分布式文件系统性能测试

- HADOOP 系统提供了丰富的测试，有些测试在公开的网上并不经常引用，我们会适当地加大测试范围。以保证我们的集群稳定性。

- 下图为 hadoop-mapreduce-client-jobclient-2.7.3-tests.jar r 文件中的测试程序

测试	***描述***
DFSCIOTest	测试 libhdfs 中的分布式 I/O 的基准。Libhdfs 是一个为 C/C++应用程序提供 HDFS 文件服务的共享库。
DistributedFSCheck	文件系统一致性的分布式检查。
TestDFSIO	分布式的 I/O 基准。
clustertestdfs	对分布式文件系统的伪分布式测试。
dfsthroughput	测量 HDFS 的吞吐量。
filebench	SequenceFileInputFormat 和 SequenceFileOutputFormat 的基准，这包含 BLOCK 压缩，RECORD 压缩和非压缩的情况。 TextInputFormat 和 TextOutputFormat 的基准，包括压缩和非压缩的情况。
loadgen	通用的 MapReduce 加载产生器。
mapredtest	MapReduce 作业的测试和检测。
mrbench	创建大量小作业的 MapReduce 基准。
nnbench	NameNode 的性能基准。
testarrayfile	对有键值对的文本文件的测试。
testbigmapoutput	这是一个 MapReduce 作业，它用来处理不可分割的大文件来产生一个标志 MapReduce 作业。
testfilesystem	文件系统读写测试。
testipc	Hadoop 核心的进程间交互测试。
testmapredsort	用于校验 MapReduce 框架的排序的程序。

testrpc	对远程过程调用的测试。
testsequencefile	对包含二进制键值对的文本文件的测试。
testsequencefileinputformat	对序列文件输入格式的测试。
testsetfile	对包含二进制键值对文本文件的测试。
testtextinputformat	对文本输入格式的测试。
threadedmapbench	对比输出一个排序块的 Map 作业和输出多个排序块的 Map 作业的性能。

Valid program names are:
DFSCIOTest: Distributed i/o benchmark of libhdfs.
DistributedFSCheck: Distributed checkup of the file system consistency.
JHLogAnalyzer: Job History Log analyzer.
MRReliabilityTest: A program that tests the reliability of the MR framework by injecting faults/failures
NNdataGenerator: Generate the data to be used by NNloadGenerator
NNloadGenerator: Generate load on Namenode using NN loadgenerator run WITHOUT MR
NNloadGeneratorMR: Generate load on Namenode using NN loadgenerator run as MR job
NNstructureGenerator: Generate the structure to be used by NNdataGenerator
SliveTest: HDFS Stress Test and Live Data Verification.
TestDFSIO: Distributed i/o benchmark.
fail: a job that always fails
filebench: Benchmark SequenceFile(Input|Output)Format (block,record compressed and uncompressed), Text(Input|Output)Format (compressed and uncompressed)
largesorter: Large-Sort tester
loadgen: Generic map/reduce load generator
mapredtest: A map/reduce test check.
minicluster: Single process HDFS and MR cluster.
mrbench: A map/reduce benchmark that can create many small jobs
nnbench: A benchmark that stresses the namenode.
sleep: A job that sleeps at each map and reduce task.
testbigmapoutput: A map/reduce program that works on a very big non-splittable file and does identity map/reduce
testfilesystem: A test for FileSystem read/write.
testmapredsort: A map/reduce program that validates the map-reduce framework's sort.
testsequencefile: A test for flat files of binary key value pairs.
testsequencefileinputformat: A test for sequence file input format.
testtextinputformat: A test for text input format.
threadedmapbench: A map/reduce benchmark that compares the performance of maps with multiple spills over maps with 1 spill

- TestDFSIO 测试
- nnbench 测试
- mrbench 测试
- 文件系统一致性
- TeraSORT BENCH
- SliveTest

测试之前，需要启动 yarn 的调度，并确保文件系统工作正常：

```
#  start-yarn.sh
#  yarn node –list
#  hadoop fs –ls  /
```

在执行测试或结束时需要查看测试 LOG，在命令后要指明该文件的保存路径，

```
bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-client-jobclient-2.6.0-bc1.3.2-tests.jar TestDFSIO -read -nrFiles 10 -fileSize 10B -resFile /tmp/TestDFSIOresults.log
```

LOG 是一个本地 LINUX 文件。

基本的性能指标，仅供参考：

IO 读 100Mbps

IO 写 60Mbps

由于大数据系统，是一次性写，多次读的特点，更多地关注读的性能。

对应的硬盘的读写速度通常是

5400 转的笔记本硬盘：50-90MB 每秒

7200 转的台式机硬盘：90-190MB 每秒

SSD 固态硬盘由于工作原理与机械硬盘不同并无较为准确的范围，

200MB 每秒~500MB 每秒

sata2 接口的极限是 300M，但实际达不到理论速度，大约是 260 左右就到了极限了。

sata3.0 的固态盘是没问题的

我们说的读，指的是硬盘到内存，如果没有缓存，可能就是这个速度，但如果缓存，

如果三个节点都是本地：

```
read file, block_index: 0cost time: 5656us
19/03/08 15:43:38 INFO fs.TestDFSIO: ----- TestDFSIO ----- : read
19/03/08 15:43:38 INFO fs.TestDFSIO:                Date & time: Fri Mar 08 15:43:38 CST 2019
19/03/08 15:43:38 INFO fs.TestDFSIO:            Number of files: 3
19/03/08 15:43:38 INFO fs.TestDFSIO: Total MBytes processed: 3072.0
19/03/08 15:43:38 INFO fs.TestDFSIO:         Throughput mb/sec: 2096.928327645051
```

19/03/08 15:43:38 INFO fs.TestDFSIO: Average IO rate mb/sec: 2428.873291015625
19/03/08 15:43:38 INFO fs.TestDFSIO: IO rate std deviation: 788.7781920102299
19/03/08 15:43:38 INFO fs.TestDFSIO: Test exec time sec: 20.674
19/03/08 15:43:38 INFO fs.TestDFSIO:

一般写的 IO 大概在 30Mbps，需要向各个节点复制备份（有的通过网络），这个速度大概是平时硬盘复制的 1/2-1/4 ？(估计)。

下图是不同品牌的usb3.0机械移动硬盘的速度对比图。

牌	型号	参考价	顺序读取	顺序写入	随机读取	随机写入
LaCie	PorscheDesign	569	130.78	130.72	29.31	56.28
创见	TS1TSJ25M3	439	129.87	129.13	25.70	55.16
索尼	HD-E1	489	125.38	125.10	20.91	58.25
希捷	STBX1000	419	127.27	127.30	17.21	59.63
LaCie	RuggedMini	749	129.12	130.20	19.06	54.06
希捷	STDR1000	439	102.78	101.78	19.54	51.48
cool－fish	S2-1T	599	111.37	112.25	17.16	34.35
西部数据	WDBUZG0010	409	115.97	115.90	16.92	32.51
奥睿	MS2510	599	116.47	115.57	17.13	26.83
西部数据	WDBTYH0010	539	107.27	106.60	17.04	29.11
西部数据	WDBGPU0010	419	106.27	106.37	17.57	29.23
蓝硕	RL-2100	799	111.88	110.23	16.32	27.64
HGST	TouroMobile	389	109.10	108.53	16.09	28.50
东芝	HDTH310E	419	110.70	112.27	18.42	20.80
爱国者	HD816	649	110.43	110.35	17.42	22.69
东芝	HDTC810H	435	109.55	109.53	17.24	21.53
Maxtor	M3	379	114.18	111.50	16.92	18.48
朗科	K360	379	111.17	109.88	17.72	13.29
联想	F360S	449	110.20	109.75	17.97	12.94
东芝	HDTB110A	399	108.90	106.05	17.85	14.19
埃森客	B52	359	108.88	[illegible]	[illegible]	[illegible]

下图为不同品牌的固态移动硬盘的速度对比图

牌	型号	规格	参考价	顺序读	顺序写	随机读	随机写
三星	T3	250G	799	428.53	375.45	190.41	198.50
闪迪	SSDEXT	240G	1288	428.03	354.18	148.98	180.20
朗科	Z2	256G	659	427.32	289.22	150.21	182.08
索尼	SL-BG2	250G	759	421.98	358.83	145.99	173.88
爱国者	SD01S	240G	699	267.67	251.05	111.85	136.44
创见	ESD400	256G	799	426.58	315.		

HDFS 的参考数据：（初期目标）

查看运行的情况

```
# cat TestDFSIO_results.log
----- TestDFSIO ----- : write
            Date & time: Tue Nov 21 14:53:44 CST 2017
        Number of files: 10
Total MBytes processed: 100.0
     Throughput mb/sec: 19.485580670303975
Average IO rate mb/sec: 24.091276168823242
 IO rate std deviation: 9.242316274402379
    Test exec time sec: 63.103


----- TestDFSIO ----- : read
            Date & time: Tue Nov 21 15:04:33 CST 2017
        Number of files: 10
Total MBytes processed: 100.0
     Throughput mb/sec: 617.283950617284
Average IO rate mb/sec: 688.1331176757812
 IO rate std deviation: 182.42935237458195
    Test exec time sec: 36.148
```

结果说明：

Total MBytes processed ： 总共需要写入的数据量 ＝＝》 512X1000

Throughput mb/sec ：总共需要写入的数据量/（每个 map 任务实际写入数据的执行时间之和（这个时间会远小于 Test exec time sec））＝＝》512000/(map1 写时间+map2 写时间+...)

Average IO rate mb/sec ：（每个 map 需要写入的数据量/每个 map 任务实际写入数据的执行时间）之和/任务数＝＝》(512/map1 写时间＋512/map2 写时间+...)/1000，所以这个值跟上面一个值总是存在差异。

IO rate std deviation ：上一个值的标准差

Test exec time sec ：整个 job 的执行时间

集群比较小的情况下，这个测试并发 map（mapred.tasktracker.map.tasks.maximum）数越少，则显示出来的速度会越快。

一、TestDFSIO 测试

TestDFSIO 的用法如下：

Usage: TestDFSIO [genericOptions] -read [-random | -backward | -skip [-skipSize Size]] | -write | -append | -clean [-compression codecClassName] [-nrFiles N] [-size Size[B|KB|MB|GB|TB]] [-resFile resultFileName] [-bufferSize Bytes] [-rootDir]

注意：在执行 read ，先要执行 write 创建所需要的数据；每个一个 case 执行完，都用 clean 清空测试环境。

1.清除命令

hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-jobclient-2.7.3-tests.jar TestDFSIO –clean

查看参数

hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-jobclient-2.7.3-tests.jar TestDFSIO

2. TestDFSIO write

测试 hadoop 写的速度。

1> 向 HDFS 文件系统中写入数据，10 个文件，每个文件 1000MB，文件存放到 /benchmarks/TestDFSIO/io_data 下面。

hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-jobclient-2.7.3-tests.jar TestDFSIO -write -nrFiles 10 -size 1000MB -resFile /tmp/TestDFSIO_results.log

2> 查看写入的结果：

cat /tmp/TestDFSIO_results.log

3. TestDFSIO read

测试 hadoop 读文件的速度

1> 从 HDFS 文件系统中读入 10 个文件，每个文件大小为 1000MB

hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-jobclient-2.7.3-tests.jar TestDFSIO -read -nrFiles 10 -size 1000MB

2> 查看写入的结果：

cat TestDFSIO_results.log

脚本可参考 stressTestForParaFS.sh

BenchmarkForms 是个 vs 2012 开发的程序。其作用是将 TestDFSIO_results.log 转换为数据库，安装需要的格式进行排布。

二、nnbench 测试 [NameNode benchmark (nnbench)]

nnbench 用于测试 NameNode 的负载，它会生成很多与 HDFS 相关的请求，给 NameNode 施加较大的压力。
这个测试能在 HDFS 上创建、读取、重命名和删除文件操作。

hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-jobclient-2.7.3-tests.jar nnbench -help

NameNode Benchmark 0.4

Usage: nnbench <options>

Options:

-operation <Available operations are create_write open_read rename delete. This option is mandatory>

* NOTE: The open_read, rename and delete operations assume that the files they operate on, are already available. The create_write operation must be run before running the other operations.

-maps <number of maps. default is 1. This is not mandatory>

-reduces <number of reduces. default is 1. This is not mandatory>

-startTime <time to start, given in seconds from the epoch. Make sure this is far enough into the future, so all maps (operations) will start at the same time. default is launch time + 2 mins. This is not mandatory>

-blockSize <Block size in bytes. default is 1. This is not mandatory>

-bytesToWrite <Bytes to write. default is 0. This is not mandatory>

-bytesPerChecksum <Bytes per checksum for the files. default is 1. This is not mandatory>

-numberOfFiles <number of files to create. default is 1. This is not mandatory>

-replicationFactorPerFile <Replication factor for the files. default is 1. This is not mandatory>

-baseDir <base DFS path. default is /becnhmarks/NNBench. This is not mandatory>

-readFileAfterOpen <true or false. if true, it reads the file and reports the average time to read. This is valid with the open_read operation. default is false. This is not mandatory>

-help: Display the help statement

注意：在执行 open_read, rename and delete ，先要执行 create_write 创建所需要的数据；每个一个 case 执行完，都用 delete 清空测试环境。

1. 创建文件并写入数据

使用 4 个 mapper 和 2 个 reducer 来创建 10 个文件，每个文件 10MB

hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-jobclient-2.7.3-tests.jar nnbench -operation create_write -maps 4 -reduces 2 -bytesToWrite 10000000 -numberOfFiles 10 -replicationFactorPerFile 3 -readFileAfterOpen true

2. 读文件
使用 4 个 mapper 和 2 个 reducer 来读 10 个文件

hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-jobclient-2.7.3-tests.jar nnbench -operation open_read -maps 4 -reduces 2 -bytesToWrite 10000000 -numberOfFiles 10 -replicationFactorPerFile 3

3. rename 文件

1> 使用 4 个 mapper 和 2 个 reducer 来重命名 10 个文件
hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-jobclient-2.7.3-tests.jar nnbench -operation rename -maps 4 -reduces 2 -bytesToWrite 10000000 -numberOfFiles 10 -replicationFactorPerFile 3 -readFileAfterOpen true

4. delete 文件
1> 使用 4 个 mapper 和 2 个 reducer 来删除 10 个文件
hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-jobclient-2.7.3-tests.jar nnbench -operation delete -maps 4 -reduces 2 -bytesToWrite 10000000 -numberOfFiles 10 -replicationFactorPerFile 3

结果在
NNBench_results.log

三、mrbench 测试[MapReduce benchmark (mrbench)]

hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-jobclient-2.7.3-tests.jar mrbench -help

MRBenchmark.0.0.2

Usage: mrbench [-baseDir <base DFS path for output/input, default is /benchmarks/MRBench>] [-jar <local path to job jar file containing Mapper and Reducer implementations, default is current jar file>]

[-numRuns <number of times to run the job, default is 1>] [-maps <number of maps for each run, default is 2>] [-reduces <number of reduces for each run, default is 1>]

[-inputLines <number of input lines to generate, default is 1>] [-inputType <type of input to generate, one of ascending (default), descending, random>] [-verbose]

mrbench 会多次重复执行一个小作业，用于检查在机群上小作业的运行是否可重复以及运行是否高效。

生成每个文件 10 行，4 个 mapper，2 个 reducer，执行 5 次：

hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-jobclient-2.7.3-tests.jar mrbench -numRuns 5 -inputLines 10 -maps 4 -reduces 2

四、文件系统一致性的分布式检查

hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-jobclient-2.7.3-tests.jar DistributedFSCheck

18/05/10 19:50:22 INFO fs.DistributedFSCheck: ----- DistributedFSCheck ----- :
18/05/10 19:50:22 INFO fs.DistributedFSCheck: Date & time: Thu May 10 19:50:22 CST 2018
18/05/10 19:50:22 INFO fs.DistributedFSCheck: Total number of blocks: 5367
18/05/10 19:50:22 INFO fs.DistributedFSCheck: Total number of files: 3451
18/05/10 19:50:22 INFO fs.DistributedFSCheck: Number of corrupted blocks: 0
18/05/10 19:50:22 INFO fs.DistributedFSCheck: Number of corrupted files: 0

五、TeraSort Benchmarks

从文件系统角度出发的性能测试工具，大多不离吞吐率这个指标。转化到 HDFS 这边，则是 rpc 次数、opt 次数、sync 时长这样的指标信息，然而 Terasort 是个异类。这个工具不仅考验文件系统的性能，更是对 MR 自动排序能力的一种检测。Terasort 位于 hadoop 的 example 包中，是 SortBenchmark（http://sortbenchmark.org）排序比赛使用的标准工具，它使用了 Hadoop 默认的 IdentityMapper 和 IdentityReducer，利用 MR 自有的 sort 机制来保证每个 partition 内数据是有序的。
显然我们使用 Terasort，判断 HDFS 文件系统（其实还包含了 MR）性能好坏的依据与 SortBenchmark 是一致的，即计算数据量和运行时长的比值，根据单位时间的排序数据量来判断不同 HDFS 版本之间的性能差异。
Terasort 包含三个工具，分别是 teragen：用来生成供排序的随机数据；terasort：用来将随机数据排序；teravalidate：校验 terasort 的排序结果是否正确。
通过控制 teragen 的 map 数和 block size，我们得以检验多种测试场景下 HDFS 的性能状况。例如执行：

```
hadoop jar hadoop-current/hadoop-0.19.1-dc-examples.jar teragen -
Dmapred.map.tasks=100 -Ddfs.block.size=134217728 10000000000
/terasort/output
```

这个命令会在/terasort/output 下生成 100 个 10G 大小的文件，文件的 blocksize 是 128MB。这就类似于我们在测试单机磁盘 IO 的时候，分别要获取零碎文件吞吐率和整块大文件吞吐率数据一样。控制 teragen 生成的文件个数，文件大小，虽然总数据量一致，但是 terasort 的排序时间是有差异的。与普通磁盘不一样的地方在于，在集群资源不成为瓶颈的时候，文件越零碎，由于可以启动更多的 map 并行排序，相当于并发度提高了，terasort 的总耗时就会变得更少。

- 生成数据，teragen 会按行生成数据，每行 100 字节， 生成 10M 数据，需要行数 10Mb/100b ，生成的数据存入/teradata/10M-input

hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.3.jar teragen -Dmapred.map.tasks=10 1048576 /teradata/10M-input

- 查看数据是否已生成

hadoop fs -ls /Teradata

- 排序数据

terasort：将 teragen 生成的数据/teradata/100M-input 进行排序，将排序结果存入 /teradata/10M-output

hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.3.jar terasort -Dmapred.reduce.tasks=5 /teradata/10M-input /teradata/100M-output

- 校验数据

teravalidate：对 terasort 的排序结果进行验证，验证结果存入到/teradata/100M-validate

hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.3.jar teravalidate /teradata/10M-output /teradata/10M-validate

六、SilverTest

其主要功能是通过大量 map 制造多种 rpc 请求，检测 Namenode 的性能。我们可以设定 map 数量，每个 map 发起的 rpc 请求次数，每一种 rpc 操作占总操作的百分比，以及读写数据量、block size 等配置。下面列出 slive 可以调用的 rpc 操作种类：

ls	列出路径下所有文件和目录
append	追加写文件操作
create	创建文件操作
delete	删除文件操作
mkdir	创建目录操作
rename	重命名文件操作
read	读取文件内容操作

默认情况下，每个 map 有 1000 次操作，7 种操作均匀的随机出现。slivetest 运行时相关参数如下表所示：

maps	一共运行多少个 mapper，默认值为 10
ops	每个 map 跑多少个操作，默认值为 1000
duration	每个 map task 的持续时间，默认值为 MAX_INT，也就是无限制
exitOnError	遇到第一个 Error 是否要立即退出，默认不退出
files	最大生成文件数，默认为 10
dirSize	每个文件夹最多允许生成多少个文件，默认为 32
baseDir	SliveTest 运行后默认存放的文件根目录，默认为“/test/slive”
resFile	结果文件名，默认为“part-0000”
replication	备份数，可设置最小，最大备份数，默认为 3

blockSize	设置文件 block 大小，默认为 64M（64*1048576）
readSize	读入大小可设置为最小值，最大值形式，例如“-readSize 100,1000”，默认无限制（min=max=MAX_LONG=read entire file）
writeSize	写入大小，最小，最大形式，默认等于 blockSize（min=max=blocksize）
sleep	在不同次操作之间随机的插入 sleep，这个参数用于定义 sleep 的时间范围，设置同样是最小，最大，单位是毫秒，默认为 0）
appendSize	追加写大小，最小，最大形式，默认等于 blockSize（min=max=blocksize）
seed	随机数种子
cleanup	执行完所有操作并报告之后，清理目录
queue	指定队列名，默认为“default”
packetSize	指定写入的包大小
ls	指定 ls 操作占总操作数的百分比
append	指定 append 操作占总操作数的百分比
create	指定 create 操作占总操作数的百分比
delete	指定 delete 操作占总操作数的百分比
mkdir	指定 mkdir 操作占总操作数的百分比
rename	指定 rename 操作占总操作数的百分比
read	指定 read 操作占总操作数的百分比

SliveTest 可以给 Namenode 带来很大的压力，用来做极限情况下的压力测试非常合适。

```
19/03/08 16:55:58 INFO slive.SliveTest: Running with option list
19/03/08 16:55:58 INFO slive.SliveTest: Options are:
19/03/08 16:55:58 INFO slive.ConfigExtractor: Base directory = /test/slive/slive
19/03/08 16:55:58 INFO slive.ConfigExtractor: Data directory = /test/slive/slive/data
19/03/08 16:55:58 INFO slive.ConfigExtractor: Output directory = /test/slive/slive/output
19/03/08 16:55:58 INFO slive.ConfigExtractor: Result file = part-0000
19/03/08 16:55:58 INFO slive.ConfigExtractor: Grid queue = default
19/03/08 16:55:58 INFO slive.ConfigExtractor: Should exit on first error = false
```

19/03/08 16:55:58 INFO slive.ConfigExtractor: Duration = unlimited
19/03/08 16:55:58 INFO slive.ConfigExtractor: Map amount = 10
19/03/08 16:55:58 INFO slive.ConfigExtractor: Reducer amount = 10
19/03/08 16:55:58 INFO slive.ConfigExtractor: Operation amount = 1000
19/03/08 16:55:58 INFO slive.ConfigExtractor: Total file limit = 10
19/03/08 16:55:58 INFO slive.ConfigExtractor: Total dir file limit = 32
19/03/08 16:55:58 INFO slive.ConfigExtractor: Read size = entire file
19/03/08 16:55:58 INFO slive.ConfigExtractor: Write size = blocksize
19/03/08 16:55:58 INFO slive.ConfigExtractor: Append size = blocksize
19/03/08 16:55:58 INFO slive.ConfigExtractor: Block size = 67108864,67108864 bytes
19/03/08 16:55:58 INFO slive.ConfigExtractor: Replication amount = 3,3
19/03/08 16:55:58 INFO slive.ConfigExtractor: Operations are:
19/03/08 16:55:58 INFO slive.ConfigExtractor: LS
19/03/08 16:55:58 INFO slive.ConfigExtractor: UNIFORM
19/03/08 16:55:58 INFO slive.ConfigExtractor: 12.5%
19/03/08 16:55:58 INFO slive.ConfigExtractor: APPEND
19/03/08 16:55:58 INFO slive.ConfigExtractor: UNIFORM
19/03/08 16:55:58 INFO slive.ConfigExtractor: 12.5%
19/03/08 16:55:58 INFO slive.ConfigExtractor: MKDIR
19/03/08 16:55:58 INFO slive.ConfigExtractor: UNIFORM
19/03/08 16:55:58 INFO slive.ConfigExtractor: 12.5%
19/03/08 16:55:58 INFO slive.ConfigExtractor: DELETE
19/03/08 16:55:58 INFO slive.ConfigExtractor: UNIFORM
19/03/08 16:55:58 INFO slive.ConfigExtractor: 12.5%
19/03/08 16:55:58 INFO slive.ConfigExtractor: CREATE
19/03/08 16:55:58 INFO slive.ConfigExtractor: UNIFORM
19/03/08 16:55:58 INFO slive.ConfigExtractor: 12.5%
19/03/08 16:55:58 INFO slive.ConfigExtractor: TRUNCATE
19/03/08 16:55:58 INFO slive.ConfigExtractor: UNIFORM
19/03/08 16:55:58 INFO slive.ConfigExtractor: 12.5%
19/03/08 16:55:58 INFO slive.ConfigExtractor: READ
19/03/08 16:55:58 INFO slive.ConfigExtractor: UNIFORM
19/03/08 16:55:58 INFO slive.ConfigExtractor: 12.5%
19/03/08 16:55:58 INFO slive.ConfigExtractor: RENAME
19/03/08 16:55:58 INFO slive.ConfigExtractor: UNIFORM
19/03/08 16:55:58 INFO slive.ConfigExtractor: 12.5%
19/03/08 16:55:58 INFO slive.SliveTest: Running job:
19/03/08 16:55:58 INFO client.RMProxy: Connecting to ResourceManager at /192.168.1.41:8032
19/03/08 16:55:58 INFO client.RMProxy: Connecting to ResourceManager at /192.168.1.41:8032
19/03/08 16:56:00 INFO mapreduce.JobSubmitter: number of splits:10
19/03/08 16:56:00 INFO Configuration.deprecation: io.bytes.per.checksum is deprecated.
Instead, use dfs.bytes-per-checksum
19/03/08 16:56:00 INFO Configuration.deprecation: mapred.job.queue.name is deprecated.
Instead, use mapreduce.job.queuename

19/03/08 16:56:00 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1552026583145_0006
19/03/08 16:56:00 INFO impl.YarnClientImpl: Submitted application application_1552026583145_0006
19/03/08 16:56:00 INFO mapreduce.Job: The url to track the job: http://testing1.r1.test41:8088/proxy/application_1552026583145_0006/
19/03/08 16:56:00 INFO mapreduce.Job: Running job: job_1552026583145_0006
19/03/08 16:56:06 INFO mapreduce.Job: Job job_1552026583145_0006 running in uber mode : false
19/03/08 16:56:06 INFO mapreduce.Job: map 0% reduce 0%

Note:
看看内存情况：
free -g
同步内存文件缓存到硬盘：
sync
清空 linux 文件系统缓存：
echo 3 > /proc/sys/vm/drop_caches
再次查看内存情况：
free -g

19/03/08 16:13:31 INFO fs.TestDFSIO: ----- TestDFSIO ----- : write
19/03/08 16:13:31 INFO fs.TestDFSIO: Date & time: Fri Mar 08 16:13:31 CST 2019
19/03/08 16:13:31 INFO fs.TestDFSIO: Number of files: 10
19/03/08 16:13:31 INFO fs.TestDFSIO: Total MBytes processed: 100000.0
19/03/08 16:13:31 INFO fs.TestDFSIO: Throughput mb/sec: 6.529401667178245
19/03/08 16:13:31 INFO fs.TestDFSIO: Average IO rate mb/sec: 6.541773796081543
19/03/08 16:13:31 INFO fs.TestDFSIO: IO rate std deviation: 0.28662929943641546
19/03/08 16:13:31 INFO fs.TestDFSIO: Test exec time sec: 1628.492
19/03/08 16:13:31 INFO fs.TestDFSIO:
[root@ht1.r1.n15 ~]$hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-jobclient-2.7.3-tests.jar TestDFSIO -write -nrFiles 10 -size 10000MB -resFile /tmp/TestDFSIO_results.log