TAJO-544

coderplay

2014-01-22

Contents

Chapter 1

Root issue TAJO-544

1.1 Summary

Thread pool abusing

1.2 Description

After TAJO-522, TAJO-537, I still came across the "can't create native thread" OOM. I will submit a stack trace which tracking all the source of creating new threads.

1.3 Attachments

1. fetcher.patch
2. thread.txt

1.4 Comments

1. **coderplay:** From the result of I have 3 suspicions.

   # Take a look at Fetcher.java. Each data fetching create a threadpool and this thread pool live during the whole time of data fetching.

   # QueryMasterTask.init, each QueryMasterTask create a threadpool by instancing a brand new TajoResourceAllocator. If there are concurrent queries, there should be several thread pools.

   # ugly implementation of hadoop ipc, each connection start a thread

   ```
   org.apache.hadoop.ipc.Client$Connection.setupIOstreams(Client.java:704)
   org.apache.hadoop.ipc.Client$Connection.access$2600(Client.java:314)
   org.apache.hadoop.ipc.Client.getConnection(Client.java:1399)
   org.apache.hadoop.ipc.Client.call(Client.java:1318)
   org.apache.hadoop.ipc.Client.call(Client.java:1300)
   org.apache.hadoop.ipc.ProtobufRpcEngine$Invoker.invoke(ProtobufRpcEngine.java:206)
   $Proxy9.getBlockLocations(Unknown Source)
   sun.reflect.GeneratedMethodAccessor18.invoke(Unknown Source)
   ```

```
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:2
java.lang.reflect.Method.invoke(Method.java:597)
org.apache.hadoop.io.retry.RetryInvocationHandler.invokeMethod(RetryInvocationHandl
org.apache.hadoop.io.retry.RetryInvocationHandler.invoke(RetryInvocationHandler.jav
$Proxy9.getBlockLocations(Unknown Source)
org.apache.hadoop.hdfs.protocolPB.ClientNamenodeProtocolTranslatorPB.getBlockLocati
org.apache.hadoop.hdfs.DFSClient.callGetBlockLocations(DFSClient.java:1064)
org.apache.hadoop.hdfs.DFSClient.getLocatedBlocks(DFSClient.java:1054)
org.apache.hadoop.hdfs.DFSClient.getLocatedBlocks(DFSClient.java:1044)
org.apache.hadoop.hdfs.DFSInputStream.fetchLocatedBlocksAndGetLastBlockLength(DFSIn
org.apache.hadoop.hdfs.DFSInputStream.openInfo(DFSInputStream.java:202)
org.apache.hadoop.hdfs.DFSInputStream.<init>(DFSInputStream.java:195)
org.apache.hadoop.hdfs.DFSClient.open(DFSClient.java:1212)
org.apache.hadoop.hdfs.DistributedFileSystem$3.doCall(DistributedFileSystem.java:29
org.apache.hadoop.hdfs.DistributedFileSystem$3.doCall(DistributedFileSystem.java:28
org.apache.hadoop.fs.FileSystemLinkResolver.resolve(FileSystemLinkResolver.java:81)
org.apache.hadoop.hdfs.DistributedFileSystem.open(DistributedFileSystem.java:286)
org.apache.hadoop.fs.FileSystem.open(FileSystem.java:763)
org.apache.tajo.storage.CSVFile$CSVScanner.init(CSVFile.java:292)
org.apache.tajo.engine.planner.physical.SeqScanExec.init(SeqScanExec.java:165)
org.apache.tajo.engine.planner.physical.UnaryPhysicalExec.init(UnaryPhysicalExec.ja
org.apache.tajo.engine.planner.physical.UnaryPhysicalExec.init(UnaryPhysicalExec.ja
org.apache.tajo.engine.planner.physical.HashShuffleFileWriteExec.init(HashShuffleFi
org.apache.tajo.worker.Task.run(Task.java:369)
org.apache.tajo.worker.TaskRunner$1.run(TaskRunner.java:392)
java.lang.Thread.run(Thread.java:662)
```

But this thread live for a short time.

2. **coderplay:** AS TAJO-537, sharing factory among all of the fetchers.
3. **hyunsik:** +1 for the patch, and it's a nice investigation!

   As you pointed out, we have abused ThreadFactory. This is the first point that we can easily try to improve. I've tested it on a real cluster. It works fine.
4. **hyunsik:** committed it to master branch. Thank you for nice contribution.
5. **hudson:** ABORTED: Integrated in Tajo-master-build #28 (See [https://builds.apache.org/job/Tajo-master-build/28/])
   TAJO-544: Thread pool abusing. (Min Zhou via hyunsik) (hyunsik: https://git-wip-us.apache.org/repos/asf?p=inc
   tajo.git&a=commit&h=3125733cc6f2bbce306afd00b8fa8440f07b6e1e)
   * tajo-core/tajo-core-backend/src/main/java/org/apache/tajo/worker/Fetcher.java
   TAJO-544: Thread pool abusing. (add a missed change log) (hyunsik: https://git-wip-us.apache.org/repos/asf?p=incubator-tajo.git&a=commit&h=a74e52188b82b2c753846ddabd9f74bfe0e94e2f)
   * CHANGES.txt

Chapter 2

Connected issue TAJO-522

2.1 Summary

OutOfMemoryError: unable to create new native thread

2.2 Description

Another exception occurs on the client side when I run a query

```
Progress: 0%, response time: 190.197 sec
com.google.protobuf.ServiceException: java.lang.OutOfMemoryError: unable
to create new native thread
at org.apache.tajo.client.TajoClient.getQueryStatus(TajoClient.java:193)
at org.apache.tajo.cli.TajoCli.getQueryResult(TajoCli.java:353)
at org.apache.tajo.cli.TajoCli.executeStatements(TajoCli.java:319)
at org.apache.tajo.cli.TajoCli.runShell(TajoCli.java:228)
at org.apache.tajo.cli.TajoCli.main(TajoCli.java:735)
Caused by: java.io.IOException: java.lang.OutOfMemoryError: unable to create
new native thread
at org.apache.tajo.rpc.NettyClientBase.init(NettyClientBase.java:76)
at org.apache.tajo.rpc.BlockingRpcClient.<init>(BlockingRpcClient.java:71)
at org.apache.tajo.rpc.RpcConnectionPool.makeConnection(RpcConnectionPool.java:54)
at org.apache.tajo.rpc.RpcConnectionPool.getConnection(RpcConnectionPool.java:63)
at org.apache.tajo.client.TajoClient.getQueryStatus(TajoClient.java:188)
... 4 more
Caused by: java.lang.OutOfMemoryError: unable to create new native thread
at java.lang.Thread.start0(Native Method)
at java.lang.Thread.start(Thread.java:640)
at java.util.concurrent.ThreadPoolExecutor.addIfUnderMaximumPoolSize(ThreadPoolExecutor
at java.util.concurrent.ThreadPoolExecutor.execute(ThreadPoolExecutor.java:657)
at org.jboss.netty.util.internal.DeadLockProofWorker.start(DeadLockProofWorker.java:38)
at org.jboss.netty.channel.socket.nio.AbstractNioSelector.openSelector(AbstractNioSelect
at org.jboss.netty.channel.socket.nio.AbstractNioSelector.<init>(AbstractNioSelector.jav
at org.jboss.netty.channel.socket.nio.AbstractNioWorker.<init>(AbstractNioWorker.java:5
at org.jboss.netty.channel.socket.nio.NioWorker.<init>(NioWorker.java:45)
at org.jboss.netty.channel.socket.nio.NioWorkerPool.createWorker(NioWorkerPool.java:45)
at org.jboss.netty.channel.socket.nio.NioWorkerPool.createWorker(NioWorkerPool.java:28)
```

```
at org.jboss.netty.channel.socket.nio.AbstractNioWorkerPool.newWorker(AbstractNioWorkerP
at org.jboss.netty.channel.socket.nio.AbstractNioWorkerPool.init(AbstractNioWorkerPool.j
at org.jboss.netty.channel.socket.nio.NioWorkerPool.<init>(NioWorkerPool.java:39)
at org.jboss.netty.channel.socket.nio.NioWorkerPool.<init>(NioWorkerPool.java:33)
at org.jboss.netty.channel.socket.nio.NioClientSocketChannelFactory.<init>(NioClientSocke
at org.jboss.netty.channel.socket.nio.NioClientSocketChannelFactory.<init>(NioClientSocke
at org.apache.tajo.rpc.NettyClientBase.init(NettyClientBase.java:54)
... 8 more
java.lang.OutOfMemoryError: unable to create new native thread
2014-01-19 23:12:19,975 WARN  client.TajoClient (TajoClient.java:closeQuery(110))
- Fail to close a QueryMaster connection (qid=q_1390100273039_0012, msg=java.lang.OutOfM
unable to create new native thread)
java.io.IOException: java.lang.OutOfMemoryError: unable to create new native
thread
at org.apache.tajo.rpc.NettyClientBase.init(NettyClientBase.java:76)
at org.apache.tajo.rpc.BlockingRpcClient.<init>(BlockingRpcClient.java:71)
at org.apache.tajo.rpc.RpcConnectionPool.makeConnection(RpcConnectionPool.java:54)
at org.apache.tajo.rpc.RpcConnectionPool.getConnection(RpcConnectionPool.java:63)
at org.apache.tajo.client.TajoClient.closeQuery(TajoClient.java:106)
at org.apache.tajo.cli.TajoCli.executeStatements(TajoCli.java:323)
at org.apache.tajo.cli.TajoCli.runShell(TajoCli.java:228)
at org.apache.tajo.cli.TajoCli.main(TajoCli.java:735)
Caused by: java.lang.OutOfMemoryError: unable to create new native thread
at java.lang.Thread.start0(Native Method)
at java.lang.Thread.start(Thread.java:640)
at java.util.concurrent.ThreadPoolExecutor.addIfUnderMaximumPoolSize(ThreadPoolExecutor
at java.util.concurrent.ThreadPoolExecutor.execute(ThreadPoolExecutor.java:657)
at org.jboss.netty.util.internal.DeadLockProofWorker.start(DeadLockProofWorker.java:38)
at org.jboss.netty.channel.socket.nio.AbstractNioSelector.openSelector(AbstractNioSelect
at org.jboss.netty.channel.socket.nio.AbstractNioSelector.<init>(AbstractNioSelector.jav
at org.jboss.netty.channel.socket.nio.AbstractNioWorker.<init>(AbstractNioWorker.java:53
at org.jboss.netty.channel.socket.nio.NioWorker.<init>(NioWorker.java:45)
at org.jboss.netty.channel.socket.nio.NioWorkerPool.createWorker(NioWorkerPool.java:45)
at org.jboss.netty.channel.socket.nio.NioWorkerPool.createWorker(NioWorkerPool.java:28)
at org.jboss.netty.channel.socket.nio.AbstractNioWorkerPool.newWorker(AbstractNioWorkerP
at org.jboss.netty.channel.socket.nio.AbstractNioWorkerPool.init(AbstractNioWorkerPool.j
at org.jboss.netty.channel.socket.nio.NioWorkerPool.<init>(NioWorkerPool.java:39)
at org.jboss.netty.channel.socket.nio.NioWorkerPool.<init>(NioWorkerPool.java:33)
at org.jboss.netty.channel.socket.nio.NioClientSocketChannelFactory.<init>(NioClientSocke
at org.jboss.netty.channel.socket.nio.NioClientSocketChannelFactory.<init>(NioClientSocke
at org.apache.tajo.rpc.NettyClientBase.init(NettyClientBase.java:54)
... 7 more
```

I am not sure why client side need to create quite a lot of threads.

2.3 Attachments

1. TAJO-522.patch
2. tajo-site.xml

2.4 Comments

1. **hyunsik:** Hi Min,

   Actually, I couldn't reproduce it in my environment. Could you share some of your environment information?
   'ulimit -a' and 'tajo.worker.resource.memory-mb' in tajo-site.xml would be helpful to find the cause of problems.

2. **coderplay:** This line is reported by tsql client, not the worker.

   Here is the result of ulimit -a

```
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 256476
max locked memory       (kbytes, -l) 64
max memory size         (kbytes, -m) unlimited
open files                      (-n) 65536
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority              (-r) 0
stack size              (kbytes, -s) 10240
cpu time               (seconds, -t) unlimited
max user processes              (-u) 10000
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
```

   I will submit my tajo-site.xml soon

3. **hyunsik:** Could you show me conf/tajo-env.sh?

   I wonder that you set a proper memory size to worker heap memory.
   http://tajo.incubator.apache.org/tajo-0.8.0-doc.html#TajoMasterHeap

4. **hyunsik:** This patch is moved from TAJO-525.

5. **hyunsik:** According to Min's reports in TAJO-525, we can expect that this problem is essentially caused by one bug that creates too many threads in TajoContainerProxy::assignExecutionBlock.

6. **tajoqa:** {color:red}*-1 overall.*{color} Here are the results of testing the latest attachment
   http://issues.apache.org/jira/secure/attachment/12623913/TAJO-522.patch
   against master revision d04f9a5.

   {color:green}+1 @author.{color} The patch does not contain any @author tags.

   {color:red}-1 tests included.{color} The patch doesn't appear to include any new or modified tests.
   Please justify why no new tests are needed for this patch.
   Also please list what manual steps were performed to verify this patch.

{color:green}+1 javac.{color} The applied patch does not increase the total number of javac compiler warnings.

{color:green}+1 javadoc.{color} The applied patch does not increase the total number of javadoc warnings.

{color:green}+1 checkstyle.{color} The patch generated 0 code style errors.

{color:red}-1 findbugs.{color} The patch appears to introduce 181 new Findbugs (version 1.3.9) warnings.

{color:green}+1 release audit.{color} The applied patch does not increase the total number of release audit warnings.

{color:green}+1 core tests.{color} The patch passed unit tests in tajo-common tajo-core/tajo-core-backend tajo-rpc.

Test results: https://builds.apache.org/job/PreCommit-TAJO-Build/65//testReport/
Findbugs warnings: https://builds.apache.org/job/PreCommit-TAJO-Build/65//artifact/incubator-tajo/patchprocess/newPatchFindbugsWarningstajo-common.html
Findbugs warnings: https://builds.apache.org/job/PreCommit-TAJO-Build/65//artifact/incubator-tajo/patchprocess/newPatchFindbugsWarningstajo-rpc.html
Findbugs warnings: https://builds.apache.org/job/PreCommit-TAJO-Build/65//artifact/incubator-tajo/patchprocess/newPatchFindbugsWarningstajo-core-backend.html
Console output: https://builds.apache.org/job/PreCommit-TAJO-Build/65//console

This message is automatically generated.

7. **hyunsik:** +1

   Thank you for the quick fix.

8. **hyunsik:** I've just commit the latest patch submitted in RB. Also, I appreciate Min for the detailed report. It was very helpful to find the problem and fix.

9. **hudson:** SUCCESS: Integrated in Tajo-master-build #21 (See [https://builds.apache.org/job/Tajo-master-build/21/])
   TAJO-522: OutOfMemoryError: unable to create new native thread. (hyoungjunkim via hyunsik) (hyunsik: https://git-wip-us.apache.org/repos/asf?p=incubator-tajo.git&a=commit&h=948915151b1477
   * tajo-rpc/src/main/java/org/apache/tajo/rpc/NettyClientBase.java
   * tajo-common/src/main/java/org/apache/tajo/conf/TajoConf.java
   * CHANGES.txt
   * tajo-rpc/pom.xml
   * tajo-core/tajo-core-backend/src/main/java/org/apache/tajo/master/TajoContainerProxy.java

10. **coderplay:** Hi all, I ran a big query applied this patch, still OOM. Please submit this patch and I will the next step.

11. **hyunsik:** Is your OOM the same phenomenon reported here?

12. **coderplay:** Yes, still create so many threads. As I mentioned on the reviewboard, this patch can significantly alleviate the native memory pressure of worker. However, if the quantity of different connections is large, this patch won't help.

13. **hudson:** SUCCESS: Integrated in Tajo-master-build #22 (See [https://builds.apache.org/job/Tajo-master-build/22/])

TAJO-537: After TAJO-522, still OutOfMemoryError: unable to create new native thread. (Min Zhou via hyunsik) (hyunsik: https://git-wip-us.apache.org/repos/asf?p=incubator-tajo.git&a=commit&h=e57cf426d8d1d1b924a32a10f81a8912f2b0c45f)
* tajo-rpc/src/main/java/org/apache/tajo/rpc/AsyncRpcClient.java
* CHANGES.txt
* tajo-rpc/src/main/java/org/apache/tajo/rpc/BlockingRpcClient.java
* tajo-rpc/src/main/java/org/apache/tajo/rpc/NettyClientBase.java
* tajo-core/tajo-core-backend/src/main/java/org/apache/tajo/master/querymaster/QueryInProgress.java

Chapter 3

Connected issue TAJO-537

3.1 Summary

After TAJO-522, still OutOfMemoryError: unable to create new native thread

3.2 Description

I ran a big query applied this patch, still OOM. This is because if the quantity of different connections is large, Hyoungjun's patch won't help. Different connection create different thread pool.
Actually, we can share the thread pool among all the clients.

3.3 Attachments

1. TAJO-537.patch
2. TAJO-537-v2.patch

3.4 Comments

1. **coderplay:** I tested it in a real cluster. This patch worked well.
2. **coderplay:** Cleaner patch.
3. **hyunsik:** +1

   The patch looks pretty good for me. I'll commit.

   There is one suggestion that can be dealt in a separate jira issue. It would be great to move the factory construction outside from the NettyClientBase. Then, factory release becomes possible when a JVM shutdowns.
4. **hudson:** SUCCESS: Integrated in Tajo-master-build #22 (See [https://builds.apache.org/job/Tajo-master-build/22/])
   TAJO-537: After TAJO-522, still OutOfMemoryError: unable to create new native thread. (Min Zhou via hyunsik) (hyunsik: https://git-wip-us.apache.org/repos/asf?p=incubator-tajo.git&a=commit&h=e57cf426d8d1d1b924a32a10f81a8912f2b0c45f)
   * tajo-rpc/src/main/java/org/apache/tajo/rpc/AsyncRpcClient.java
   * CHANGES.txt

* tajo-rpc/src/main/java/org/apache/tajo/rpc/BlockingRpcClient.java
* tajo-rpc/src/main/java/org/apache/tajo/rpc/NettyClientBase.java
* tajo-core/tajo-core-backend/src/main/java/org/apache/tajo/master/querymaster/QueryInProgress.java

5. **coderplay:** Sure. Let's do it through another ticket. Actually, I still see over 300 threads in worker's process if I run a big query. But the number will drop later. Seems it's due to the thread pool is a cached pool.

6. **hyunsik:** committed it to master branch.

7. **hyunsik:** Thank you Min for your contribution.