HADOOP-17224

tasanuma

2020-08-25

Contents

Chapter 1

Root issue HADOOP-17224

1.1 Summary

Install Intel ISA-L library in Dockerfile

1.2 Description

Currently, there is not isa-l library in the docker container, and jenkins skips the natvie tests, TestNativeRSRawCoder and TestNativeXORRawCoder.

1.3 Attachments

No attachments

1.4 Comments

1. **tasanuma:** Merged to trunk. Thanks for reviewing it, [~iwasakims].
2. **ahussein:** [~tasanuma], [~iwasakims] Thank you for creating that patch. I agree that it is important that the environment does not skip any tests.

   However, IMHO that change was *never safe* to be merged into trunk without evaluating its side effect on all the test cases.

   It blocked everyone else committing to Trunk.

   Adding ISA-L triggered several failures listed under HDFS-15646 (i.e., HDFS-15643, HDFS-15654, HDFS-15461 ..etc)

   Also, the testing run out of native memory, and it is possible that the bug was triggered as aside effect of this native library.

+I suggest that:+

* This change gets reverted.

* A thorough evaluation is performed on a local environment. This includes the native unit tests, and any test that might load the new library (EC, Stripedfiles..etc).

* When the failing units are fixed, then a PR that includes all the changes may be merged.

* keep an eye on the qbt-report to evaluate the environment post the merge.

CC: [~aajisaka], [~elgoiri], [~kihwal], [~daryn], [~weichiu], [~ayushtkn], [~ebadger]

3. **tasanuma:** Thanks for reporting it, [~ahussein].

   Sorry that I didn't realize the side effects like HDFS-15643. But it seems that the bug was already there, and HADOOP-17224 is not the root cause.

   {quote}Adding ISA-L triggered several failures listed under HDFS-15646

   {quote}

   If many failed after HADOOP-17224, I agree with reverting HADOOP-17224 once even if it is not the root cause. But does ISA-L really relate to them except for HDFS-15643?

4. **iwasakims:** {quote}

   Adding ISA-L triggered several failures listed under HDFS-15646 (i.e., HDFS-15643, HDFS-15654, HDFS-15461 ..etc)

   {quote}

   [~ahussein] If the cause is in the tests, we should fix them or make them skipped if it takes time. If the cause is ISA-L feature itself, we should disable it first then fix or drop it.

   I'm going to look into relevant OPEN issues under HDFS-15646.

5. **ahussein:** {quote}Ahmed Hussein If the cause is in the tests, we should fix them or make them skipped if it takes time. If the cause is ISA-L feature itself, we should disable it first then fix or drop it.

   I'm going to look into relevant OPEN issues under HDFS-15646.{quote}

   Thanks [~iwasakims]. I certainly agree with you that the tests need to be fixed.

   This change adds native libraries and it looks plausible that those JUnits in question are affected by loading the native library.

   [~ayushtkn] and [~aajisaka] were able to narrow the reason of the failure (see [comment|https://issues.apache.org/jira
   15643?focusedCommentId=17223204&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-
   tabpanel#comment-17223204]).

   While this was a bug in the implementation, it is still does not seem right that those debugging are done after merging HADOOP-17224 .

   If [~ayushtkn] and [~aajisaka] were not aware of that change, I would not figure out the problem on my own.

   The closed jiras HDFS-15654, HDFS-15461 need to be re-investigated after the fix done in HDFS-15643.

   Also, Yetus fails when the JVM fails to allocate memory to create new threads. It is unclear so far when that error was introduced in Hadoop. Yet, I can imagine that calling a new native library could lead to different memory behavior.

   For Example, there could be a bug in the native code that leads to a memory leak, or the library reserves buffer that are not released back.

   If you think HADOOP-17224 is unlikely to trigger the failures of Junits, then do you have suggestions what could be the code commit that change the behavior of those units and trigger OOM on Yetus?

6. **tasanuma:** bq. The closed jiras HDFS-15654, HDFS-15461 need to be re-investigated after the fix done in HDFS-15643.

Aren't they covered by the last jenkins result of HDFS-15643? Sorry if I'm missing something.

About the OOM error, I think it happened occasionally before that. HDFS-12711 is a similar issue and closed as incomplete. I'm not sure if the ISA-L has made that more likely to happen. We could revert HADOOP-17224 once and see the qbt results for a while.

7. **ahussein:** {quote}Aren't they covered by the last jenkins result of HDFS-15643? Sorry if I'm missing something.{quote}

   Hey [~tasanuma]. I put a patch to fix HDFS-15654, HDFS-15461 before we learned about the bug in HDFS-15643 .

   Once HDFS-15643 got merged new errors started to show up in {{TestDFSClientRetries}} and {{TestBPOfferService}} (Example see [this comment|https://issues.apache.org/jira/browse/HDFS-15654?focusedCommentId=17226334&page=com.atlassian.jira.plugin.system.issuetabpanels:comment-tabpanel#comment-17226334])

   BTW, the reason I point to HADOOP-17224 to be the trigger of those failures is that the errors can only be reproduced with native flag. (See [Akira's comment on GitHub|https://github.com/apache/hadoop/

8. **kihwal:** [~tasanuma], If there are many test failures caused after the commit of this feature, it might be better to revert and rework, unless there is a simple change that will fix most failures quickly. It does not matter which part is to blame. It could be faulty assumptions or designs in existing tests or unforeseen negative side-effect of the feature. The point is, trunk build is in a broken state and more commits continue to come in with the justification like "I didn't break it. It was broken before".This is a situation we really want to avoid.

   If the test failures cannot be fixed quickly, we need to bring it back to the sane state first. Again, this has nothing to do with who is technically right or wrong. Please review the test failures and determine whether they can be fixed quickly. Feel free to ask if you need additional eyes.

9. **tasanuma:** [~ahussein] I understood. Thanks for your explanation.

   [~kihwal] I agree with you to revert and rework. It's not going to be easy to fix them.

   I will revert HADOOP-17224 and let's see how it goes.

10. **tasanuma:** Reverted it by[#2440|https://github.com/apache/hadoop/pull/2440].

11. **ahussein:** [~tasanuma] Thank you.

    Let's watch the qbt report after merging that change.

    So far, there is a noticeable decrease in failed tests in HDFS.

12. **iwasakims:** {quote}

    So far, there is a noticeable decrease in failed tests in HDFS.

    {quote}

    [~ahussein] Could you give me the name of relevant test cases? I would like good example for digging the effect of ISA-L.

    {quote}

    Once HDFS-15643 got merged new errors started to show up in TestDFSClientRetries and TestBPOfferService (Example see this comment)

    {quote}

    TestDFSClientRetries looks fixed by HDFS-15461. TestBPOfferService is still flaky.

Chapter 2

Connected issue HDFS-15643

2.1 Summary

EC: Fix checksum computation in case of native encoders

2.2 Description

There are many failures in {{TestFileChecksumCompositeCrc}}. The test cases {{testStripedFileChecksumWithMissedDataBlocksRangeQueryXX}} fail. The following is a sample of the stack trace in two of them Query7 and Query8.

```
org.apache.hadoop.fs.PathIOException: '/striped/stripedFileChecksum1': Fail to get block checksum for LocatedStripedBlock{BP-1812707539-172.17.0.3-1602771351154:blk_-9223372036854775792_1001; getBlockSize()=37748736; corrupt=false; offset=0; locs=[DatanodeInfoWithStorage[127.0.0.1:36687,DS-b00139f0-4f28-4870-8f72-b726bd339e23,DISK], DatanodeInfoWithStorage[127.0.0.1:36303,DS-49a3c58e-da4a-4256-b1f9-893e4003ec94,DISK], DatanodeInfoWithStorage[127.0.0.1:43975,DS-ac278858-b6c8-424f-9e20-58d718dabe31,DISK], DatanodeInfoWithStorage[127.0.0.1:37507,DS-17f9d8d8-f8d3-443b-8df7-29416a2f5cb0,DISK], DatanodeInfoWithStorage[127.0.0.1:36441,DS-7e9d19b5-6220-465f-b33e-f8ed0e60fb07,DISK], DatanodeInfoWithStorage[127.0.0.1:42555,DS-ce679f5e-19fe-45b0-a0cd-8d8bec2f4735,DISK], DatanodeInfoWithStorage[127.0.0.1:39093,DS-4a7f54bb-dd39-4b5b-8dee-31a1b565cd7f,DISK], DatanodeInfoWithStorage[127.0.0.1:41699,DS-e1f939f3-37e7-413e-a522-934243477d81,DISK]]; indices=[1, 2, 3, 4, 5, 6, 7, 8]}
    at org.apache.hadoop.hdfs.FileChecksumHelper$StripedFileNonStripedChecksumComputer.checksumBlocks(FileChecksumHelper.java:640)
    at org.apache.hadoop.hdfs.FileChecksumHelper$FileChecksumComputer.compute(FileChecksumHelper.java:252)
    at org.apache.hadoop.hdfs.DFSClient.getFileChecksumInternal(DFSClient.java:1851)
    at org.apache.hadoop.hdfs.DFSClient.getFileChecksumWithCombineMode(DFSClient.java:1871)
    at org.apache.hadoop.hdfs.DistributedFileSystem$34.doCall(DistributedFileSystem.java:1902)
    at org.apache.hadoop.hdfs.DistributedFileSystem$34.doCall(DistributedFileSystem.java:1899)

    at org.apache.hadoop.fs.FileSystemLinkResolver.resolve(FileSystemLinkResolver.java:81)
    at org.apache.hadoop.hdfs.DistributedFileSystem.getFileChecksum(DistributedFileSystem.java:1916)
    at org.apache.hadoop.hdfs.TestFileChecksum.getFileChecksum(TestFileChecksum.java:584)
```

```
    at org.apache.hadoop.hdfs.TestFileChecksum.
        testStripedFileChecksumWithMissedDataBlocksRangeQuery(TestFileChecksum.java:295)
    a
t org.apache.hadoop.hdfs.TestFileChecksum.
    testStripedFileChecksumWithMissedDataBlocksRangeQuery7(TestFileChecksum.java:377)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java
        :43)
    at java.lang.reflect.Method.invoke(Meth
od.java:498)
    at org.junit.runners.model.FrameworkMethod$1.runReflectiveCall(FrameworkMethod.java
        :50)
    at org.junit.internal.runners.model.ReflectiveCallable.run(ReflectiveCallable.java
        :12)
    at org.junit.runners.model.FrameworkMethod.invokeExplosively(FrameworkMethod.java
        :47)
    at org.junit.internal.runners.statements.InvokeMethod.evaluate(InvokeMethod.java:17)
    at org.junit.internal.runners.statem
ents.FailOnTimeout$CallableStatement.call(FailOnTimeout.java:298)
    at org.junit.internal.runners.statements.FailOnTimeout$CallableStatement.call(
        FailOnTimeout.java:292)
    at java.util.concurrent.FutureTask.run(FutureTask.java:266)
    at java.lang.Thread.run(Thread.java:748)
```

```
Error Message
'/striped/stripedFileChecksum1': Fail to get block checksum for LocatedStripedBlock{BP
    -1299291876-172.17.0.3-1602771356932:blk_-9223372036854775792_1001; getBlockSize()
    =37748736; corrupt=false; offset=0; locs=[DatanodeInfoWithStorage[127.0.0.1:42217,DS
    -6c29e4b7-e4f1-4302-ad23-fb078f37d783,DISK], DatanodeInfoWithStorage
    [127.0.0.1:41307,DS-3d824f14-3cd
0-46b1-bef1-caa808bf278d,DISK], DatanodeInfoWithStorage[127.0.0.1:37193,DS-eeb44ff5-fdf1
    -4774-b6cf-5be7c40147a9,DISK], DatanodeInfoWithStorage[127.0.0.1:39897,DS-36d2fbfc
    -64bc-405c-8360-735f1ad92e30,DISK], DatanodeInfoWithStorage[127.0.0.1:35545,DS-6
    fd42817-efea-416e-92fb-3e9034705142,DISK], DatanodeInfoWithStorage[127.0.0.1:39945,
    DS-501deff8-b6df-4cf0-9ac1-154a4253eec8,DISK], DatanodeInfoWithStor
age[127.0.0.1:41359,DS-9b0449f5-377b-4a76-9eb6-0bcf2984b4bb,DISK],
    DatanodeInfoWithStorage[127.0.0.1:36123,DS-4184ab4a-079d-4b1c-a8cb-2ba22b0baafb,DISK
    ]]; indices=[0, 1, 2, 3, 4, 6, 7, 8]}
Stacktrace
org.apache.hadoop.fs.PathIOException: '/striped/stripedFileChecksum1': Fail to get block
     checksum for LocatedStripedBlock{BP-1299291876-172.17.0.3-1602771356932:blk_
    -9223372036854775792_1001; getBlock
Size()=37748736; corrupt=false; offset=0; locs=[DatanodeInfoWithStorage[127.0.0.1:42217,
    DS-6c29e4b7-e4f1-4302-ad23-fb078f37d783,DISK], DatanodeInfoWithStorage
    [127.0.0.1:41307,DS-3d824f14-3cd0-46b1-bef1-caa808bf278d,DISK],
    DatanodeInfoWithStorage[127.0.0.1:37193,DS-eeb44ff5-fdf1-4774-b6cf-5be7c40147a9,DISK
    ], DatanodeInfoWithStorage[127.0.0.1:39897,DS-36d2fbfc-64bc-405c-8360-735f1ad92e30,
    DISK], Data
nodeInfoWithStorage[127.0.0.1:35545,DS-6fd42817-efea-416e-92fb-3e9034705142,DISK],
    DatanodeInfoWithStorage[127.0.0.1:39945,DS-501deff8-b6df-4cf0-9ac1-154a4253eec8,DISK
    ], DatanodeInfoWithStorage[127.0.0.1:41359,DS-9b0449f5-377b-4a76-9eb6-0bcf2984b4bb,
    DISK], DatanodeInfoWithStorage[127.0.0.1:36123,DS-4184ab4a-079d-4b1c-a8cb-2
    ba22b0baafb,DISK]]; indices=[0, 1, 2, 3, 4, 6, 7, 8]}
    at org.apache.hadoop
.hdfs.FileChecksumHelper$StripedFileNonStripedChecksumComputer.checksumBlocks(
    FileChecksumHelper.java:640)
    at org.apache.hadoop.hdfs.FileChecksumHelper$FileChecksumComputer.compute(
```

```
        FileChecksumHelper.java:252)
    at org.apache.hadoop.hdfs.DFSClient.getFileChecksumInternal(DFSClient.java:1851)
    at org.apache.hadoop.hdfs.DFSClient.getFileChecksumWithCombineMode(DFSClient.java
        :1871)
    at org.apache.had
oop.hdfs.DistributedFileSystem$34.doCall(DistributedFileSystem.java:1902)
    at org.apache.hadoop.hdfs.DistributedFileSystem$34.doCall(DistributedFileSystem.java
        :1899)
    at org.apache.hadoop.fs.FileSystemLinkResolver.resolve(FileSystemLinkResolver.java
        :81)
    at org.apache.hadoop.hdfs.DistributedFileSystem.getFileChecksum(
        DistributedFileSystem.java:1916)
    at org.apache.hadoop.hdfs.TestFileChecksum.getF
ileChecksum(TestFileChecksum.java:584)
    at org.apache.hadoop.hdfs.TestFileChecksum.
        testStripedFileChecksumWithMissedDataBlocksRangeQuery(TestFileChecksum.java:295)
    at org.apache.hadoop.hdfs.TestFileChecksum.
        testStripedFileChecksumWithMissedDataBlocksRangeQuery8(TestFileChecksum.java
        :388)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invok
e(NativeMethodAccessorImpl.java:62)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java
        :43)
    at java.lang.reflect.Method.invoke(Method.java:498)
    at org.junit.runners.model.FrameworkMethod$1.runReflectiveCall(FrameworkMethod.java
        :50)
    at org.junit.internal.runners.model.ReflectiveCallable.run(ReflectiveCallable.java
        :12)
    at org.junit.runners.model.FrameworkMethod.
invokeExplosively(FrameworkMethod.java:47)
    at org.junit.internal.runners.statements.InvokeMethod.evaluate(InvokeMethod.java:17)
    at org.junit.internal.runners.statements.FailOnTimeout$CallableStatement.call(
        FailOnTimeout.java:298)
    at org.junit.internal.runners.statements.FailOnTimeout$CallableStatement.call(
        FailOnTimeout.java:292)
    at java.util.concurrent.FutureTask.run(FutureTask.java:266)
    at
java.lang.Thread.run(Thread.java:748)
```

2.3 Attachments

1. HDFS-15643-01.patch
2. org.apache.hadoop.hdfs.TestFileChecksum.txt
3. org.apache.hadoop.hdfs.TestFileChecksum-output.txt
4. TestFileChecksum.testStripedFileChecksumWithMissedDataBlocksRangeQuery17.log
5. Test-Fix-01.patch

2.4 Comments

1. **ahussein:** Looking at the logs, my best guess is that there is a race when the unit-test shuts down a datanode and trying to access the file.

It seems that in a slow execution, the threads will race and there is no enough time to reconstruct the blocks following a shutdown.

The fix also applies for {{TestFileChecksum}}.

2. **aajisaka:**

Error log:

```
2020-10-29 04:55:00,418 [DataXceiver for client /127.0.0.1:60824 [Getting
checksum for block groupBP-406455512-192.168.249.5-1603947295509:blk_-9223372036854
ERROR datanode.DataNode (DataXceiver.java:run(3

24)) - 127.0.0.1:36907:DataXceiver error processing BLOCK_GROUP_CHECKSUM
operation src: /127.0.0.1:60824 dst: /127.0.0.1:36907

java.lang.UnsupportedOperationException

    at java.nio.ByteBuffer.array(ByteBuffer.java:994)

    at org.apache.hadoop.hdfs.server.datanode.erasurecode.StripedBlockChecksumRecon

    at org.apache.hadoop.hdfs.server.datanode.BlockChecksumHelper$BlockGroupNonStri

    at org.apache.hadoop.hdfs.server.datanode.BlockChecksumHelper$BlockGroupNonStri

    at org.apache.hadoop.hdfs.server.datanode.DataXceiver.blockGroupChecksum(DataXc

    at org.apache.hadoop.hdfs.protocol.datatransfer.Receiver.opStripedBlockChecksum

    at org.apache.hadoop.hdfs.protocol.datatransfer.Receiver.processOp(Receiver.jav

    at org.apache.hadoop.hdfs.server.datanode.DataXceiver.run(DataXceiver.java:292)

    at java.lang.Thread.run(Thread.java:748)

2020-10-29 04:55:00,418 [Time-limited test] WARN hdfs.FileChecksumHelper
(FileChecksumHelper.java:checksumBlockGroup(679)) - src=/striped/stripedFileChecksu
datanodes[1]=DatanodeInfoWithStorage[127.0.0.1:36907,DS-e9365b1c-9803-4d66-ac04-9aa

java.io.EOFException: Unexpected EOF while trying to read response from
server

    at org.apache.hadoop.hdfs.protocolPB.PBHelperClient.vintPrefixed(PBHelperClient

    at org.apache.hadoop.hdfs.FileChecksumHelper$StripedFileNonStripedChecksumComput

    at org.apache.hadoop.hdfs.FileChecksumHelper$StripedFileNonStripedChecksumComput
```

```
at org.apache.hadoop.hdfs.FileChecksumHelper$StripedFileNonStripedChecksumComput
at org.apache.hadoop.hdfs.FileChecksumHelper$FileChecksumComputer.compute(FileC
at org.apache.hadoop.hdfs.DFSClient.getFileChecksumInternal(DFSClient.java:1851
at org.apache.hadoop.hdfs.DFSClient.getFileChecksumWithCombineMode(DFSClient.ja
at org.apache.hadoop.hdfs.DistributedFileSystem$34.doCall(DistributedFileSystem
at org.apache.hadoop.hdfs.DistributedFileSystem$34.doCall(DistributedFileSystem
at org.apache.hadoop.fs.FileSystemLinkResolver.resolve(FileSystemLinkResolver.j
at org.apache.hadoop.hdfs.DistributedFileSystem.getFileChecksum(DistributedFile
at org.apache.hadoop.hdfs.TestFileChecksum.getFileChecksum(TestFileChecksum.jav
at org.apache.hadoop.hdfs.TestFileChecksum.testStripedFileChecksumWithMissedDat
at org.apache.hadoop.hdfs.TestFileChecksum.testStripedFileChecksumWithMissedDat
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl
at java.lang.reflect.Method.invoke(Method.java:498)
at org.junit.runners.model.FrameworkMethod$1.runReflectiveCall(FrameworkMethod.
at org.junit.internal.runners.model.ReflectiveCallable.run(ReflectiveCallable.j
at org.junit.runners.model.FrameworkMethod.invokeExplosively(FrameworkMethod.ja
at org.junit.internal.runners.statements.InvokeMethod.evaluate(InvokeMethod.jav
at org.junit.internal.runners.statements.FailOnTimeout$CallableStatement.call(F
at org.junit.internal.runners.statements.FailOnTimeout$CallableStatement.call(F
at java.util.concurrent.FutureTask.run(FutureTask.java:266)
at java.lang.Thread.run(Thread.java:748)
```

3. **ahussein:** Thank you [~aajisaka], I could not reproduce it on my local environment.

Anyway, the exception is thrown after the unit test shuts down a datanode.

The following {{getFileChecksum()}} seems to construct the checksum before the block

reconstruction is complete.

Because reconstruction is not complete, the {{StripedBlockChecksumReconstructor}} uses the wrong buffer type,

which leads to calling {{toArray()}} on a {{DirectBuffer}}.

I remember that I pointed out to similar issue with EC and stripped group blocks in HDFS-15459.

Regarding this unit test, I think that my early suggestion to wait until reconstruction is complete needs to be added to the test.

However, I prefer that someone familiar with EC takes a look at the implementation because this is most probably a major bug

in the code of the EC and striped files.

[~inigoiri], [~aajisaka]. Thank you guys for your help. Unfortunately, I won't have much time in the next couple of weeks to dig

into the EC and striped files implementation.

[~weichiu] , who is familiar with EC and can help with looking at those bugs? (this one and HDFS-15459)

Until then, I suggest we turn off both {{TestFileChecksumCompositeCrc}} and {{TestFileChecksum}} at least to give us a chance

to save time and get more accurate reports excluding the side effects of those two test classes.

4. **ayushtkn:** [~aajisaka] can you try the patch. I think there is an issue with the code itself, In case of Native Encoders, In case EC isn't using the Native Encoding, this seems to be good. Well, It didn't directly repro for me also, I just tried to repro this by some code tweaks, And tried just one test in TestFileChecksum.
5. **ahussein:** [~ayushtkn], do you think this is the root of the problem or the symptoms?

That code in \{{StripedBlockChecksumReconstructor}} is pretty old.

6. **ayushtkn:** Yeps, That is quite old, but in EC, if you don't have the native loaded, it will silently ignore and go ahead with the non-native encoders. So, I think this issue was there already but the native wasn't loaded during test.

I suspect it may be because of HADOOP-17224

7. **ahussein:** Thanks [~ayushtkn], I probably messed HADOOP-17224

   I remember I this unit test used to show up earlier than August; but anyway, I hope that this will be the fix.

   Yetus was recently running Out of memory and all submitted patches did not run.You can go ahead

   with creating a new PR with your changes. Then if the test case passes, we can close the current opened one.

8. **aajisaka:** bq. Akira Ajisaka can you try the patch.

   I tried the patch in https://github.com/apache/hadoop/pull/2421/files but it failed.

   The following test cases failed in TestFileChecksum:

```
[ERROR] Errors:

[ERROR]   TestFileChecksum.testStripedFileChecksumWithMissedDataBlocks1:279->getFil
? PathIO

[ERROR]   TestFileChecksum.testStripedFileChecksumWithMissedDataBlocks2:296->getFil
? PathIO

[ERROR]   TestFileChecksum.testStripedFileChecksumWithMissedDataBlocksRangeQuery12:
? PathIO

[ERROR]   TestFileChecksum.testStripedFileChecksumWithMissedDataBlocksRangeQuery13:
? PathIO

[ERROR]   TestFileChecksum.testStripedFileChecksumWithMissedDataBlocksRangeQuery14:
? PathIO

[ERROR]   TestFileChecksum.testStripedFileChecksumWithMissedDataBlocksRangeQuery15:
```

```
? PathIO

[ERROR]   TestFileChecksum.testStripedFileChecksumWithMissedDataBlocksRangeQuery17:
? PathIO

[ERROR]   TestFileChecksum.testStripedFileChecksumWithMissedDataBlocksRangeQuery19:
? PathIO

[ERROR]   TestFileChecksum.testStripedFileChecksumWithMissedDataBlocksRangeQuery2:3
? PathIO

[ERROR]   TestFileChecksum.testStripedFileChecksumWithMissedDataBlocksRangeQuery20:
? PathIO

[ERROR]   TestFileChecksum.testStripedFileChecksumWithMissedDataBlocksRangeQuery8:4
? PathIO

[ERROR]   TestFileChecksum.testStripedFileChecksumWithMissedDataBlocksRangeQuery9:4
? PathIO
```

9. **ayushtkn:** {quote}I tried the patch in[https://github.com/apache/hadoop/pull/2421/files]but it failed.

   {quote}

   Thanx [~aajisaka]for trying out, do you mean to say you tried locally applying the patch along with the above PR? Since my changes aren't there in the PR.

   I triggered jenkins for the patch [above|https://issues.apache.org/jira/browse/HDFS-15643?focusedCommentId=17
   tabpanel#comment-17223295] , and the tests passed :

   [https://ci-hadoop.apache.org/job/PreCommit-HDFS-Build/276/testReport/org.apache.hadoop.hdfs/TestFileChec

   [https://ci-hadoop.apache.org/job/PreCommit-HDFS-Build/276/testReport/org.apache.hadoop.hdfs/TestFileChec

10. **aajisaka:** Thank you [~ayushtkn]for pointing out. I'll tryhttps://issues.apache.org/jira/secure/attachment/130143
    Fix-01.patch

11. **aajisaka:** I tried the above patch and the test passed in my docker environment. Thank you [~ayushtkn]

12. **aajisaka:** The patch copies the data in memory three times if \{{requestedLen <= toReconstructLen}}:

* L90 -> L216: Copy from direct byte buffer from bytes[] viagetBufferArray()

* L143 -> L216: Copy from direct byte buffer from bytes[] via getBufferArray()

* L143: Copy from bytes[] to bytes[] via Arrays.copyOf()

The copy operations may be reduced.

13. **ayushtkn:** A simple change could be at L143 :

```
1
2 -   outputData = Arrays.copyOf(targetBuffer.array(), remainingLen);
3
4 +   outputData = Arrays.copyOf(outputData, remainingLen);
```

So, the number should get reduced to 2 for native Encoders, for the non native it would be still 1 only.

To reduce one more copy as well, can refactor the method something like this :

```
diff --git a/hadoop-hdfs-project/hadoop-hdfs/src/main/java/org/apache/hadoop/hdfs/s
b/hadoop-hdfs-project/hadoop-hdfs/src/main/java/org/apache/hadoop/hdfs/server/datan

index b2e64966a18..848d8dee654 100644

--- a/hadoop-hdfs-project/hadoop-hdfs/src/main/java/org/apache/hadoop/hdfs/server/d

+++ b/hadoop-hdfs-project/hadoop-hdfs/src/main/java/org/apache/hadoop/hdfs/server/d

@@ -86,8 +86,7 @@ public void reconstruct() throws IOException {

     reconstructTargets(toReconstructLen);



     // step3: calculate checksum
```

```
-    checksumDataLen += checksumWithTargetOutput(
-      targetBuffer.array(), toReconstructLen);
+    checksumDataLen += checksumWithTargetOutput(toReconstructLen);


    updatePositionInBlock(toReconstructLen);
    requestedLen -= toReconstructLen;
@@ -132,7 +131,7 @@ protected DataOutputBuffer getChecksumWriter() {
  return checksumWriter;
 }


- private long checksumWithTargetOutput(byte[] outputData, int toReconstructLen)
+ private long checksumWithTargetOutput(int toReconstructLen)
   throws IOException {
  long checksumDataLength = 0;
  // Calculate partial block checksum. There are two cases.
@@ -140,7 +139,7 @@ private long checksumWithTargetOutput(byte[] outputData,
int toReconstructLen)
  // case-2) length of data bytes which is less than bytesPerCRC
  if (requestedLen <= toReconstructLen) {
   int remainingLen = Math.toIntExact(requestedLen);
-   outputData = Arrays.copyOf(targetBuffer.array(), remainingLen);
+   byte[] outputData = getBufferArray(targetBuffer, remainingLen);


   int partialLength = remainingLen % getChecksum().getBytesPerChecksum();
```

```
@@ -175,6 +174,7 @@ private long checksumWithTargetOutput(byte[] outputData,
int toReconstructLen)

   // calculated checksum for the requested length, return checksum
length.

   return checksumDataLength;

  }

+  byte[] outputData = getBufferArray(targetBuffer, targetBuffer.remaining());

  getChecksum().calculateChunkedSums(outputData, 0,

    outputData.length, checksumBuf, 0);




@@ -207,4 +207,18 @@ private void clearBuffers() {

 public long getChecksumDataLen() {

  return checksumDataLen;

 }

-}

+

+ public static byte[] getBufferArray(ByteBuffer targetBuffer,

+   int remainingLen) {

+  byte[] buff = new byte[remainingLen];

+  if (targetBuffer.hasArray()) {

+   buff = targetBuffer.array();

+   if (remainingLen != targetBuffer.remaining()) {

+    Arrays.copyOf(buff, remainingLen);

+   }

+  } else {

+   targetBuffer.slice().get(buff);
```

```
+   }

+   return buff;
```

But might be we can go with the first approach, seems little safe, in case of EC if the index gets mixed up in any way, the data gets corrupted. Let me know your thoughts, Will update accordingly

14. **aajisaka:** {quote}But might be we can go with the first approach, seems little safe, in case of EC if the index gets mixed up in any way, the data gets corrupted.

    {quote}

    Agreed. Let's refactor in a separate jira.

15. **aajisaka:** [~touchida]told me HDFS-15237 is related to this issue. Link HDFS-15237

16. **ayushtkn:** Thanx [~aajisaka], I made the change to reduce one copy operation as suggested by you here :

    https://github.com/apache/hadoop/pull/2424

17. **ahussein:** {quote}I tried the patch in https://github.com/apache/hadoop/pull/2421/files but it failed.{quote}

    I have a little bit bad feeling about that. I suspect there is something else somewhere that needs fixing.

18. **ahussein:** Once, this fix gets merged into trunk, I will close the existing PRs I created. Then open a new one that refactors that Junit test

    and improve the cleaning inside "@After" (HDFS-15648)

19. **ayushtkn:** Though the build failed, but the test passed. :P

    [https://ci-hadoop.apache.org/job/PreCommit-HDFS-Build/282/testReport/org.apache.hadoop.hdfs/TestFileChe

20. **ahussein:** Thank you [~ayushtkn] for the progress done on this jira.

    Is there any thing can be done for all those OOM on Yetus?

    I sent an email on hadoop mailing list, but this problem is there for so long.

21. **ahussein:** Thank you [~ayushtkn] for the progress done on that Jira.

    Is there something that can be done for the OOM errors on Yetus? It has been there for few months.

    I sent an email on the mailing list sometime ago to give heads-up.

    I wonder if there is a recent commit that caused leak in native memory.

22. **ayushtkn:** Committed to trunk, branch-3.3,3.2,3.1

    Thanx Everyone!!!

23. **ayushtkn:** {quote}Is there something that can be done for the OOM errors on Yetus? It has been there for few months.

    {quote}

    If we are sure it isn't because of some leak in native memory, then we can raise an INFRA Jira and ask the folks to figure out things.

    But I don't deny the fact, it can be due to some faulty test causing issue, IIRC couple of months back there was a Jira which led to such a situation.

24. **hexiaoqiao:** Thanks [~ayushtkn] and [~ahussein] for your works. Backport to branch-3.2.2.