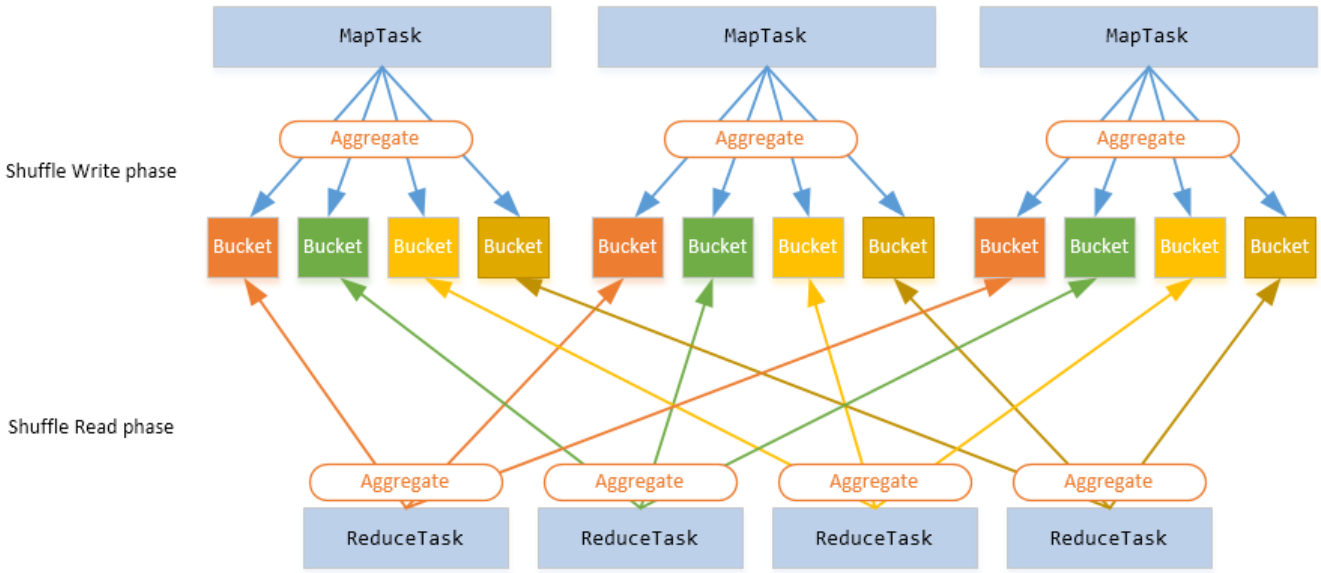
SparkRDMA：使用RDMA技术提升Spark的Shuffle性能

Spark Shuffle 基础

在 MapReduce 框架中，Shuffle 是连接 Map 和 Reduce 之间的桥梁，Reduce 要读取到 Map 的输出必须要经过 Shuffle 这个环节；而 Reduce 和 Map 过程通常不在一台节点，这意味着 Shuffle 阶段通常需要跨网络以及一些磁盘的读写操作，因此 Shuffle 的性能高低直接影响了整个程序的性能和吞吐量。

与 MapReduce 计算框架一样，Spark 作业也有 Shuffle 阶段，通常以 Shuffle 来划分 Stage；而 Stage 之间的数据交互是需要 Shuffle 来完成的。整个过程图如下所示：



如果想及时了解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

从上面简单的介绍可以得到以下几个结论：

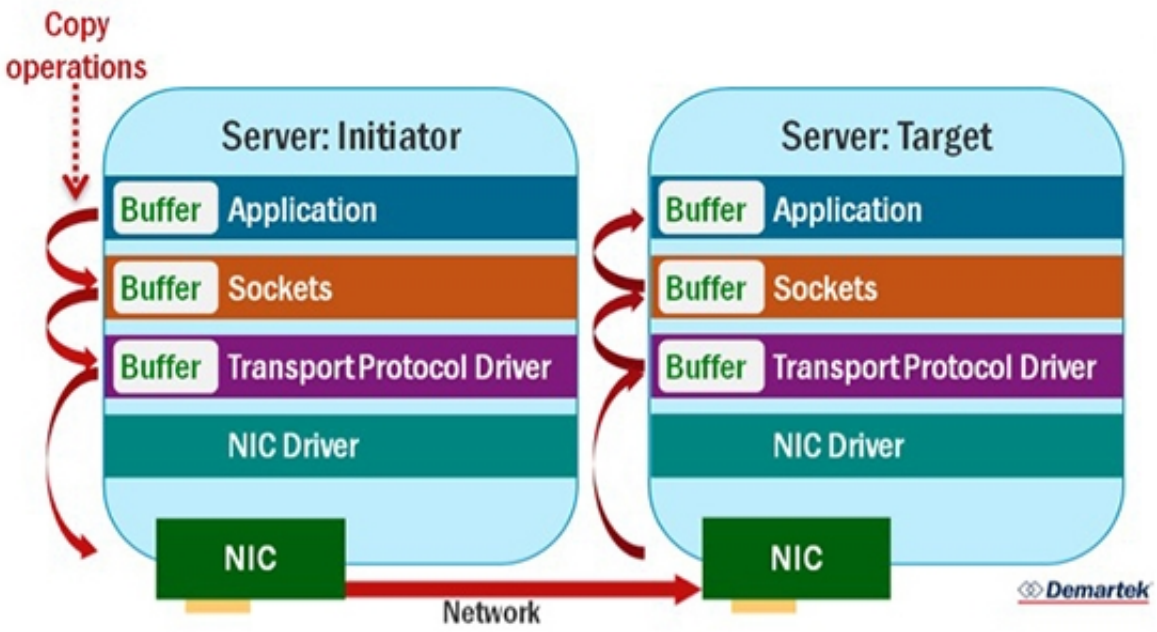
- 不管是 MapReduce 还是 Spark 作业，Shuffle 操作是很消耗资源的，这里的资源包括：CPU、RAM、磁盘还有网络；
- 我们需要尽可能地避免 Shuffle 操作

而目前最新的 Spark（2.2.0）内置只支持一种 Shuffle 实现：org.apache.spark.shuffle.sort.SortShuffleManager，通过参数 spark.shuffle.manager 来配置。这是标准的 Spark Shuffle 实现，其内部实现依赖了 Netty 框架。本文并不打算详细介绍 Spark 内部 Shuffle 是如何实现的，这里我要介绍社区对 Shuffle 的改进。

RDMA 技术

在进行下面的介绍之前，我们先来了解一些基础知识。

传统的 TCP Socket 数据传输需要经过很多步骤：数据先从源端应用程序拷贝到当前主机的 Sockets 缓存区，然后再拷贝到 TransportProtocol Driver，然后到 NIC Driver，最后 NIC 通过网络将数据发送到目标主机的
NIC，目标主机又经过上面步骤将数据传输到应用程序，整个过程如下：

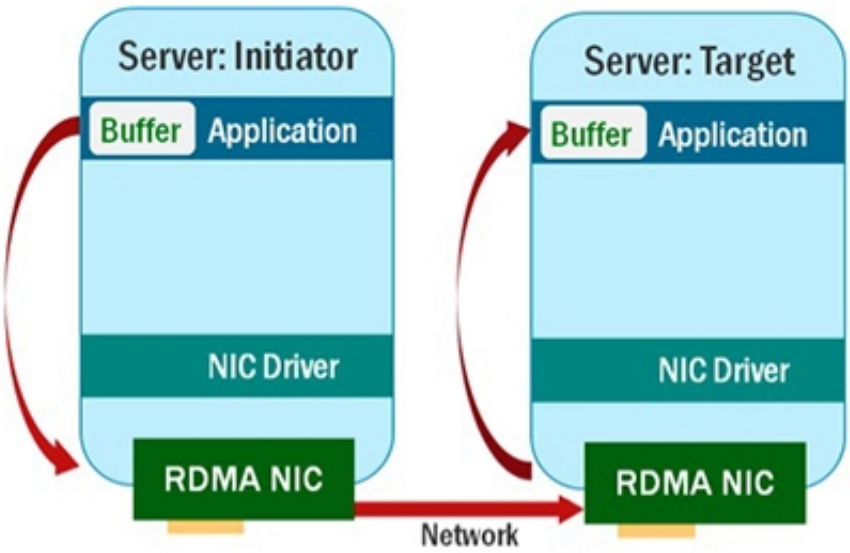


从上图可以看出，网络数据的传输很大一部分时间用于数据的拷贝！如果需要传输的数据很大，那么这个阶段用的时间很可能占整个作业运行时间的很大一部分！那么有没有一种方法直接省掉不同层的数据拷贝，使得目标主机直接从源端主机内存获取数据？还真有，这就是 RDMA 技术！

RDMA（Remote Direct Memory
Access）技术全称远程直接内存访问
，是一种直接内存访问技术，它将数据直接从一台计算机的内存传输到另一台计算机，无需双方操作系统的介入。这允许高通量、低延迟的网络通信，尤其适合在大规模并行计算机集群中使用（本段摘抄自 维基百科 - 远程直接内存访问）。RDMA 有以下几个特点：

- Zero-copy
- 直接硬件接口（Direct hardware interface），绕过内核和 TCP / IP 的 IO
- 亚微秒延迟
- Flow control and reliability is offloaded in hardware

所以利用 RDMA 技术进行数据传输看起来像下面一样：



如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

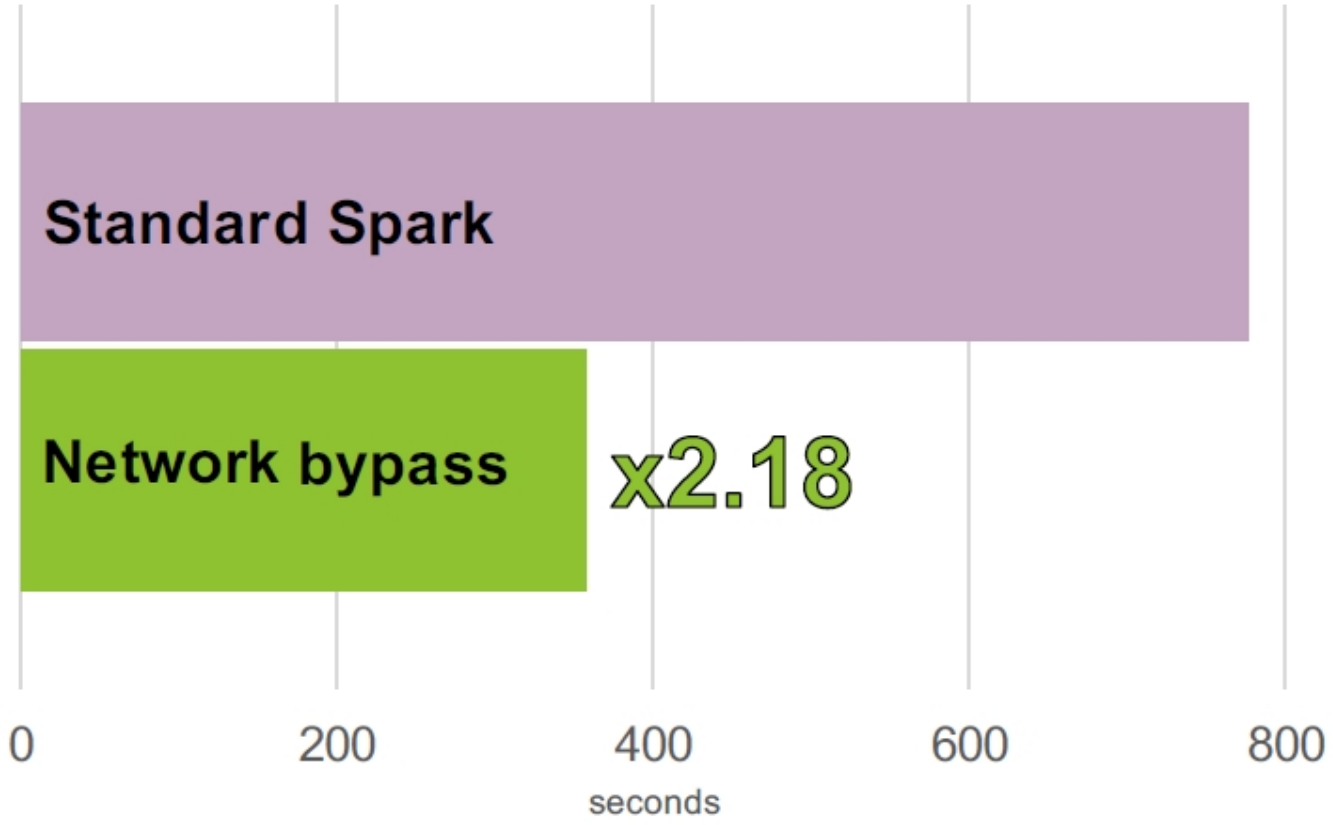
从上面看出，使用了 RDMA 技术之后，虽然源端主机和目标主机是跨网络的，但是他们之间的数据交互是直接从对方内存获取的，这明显会加快整个计算过程。

SparkRDMA

好，现在基础的知识咱们已经获取到了，我们正式进入本文主题。由 Mellanox Technologies 公司开发并开源的 SparkRDMA ShuffleManager （ GitHub 地址：https://github.com/Mellanox/SparkRDMA ）就是采用 RDMA 技术，使得 Spark 作业在 Shuffle 数据的时候使用 RDMA 方式，而非标准的 TCP 方式。在 SparkRDMA 的官方 Wiki 里面有如下介绍：

SparkRDMA is a high-performance, scalable and efficient ShuffleManager plugin for Apache Spark. It utilizes RDMA (Remote Direct Memory Access) technology to reduce CPU cycles needed for Shuffle data transfers. It reduces memory usage by reusing memory for transfers instead of copying data multiple times down the traditional TCP-stack.

可以看出，SparkRDMA 就是扩展了 Spark 的 ShuffleManager 接口，并且采用了 RDMA 技术。在测试的结果显示，采用 RDMA 进行 Shuffle 数据比标准的方式快 2.18 倍！



如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

SparkRDMA 开发者们给 Spark 社区提交了一个 Issue：[SPARK-22229] SPIP: RDMA Accelerated Shuffle Engine，详细的设计文档：这里。不过从社区的回复来看，最少目前不会整合到 Spark 代码中去。

安装使用

如果你想使用 SparkRDMA，我们需要 Apache Spark 2.0.0/2.1.0/2.2.0、Java 8 以及支持 RDMA 技术的网络（比如：RoCE 和 Infiniband）。

SparkRDMA 官方为不同版本的 Spark 预先编译好相应的 jar 包，我们可以访问 这里
下载。解压之后会得到以下四个文件：

- spark-rdma-1.0-for-spark-2.0.0-jar-with-dependencies.jar
- spark-rdma-1.0-for-spark-2.1.0-jar-with-dependencies.jar
- spark-rdma-1.0-for-spark-2.2.0-jar-with-dependencies.jar
- libdisni.so

除了 libdisni.so 文件一定要安装到 Spark 集群的所有节点上，其他的 jar 包只需要根据我们的 Spark 版本进行选择。相关的文件部署好之后，我们需要将这个 SparkRDMA 模块加入到 Spark 的运行环境中去，如下设置：

```
spark.driver.extraClassPath   /path/to/SparkRDMA/spark-rdma-1.0-for-spark-2.0.0-jar-with-dependencies.jar
spark.executor.extraClassPath /path/to/SparkRDMA/spark-rdma-1.0-for-spark-2.0.0-jar-with-dependencies.jar
```

为了启用 SparkRDMA Shuffle Manager 插件，我们还需要修改 spark.shuffle.manager 的值，只需要在 $SPARK_HOME/conf/spark-defaults.conf 里面加入以下的配合即可：

```
spark.shuffle.manager   org.apache.spark.shuffle.rdma.RdmaShuffleManager
```

其他的就和正常使用 Spark 一样。

关于配置libdisni.so

我们需要将 libdisni.so 文件分发到集群的所有节点的同一目录下，然后配置下面的环境：

```
export JAVA_LIBRARY_PATH=$JAVA_LIBRARY_PATH:/home/iteblog/spark-2.1.0-bin/rdma/
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/iteblog/spark-2.1.0-bin/rdma/
export SPARK_YARN_USER_ENV="JAVA_LIBRARY_PATH=$JAVA_LIBRARY_PATH,LD_LIBRARY_PATH=$LD_LIBRARY_PATH"
```

其中 /home/iteblog/spark-2.1.0-bin/rdma/ 存放了libdisni.so 文件。运行的过程中可能还需要 libibverbs.so.1 和 librdmacm.so.1 文件，可以通过下面命令解决

```
yum -y install   libibverbs  librdmacm
```

然后可以通过下面命令启动 Spark：

```
bin/spark-shell --master yarn-client --driver-memory 18g  --executor-memory 15g              ₩
      --queue iteblog --executor-cores 1  --num-executors 8                                  ₩
      --conf "spark.yarn.dist.archives=/home/iteblog/spark-2.1.0-bin/rdma/rdma.tgz"          ₩
      --conf "spark.executor.extraLibraryPath=/home/iteblog/spark-2.1.0-bin/rdma/libdisni.so"
₩
```

```
--conf "spark.driver.extraLibraryPath=/home/iteblog/spark-2.1.0-bin/rdma/libdisni.so"  ₩
--conf "spark.executor.extraClassPath=rdma.tgz/rdma/*"                                ₩
--conf "spark.driver.extraClassPath=/home/iteblog/spark-2.1.0-bin/rdma/*"             ₩
--conf "spark.shuffle.manager=org.apache.spark.shuffle.rdma.RdmaShuffleManager"
```

不过如果你网络不支持 RDMA 技术，那么就像我一样会遇到下面的问题：

```
17/11/15 22:01:48 ERROR rdma.RdmaNode: Failed in RdmaNode constructor
17/11/15 22:01:48 ERROR spark.SparkContext: Error initializing SparkContext.
java.lang.reflect.InvocationTargetException
 at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
 at sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.ja
va:62)
 at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccesso
rImpl.java:45)
 at java.lang.reflect.Constructor.newInstance(Constructor.java:423)
 at org.apache.spark.SparkEnv$.instantiateClass$1(SparkEnv.scala:265)
 at org.apache.spark.SparkEnv$.create(SparkEnv.scala:323)
 at org.apache.spark.SparkEnv$.createDriverEnv(SparkEnv.scala:174)
 at org.apache.spark.SparkContext.createSparkEnv(SparkContext.scala:257)
 at org.apache.spark.SparkContext.<init>(SparkContext.scala:432)
 at org.apache.spark.SparkContext$.getOrCreate(SparkContext.scala:2313)
 at org.apache.spark.sql.SparkSession$Builder$$anonfun$6.apply(SparkSession.scala:868)
 at org.apache.spark.sql.SparkSession$Builder$$anonfun$6.apply(SparkSession.scala:860)
 at scala.Option.getOrElse(Option.scala:121)
 at org.apache.spark.sql.SparkSession$Builder.getOrCreate(SparkSession.scala:860)
 at org.apache.spark.repl.Main$.createSparkSession(Main.scala:95)
 at $line3.$read$$iw$$iw.<init>(<console>:15)
 at $line3.$read$$iw.<init>(<console>:42)
 at $line3.$read.<init>(<console>:44)
 at $line3.$read$.<init>(<console>:48)
 at $line3.$read$.<clinit>(<console>)
 at $line3.$eval$.$print$lzycompute(<console>:7)
 at $line3.$eval$.$print(<console>:6)
 at $line3.$eval.$print(<console>)
 at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
 at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
 at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
 at java.lang.reflect.Method.invoke(Method.java:498)
 at scala.tools.nsc.interpreter.IMain$ReadEvalPrint.call(IMain.scala:786)
 at scala.tools.nsc.interpreter.IMain$Request.loadAndRun(IMain.scala:1047)
 at scala.tools.nsc.interpreter.IMain$WrappedRequest$$anonfun$loadAndRunReq$1.apply(IM
ain.scala:638)
```

```
 at scala.tools.nsc.interpreter.IMain$WrappedRequest$$anonfun$loadAndRunReq$1.apply(IM
ain.scala:637)
 at scala.reflect.internal.util.ScalaClassLoader$class.asContext(ScalaClassLoader.scala:31)
 at scala.reflect.internal.util.AbstractFileClassLoader.asContext(AbstractFileClassLoader.scala:1
9)
 at scala.tools.nsc.interpreter.IMain$WrappedRequest.loadAndRunReq(IMain.scala:637)
 at scala.tools.nsc.interpreter.IMain.interpret(IMain.scala:569)
 at scala.tools.nsc.interpreter.IMain.interpret(IMain.scala:565)
 at scala.tools.nsc.interpreter.ILoop.interpretStartingWith(ILoop.scala:807)
 at scala.tools.nsc.interpreter.ILoop.command(ILoop.scala:681)
 at scala.tools.nsc.interpreter.ILoop.processLine(ILoop.scala:395)
 at org.apache.spark.repl.SparkILoop$$anonfun$initializeSpark$1.apply$mcV$sp(SparkILoop.s
cala:38)
 at org.apache.spark.repl.SparkILoop$$anonfun$initializeSpark$1.apply(SparkILoop.scala:37)
 at org.apache.spark.repl.SparkILoop$$anonfun$initializeSpark$1.apply(SparkILoop.scala:37)
 at scala.tools.nsc.interpreter.IMain.beQuietDuring(IMain.scala:214)
 at org.apache.spark.repl.SparkILoop.initializeSpark(SparkILoop.scala:37)
 at org.apache.spark.repl.SparkILoop.loadFiles(SparkILoop.scala:105)
 at scala.tools.nsc.interpreter.ILoop$$anonfun$process$1.apply$mcZ$sp(ILoop.scala:920)
 at scala.tools.nsc.interpreter.ILoop$$anonfun$process$1.apply(ILoop.scala:909)
 at scala.tools.nsc.interpreter.ILoop$$anonfun$process$1.apply(ILoop.scala:909)
 at scala.reflect.internal.util.ScalaClassLoader$.savingContextLoader(ScalaClassLoader.scala:97
)
 at scala.tools.nsc.interpreter.ILoop.process(ILoop.scala:909)
 at org.apache.spark.repl.Main$.doMain(Main.scala:68)
 at org.apache.spark.repl.Main$.main(Main.scala:51)
 at org.apache.spark.repl.Main.main(Main.scala)
 at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
 at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
 at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
 at java.lang.reflect.Method.invoke(Method.java:498)
 at org.apache.spark.deploy.SparkSubmit$.org$apache$spark$deploy$SparkSubmit$$runMain
(SparkSubmit.scala:738)
 at org.apache.spark.deploy.SparkSubmit$.doRunMain$1(SparkSubmit.scala:187)
 at org.apache.spark.deploy.SparkSubmit$.submit(SparkSubmit.scala:212)
 at org.apache.spark.deploy.SparkSubmit$.main(SparkSubmit.scala:126)
 at org.apache.spark.deploy.SparkSubmit.main(SparkSubmit.scala)
Caused by: java.io.IOException: Unable to allocate RDMA Event Channel
 at org.apache.spark.shuffle.rdma.RdmaNode.<init>(RdmaNode.java:67)
 at org.apache.spark.shuffle.rdma.RdmaShuffleManager.<init>(RdmaShuffleManager.scala:181
)
 ... 62 more
java.io.IOException: Unable to allocate RDMA Event Channel
  at org.apache.spark.shuffle.rdma.RdmaNode.<init>(RdmaNode.java:67)
  at org.apache.spark.shuffle.rdma.RdmaShuffleManager.<init>(RdmaShuffleManager.scala:18
1)
```

```
  at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
  at sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62)
  at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:45)
  at java.lang.reflect.Constructor.newInstance(Constructor.java:423)
  at org.apache.spark.SparkEnv$.instantiateClass$1(SparkEnv.scala:265)
  at org.apache.spark.SparkEnv$.create(SparkEnv.scala:323)
  at org.apache.spark.SparkEnv$.createDriverEnv(SparkEnv.scala:174)
  at org.apache.spark.SparkContext.createSparkEnv(SparkContext.scala:257)
  at org.apache.spark.SparkContext.<init>(SparkContext.scala:432)
  at org.apache.spark.SparkContext$.getOrCreate(SparkContext.scala:2313)
  at org.apache.spark.sql.SparkSession$Builder$$anonfun$6.apply(SparkSession.scala:868)
  at org.apache.spark.sql.SparkSession$Builder$$anonfun$6.apply(SparkSession.scala:860)
  at scala.Option.getOrElse(Option.scala:121)
  at org.apache.spark.sql.SparkSession$Builder.getOrCreate(SparkSession.scala:860)
  at org.apache.spark.repl.Main$.createSparkSession(Main.scala:95)
  ... 47 elided
```

这样的话那就没法测试了，哈哈。。如果真要使用 RDMA ，咨询你公司的 OPS 如何配置这个吧。