# The Report For The Cipher Throughput Test

## Cipher Throughput Test

### Introduction

The goal for this test is to evaluate the throughput of cipher directly. The test is conduct by encrypting and decrypting data with the size of 8G in order to showing the throughput of java cipher, crypto IO using *JceAesCtrCryptoCodec* and crypto IO using *OpensslAesCtrCryptoCodec*.

### Environment

Hadoop common version: 2.6.

Java versions[[1]](#footnote-1): 1.7.0\_09 and 1.7.0\_71

Openssl version: 1.0.1e-fips

Linux: 2.6.32-358.el6.x86\_64

Logic CPU number: 40

CPU info: Intel(R) Xeon(R) CPU E5-2690 v2 @ 3.00GHz

Physical CPU number: 2

L1d cache: 32K

L1i cache: 32K

L2 cache: 256K

L3 cache: 25600K

### Prerequisite

The following software needs to be installed:

* Java 1.7.0\_09 and 1.7.0\_71
* Openssl 1.0.1e-fips

### Steps

In this test, three kinds of cipher are evaluated both in java 1.7\_0\_09 and 1.7.0\_71. Here come the main steps:

1. Prepare an array in size of 0.5M or 50M filled with some random data
2. For crypto IO, initial the output stream with the corresponding codec (*JceAesCtrCryptoCodec* and *OpensslAesCtrCryptoCodec*) and for the Java cipher, initial the cipher by jceks key provider and set it to encryption mode .
3. For the crypto IO, write data into output stream created in step2 and reset the offset to ensure data output stream is using the same memory block[[2]](#footnote-2). This step will be executed for 100,000 or 1000 times. For the java cipher, the doFinal method will be executed in 100,000 or 1000 times to encrypting the same block of data.
4. Trace the time cost in step 3 which is the time used for encrypting the data.
5. For crypto IO, initial the input stream with the corresponding codec(*JceAesCtrCryptoCodec* and *OpensslAesCtrCryptoCodec*) and for the Java cipher, initial the cipher by “*jceks*” key provider and set it to decryption mode .
6. For the crypto IO, write data into input stream created in step2 and reset the offset to ensure data input stream is using the same memory block[[3]](#footnote-3). This step will be executed for 100,000 or 1000 times. For the java cipher, the doFinal method will be executed in 100,000 or 1000 times to decrypting the same block of data.
7. Trace the time cost in step 6 which is the time used for decrypting the data.

### Result

The following diagram shows the result for the cipher throughput test. It is shown that using OpensslAesCtrCtyptoCodec is much faster than the other two solutions.

From the following java cipher thoughput data table, we can see that the the cipher in JDK 1.7.0\_71 has almost twice faster throughput than that in JDK 1.7.0\_09 since 1.7.0\_71 is supporting AES-NI.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **times** | **data size(MB)** | **java cipher (AES/CTR/NoPadding)** | | | |
|
|
|
|
| encryption(MB/s) | | decryption(MB/s) | |
|
| Java  1.7.0\_09 | Java 1.7.0\_71 | Java  1.7.0\_09 | Java 1.7.0\_71 |
|
| 1 | 100000 | 0.5 | 100.12 | 248.68 | 89.43 | 196.09 |
| 2 | 100000 | 0.5 | 99.95 | 249 | 88.35 | 183.97 |
| 3 | 100000 | 0.5 | 99.99 | 248.73 | 88.24 | 194.59 |
| 4 | 1000 | 50 | 96.89 | 245.47 | 88.9 | 192.53 |
| 5 | 1000 | 50 | 97.48 | 243.75 | 89.26 | 190.25 |
| 6 | 1000 | 50 | 97.29 | 243.8 | 87.9 | 196.43 |

Java cipher throughput data

From the following Hadoop crypto IO(using Openssl) throughput data, it shows that the throughput will drop down a little bit since 50M is bigger than the L3 cache size.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **times** | **data size(MB)** | **hadoop crypto IO  (using org.apache.hadoop.crypto.OpensslAesCtrCryptoCodec)** | | | |
|
|
|
|
| encryption(MB/s) | | decryption(MB/s) | |
|
| Java  1.7.0\_09 | Java 1.7.0\_71 | Java  1.7.0\_09 | Java 1.7.0\_71 |
|
| 1 | 100000 | 0.5 | 1894.73 | 1922.12 | 1725.33 | 1913 |
| 2 | 100000 | 0.5 | 1891.07 | 1922.19 | 1718.8 | 1909.42 |
| 3 | 100000 | 0.5 | 1891.22 | 1923.74 | 1720.87 | 1831.9 |
| 4 | 1000 | 50 | 1151.54 | 1096.03 | 1183.1 | 1115.57 |
| 5 | 1000 | 50 | 1142.2 | 1150.01 | 1164.23 | 1168.25 |
| 6 | 1000 | 50 | 1109.09 | 1125.44 | 1149.72 | 1146.26 |

The conclusion is like that shown in java cipher thoughput data table. The AES-NI will have an positive impact on the throughput data.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **times** | **data size(MB)** | **hadoop crypto IO (using org.apache.hadoop.crypto.JceAesCtrCryptoCodec)** | | | |
|
|
|
|
| encryption(MB/s) | | decryption(MB/s) | |
|
| Java  1.7.0\_09 | Java 1.7.0\_71 | Java  1.7.0\_09 | Java 1.7.0\_71 |
|
| 1 | 100000 | 0.5 | 96.03 | 181.36 | 95.29 | 193.02 |
| 2 | 100000 | 0.5 | 95.84 | 174.84 | 95.88 | 187.94 |
| 3 | 100000 | 0.5 | 94.41 | 179.55 | 95.86 | 211.95 |
| 4 | 1000 | 50 | 85.16 | 178.33 | 83.5 | 183.13 |
| 5 | 1000 | 50 | 88.4 | 178.1 | 83.46 | 184.56 |
| 6 | 1000 | 50 | 87.1 | 171.01 | 86.58 | 176.08 |

## TestDFSIO Test

### Introduction

The goal is to evaluate the throughputs of reading and writing operations among the non-encryption HDFS path, the encryption HDFS path with the codec of *org.apache.hadoop.crypto.JceAesCtrCryptoCodec* or the encryption HDFS path with the codec of *org.apache.hadoop.crypto.OpensslAesCtrCryptoCodec*. In this test, the tool *TestDFSIO* is used. All read and write operations will be executed in the memory and the test data size is 20GB.

### Environment

CDH version: 5.2

Java versions: 1.7.0\_71

Openssl version: 1.0.1e-fips

Linux: 2.6.32-358.el6.x86\_64

Logic CPU number: 40

CPU info: Intel(R) Xeon(R) CPU E5-2690 v2 @ 3.00GHz

Physical CPU number: 2

L1d cache: 32K

L1i cache: 32K

L2 cache: 256K

L3 cache: 25600K

### Prerequisite

The following software needs to be installed:

* Java 1.7.0\_09 and 1.7.0\_71
* Openssl 1.0.1e-fips
* CDH 5.2(Only one data node and the node manager will be the same host as the data node[[4]](#footnote-4), set the data node’s data directory to the value of “/dev/shm/dfs/dn”[[5]](#footnote-5))

### Steps

* The non-encryption mode

1. Create a path without any encryption zone associated

*su -s /bin/bash hdfs -c "hadoop fs -mkdir /unencrypted"*

1. Execute the following command to see the throughput of writing data[[6]](#footnote-6)

*su -s /bin/bash hdfs -c "hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-client-jobclient-2.5.0-cdh5.2.0-tests.jar TestDFSIO -Dtest.build.data=/benchmark -write -nrFiles 1 -fileSize 20GB -resFile /tmp/output.txt”*

1. Drop the caching data by executing the following command:

*echo 3 > /proc/sys/vm/drop\_caches*

1. Execute the following command to see the throughput of reading data:

*su -s /bin/bash hdfs -c "hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-client-jobclient-2.5.0-cdh5.2.0-tests.jar TestDFSIO -Dtest.build.data=/benchmark -read -nrFiles 1 -fileSize 20GB -resFile /tmp/ output.txt"*

1. Clean the data created in this test by the following command:

*su -s /bin/bash hdfs -c "hdfs dfs -rmr –skipTrash /unencrypted/\*"*

* The encryption mode using org.apache.hadoop.crypto.JceAesCtrCryptoCodec

1. Create a master key used for encrypting HDFS path

*hadoop key create bmkey*

1. Create a path associated with an encryption zone

*hdfs crypto -createZone -keyName 'bmkey' -path /benchmark/*

1. Set the value of the client side's configuration "*hadoop.security.crypto.codec.classes.aes.ctr.nopadding*" to "*org.apache.hadoop.crypto.JceAesCtrCryptoCodec*"
2. Execute the following command to see the throughput of writing data:

*su -s /bin/bash hdfs -c "hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-client-jobclient-2.5.0-cdh5.2.0-tests.jar TestDFSIO -Dtest.build.data=/benchmark -write -nrFiles 1 -fileSize 20GB -resFile /tmp/output.txt”*

1. Drop the caching data by executing the following command:

*echo 3 > /proc/sys/vm/drop\_caches*

1. Execute the following command to see the throughput of reading data:

*su -s /bin/bash hdfs -c "hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-client-jobclient-2.5.0-cdh5.2.0-tests.jar TestDFSIO -Dtest.build.data=/benchmark -read -nrFiles 1 -fileSize 20GB -resFile /tmp/ output.txt"*

1. Clean the data created in this test by the following command[[7]](#footnote-7):

*su -s /bin/bash hdfs -c "hdfs dfs -rmr -skipTrash /benchmark/\*"*

* The encryption mode org.apache.hadoop.crypto.OpensslAesCtrCryptoCodec

1. Create a master key used for encrypting HDFS path

*hadoop key create bmkey*

1. Create a path associated with an encryption zone

*hdfs crypto -createZone -keyName 'bmkey' -path /benchmark/*

1. Set the value of the client side's configuration "*hadoop.security.crypto.codec.classes.aes.ctr.nopadding*" to "*org.apache.hadoop.crypto.OpensslAesCtrCryptoCodec*"
2. Execute the following command to see the throughput of writing data:

*su -s /bin/bash hdfs -c "hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-client-jobclient-2.5.0-cdh5.2.0-tests.jar TestDFSIO -Dtest.build.data=/benchmark -write -nrFiles 1 -fileSize 20GB -resFile /tmp/output.txt”*

1. Drop the caching data by executing the following command:

*echo 3 > /proc/sys/vm/drop\_caches*

1. Execute the following command to see the throughput of reading data:

*su -s /bin/bash hdfs -c "hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-client-jobclient-2.5.0-cdh5.2.0-tests.jar TestDFSIO -Dtest.build.data=/benchmark -read -nrFiles 1 -fileSize 20GB -resFile /tmp/ output.txt"*

1. Clean the data created in this test by the following command[[8]](#footnote-8):

*su -s /bin/bash hdfs -c "hdfs dfs -rmr -skipTrash /benchmark/\*"*

### Result



1. Java 1.7.0\_09 does not support AES-NI while Java 1.7.0\_71 does. And we choose these two java versions to do a contrast test. [↑](#footnote-ref-1)
2. Using the same memory is to avoid the cost of allocating the data block. [↑](#footnote-ref-2)
3. Using the same memory is to avoid the cost of allocating the data block. [↑](#footnote-ref-3)
4. It will avoid the impact from the network IO. [↑](#footnote-ref-4)
5. The directory “/dev/shm/dfs/dn” is the memory. [↑](#footnote-ref-5)
6. The command is trying to write 1 file of 20G into the specified HDFS files and then getting the throughput information for the writing operation. [↑](#footnote-ref-6)
7. The “SkipTrash” option is needed here since removing operation will move the files to the trash which is not allowed in the encryption zone. [↑](#footnote-ref-7)
8. The “SkipTrash” option is needed here since removing operation will move the files to the trash which is not allowed in the encryption zone. [↑](#footnote-ref-8)