by different benchmarks in terms of I/O, network, memory, response time. Through this way administrator can verify its working condition and preheating the new platform before putting it into production. Currently the most maturely Hadoop platform benchmarking tools are Slive (Stress Test for Live Data Verification), DFSIO, TeraSort. This article will introduce their working principle and the usage of methods one by one and then summarize and analyze their testing results.

The configuration of the Hadoop platform tested in this paper is shown in Table 1, all tests were carried out under this environment. We will not repeat it later.

Table 1 Configuration List

No. of cluster	ThinkStation D20*5
CPU	Xeon E5645 12*2.4GHz*5
Memory	8GB 1333 ECC*5
OS	RHEL 6.3 x86_64
Network	100MB/sec
Hadoop	2.2.0
Replication	3

Slive（Stress Test for Live Data Verification）

Slive is based on DFSIO framework and works as MapReduce program. It can be set a variety of Map operations and the corresponding number of Reduce operations. Map does HDFS operations and evaluates the results; Reduce aggregate and average evaluation. Map execute in each round according to the execution sequence shown in Table 2 [2].

Table 2 Operation Sequence

Sequence	Operation
Op1	Select an action randomly from the write, read, append, ls, mkdir, delete. The operations can be configured by percentage, equality for the default.
Op2	Randomly select a file name, in the order of increase.
Op3	Randomly select file size of read, write, append operation.Redefine the scope by specifies parameter [min, max].
Op4	Randomly select HDFS block size for write operation.
Op5	Randomly select replication factor.
Op6	Randomly selected offset within the file length for read operation.
Op7	Verify the data for read operation.
Op8	Summary of the performance parameters

For each type of operation, only one Reduce summarize, $NUM_{Reduce} = NUM_{OperationType}$.

Enter command [hdfs@hadooptest1~]$ hadoop org.apache.hadoop.fs.slive.SliveTest -read 50 -write 50 to run Slive. The command includs 50% read operation and 50% write operation. Results is 'bytes_written = 2.055 MB/sec, bytes_read = 32.335 MB/sec'. 32.355MB/sec represents that Hadoop platform can easily breaking the bandwidth limitations of 10MB/sec. and compared to network bottlenecks (Networkmax) the write speed reduced by nearly 80%. After extensive study we found that the test started 10 Map, evenly distributed 2.5 process on per NodeManager; Default replication factor (Replication) is 3, virtually every written operation have two network traffic. So we get Eq.1.

$$\frac{Network_{max}}{2.5Map*(Replication-1)}=2MB/sec\approx 2.055MB/sec \quad (1)$$

The test runs over in 67m43.456s, platform's network and memory usage is shown in Fig.1, Fig.2. As shown in the figure, usage of network resources remain at saturation, and memory usage remained at 30%. Thus we can conclude that Slive test can well benchmark newly built Hadoop platform's I/O and network switching performance.

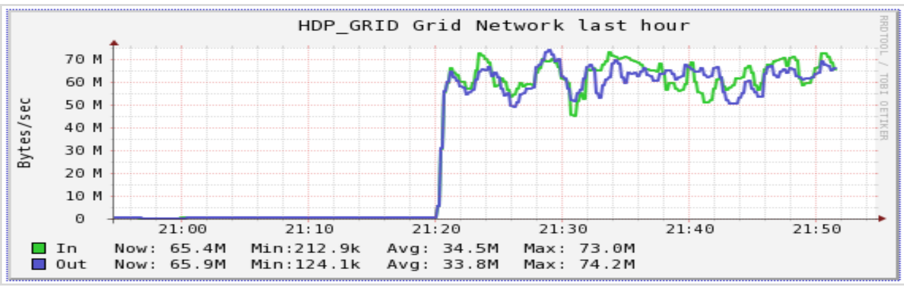

Fig.1. Network usage of Slive test

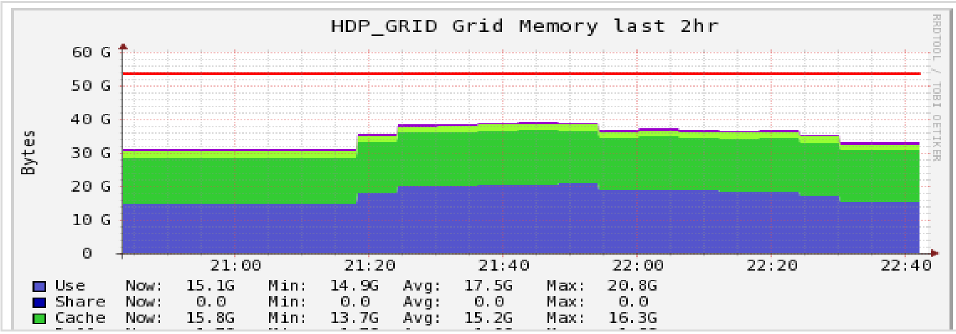

Fig.2. Memory usage of Slive test

DFSIO

DFSIO is the benchmark for HDFS operations, including modules of generating data, submitting job, summarize complete time which are needed for the evaluation of large-scale data processing. DFSIO emphasis on I/O and network performance. It has two steps: 1st generating data through MapReduce program and automatically upload to HDFS, at the same time record writing speed and time elapsed; 2nd do read operation for data and record reading speed and time elapsed.

Each test can pick one operation from read, write, append, clean, -nrFiles specified number of files generated, -size specifies the size of files generated. DFSIO underlying design provides that a Map generates a file. Here we create 10 1GB files, in order to test the efficiency of reading and writing large data on Hadoop platform. First, create files through command [hdfs@hadooptest1 ~]$ hadoop org.apache.hadoop.fs.TestDFSIO -write -nrFiles 10 -size 1GB. Then execute DFSIO read command [hdfs @ hadooptest1 ~] $ hadoop org.apache.hadoop.fs. TestDFSIO -read-nrFiles 10-size 1GB. The test runs over in 14m24.298s. Testing result is shown in Table 3.

Table 3 Testing results of DFSIO

----- TestDFSIO ----- : write	----- TestDFSIO ----- : read
Date & time: Sat Dec 28 11:03:20 CST 2013	Date & time: Sat Dec 28 11:21:24 CST 2013
Number of files: 10	Number of files: 10
Total MBytes processed: 10240.0	Total MBytes processed: 10240.0
Throughput mb/sec: 1.4648606536997888	Throughput mb/sec: 19.425948058353725
Average IO rate mb/sec: 1.4683821201324463	Average IO rate mb/sec: 22.23957633972168
IO rate std deviation: 0.07238264329665385	IO rate std deviation: 8.39188692740482
Test exec time sec: 755.592	Test exec time sec: 98.987

From the testing results we can find two indicators to measure network performance: Throughput and Average IO rate. Both of two indicators are based on the calculating of reading and writing 's data size and time-consuming of each Map, differ only in calculation. Let N be the number of Map, file size as $filesize_i$, time-consuming of each reading and writing is $time_i$, there are Eq.2 and Eq.3. Throughtput (N) represents the population mean, Average IO rate (N) represents the average of the sample mean. The closer the two values are, the smoother the platform is running and the more stable of resource usage. Calculation shows that the agreement rate of writing operation's Throughtput and Average IO is 99.998%, the reading operation's is 87.349%. It would seem the tested platform has stable resource consumption and is suitable for reading and writing operation.

$$\text{Throughput(N)}=\frac{\sum_{i=1}^{N}\text{filesize}_i}{\sum_{i=1}^{N}\text{time}_i} \qquad (2)^{[3]}$$

$$\text{Average IO rate(N)}=\frac{\sum_{i=1}^{N}\text{rate}_i}{N}=\frac{\sum_{i=1}^{N}\frac{\text{filesize}_i}{\text{time}_i}}{N} \qquad (3)^{[3]}$$

TeraSort

TeraSort is designed to rapid sequence TB level data, now has reached PB level [4,8]. It can test framework of HDFS and MapReduce at the same time. A complete TeraSort benchmarks include the following steps: 1st generate confusion large data by running TeraGen; 2nd sort the large data by running Terasort; 3rd validate results by running TeraValidate.

Generate test files. Run command [hdfs@hadooptest1 ~]$ hadoop org.apache.hadoop. examples.terasort. TeraGen 10000000 /outputpath to create a 1GB data file in /outputpath. TeraSort comply data format of gensort Data Generator [5]. Each data consists of three parts and data size is 100B. So the first parameter takes 100B as a unit. A complete data row of TeraSort is shown in Table 4.

Table 4 Data format of Terasort

<10 bytes key>	<10 bytes rowid>	<80 bytes filler>
.t^#|v$2\	3	AAAAAAAAAABBBBBBBBBBCCCCC CCCCC DDDDDDDDDEEEEEEEEEEFFFFFFFF FFGGGGGGGGGGHHHHHHHH

The first 10B randomly selects characters from the character table and generate key for each data row, the 2nd 10B is randomly generated rowid with numeric types, the 3rd 8*10B consists of 8 runs of 10 characters from ‘A’ to ‘Z’. Sort carried out only for the key, rowid and filler is filled in order to simulate the real environment’s data load[6].

Sorting. Before TeraSort, MapReduce sorting program only uses multiple Maps and one Reduce. This leads to none concurrent operations at Reduce stage. TeraSort improves this situation and achieves concurrent Map and Reduce operation.

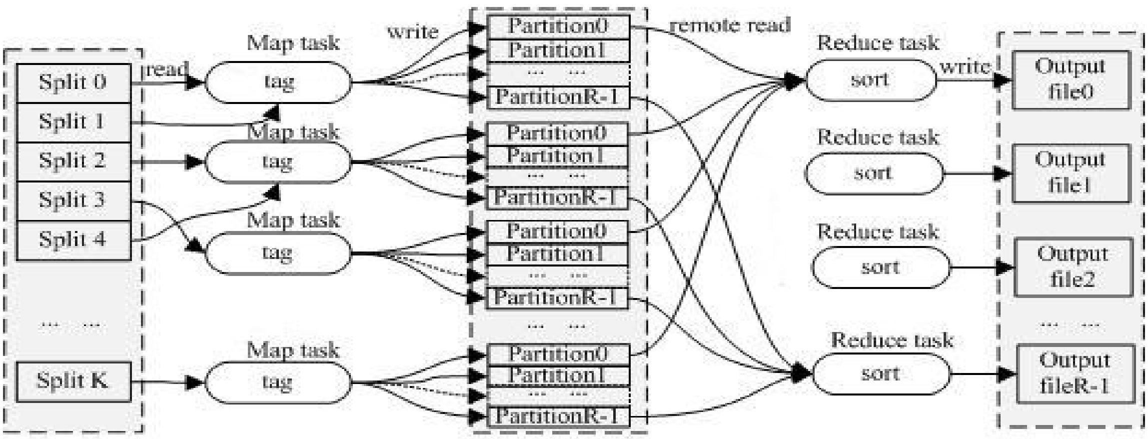

Fig. 3. Sorting process of Terasort

Firstly, each Map will divided the data block it get into N files(N equals the number of Reduce operation); the data in i-th block has to be larger than data in the (i+1)-th block. At reduce stage, the i-th Reduce process deal with the i-th data block comes from the i-th Map process. So the data generated by i-th Reduce is larger than (i+1)-th Reduce. Finally we will get sorting results by output the N Reduce's results in order[7]. The whole sorting process is shown in Fig 3.

Validate sorting results. TeraValidate creates a Map process for each file generated in paragraph 3.2 in order to validate the correctness. If something is wrong, Reduce will output information shown in Fig. 4.

```
error	misorder in part-00000 last: '7HSifVqk)-' current: '7HL.k5qzBZ'
error	misordered keys last: part-00000:end 'O(ju`RkUR_' current: part-00001:begin '7HL.k5qzBZ'
```

Fig. 4. Error information of disorder data

Conclusions

It can be revealed through analysis of principles and discovery of usage for Slive, DFSIO and Terasort that Slive and DFSIO mainly test I/O and network performance, Terasort is suitable for testing job with large amount of data. These three tools basically encompass all aspects of the stress test. So, after the establishment of Hadoop platform, these tools can be run to check the platform configuration and performance. But how to optimize the platform after finding problems requires further experimental and research.

Acknowledgements

2013 Guangxi Scientific Research and Development Program-Platform building and demonstration based on Hadoop for popularization of geological disasters. Project No. 1347016-1-4.

2013 Guilin Scientific Research and Development Program-Information service platform research and demonstration for Guilin agricultural science and technology. Project No. 20130125.

References

[1] Information on http://en.wikipedia.org/wiki/Benchmarking

[2] Information on https://issues.apache.org/jira/secure/attachment/12448004/SLiveTest.pdf.

[3] Information on http://mail-archives.apache.org/mod_mbox/hadoop-common-user/200901.mbox/%3C496EACE2.2090007@yahoo-inc.com%3E.

[4] Information on http://developer.yahoo.com/blogs/hadoop/hadoop-sorts-petabyte-16- 25-hours-terabyte-62-422.html.

[5] Information on http://www.ordinal.com/ gensort.html.

[6] Changkyu Kim, Jongsoo Park, Nadathur Satish, Hongrae Lee, Pradeep Dubey and Jatin Chhugani: *ACM SIGMOD International Conference on Management of Data* (New York 2012). Vol.1, p841-850.

[7] Information on http://sortbenchmark.org/YahooHadoop.pdf.

[8] Information on http://sortbenchmark.org/Yahoo2009.pdf