在Tachyon运行Spark应用程序

我们在《Tachyon 0.7.0伪分布式集群安装与测试》文章中介绍了如何搭建伪分布式Tachyon集群。从官方文档得知，Spark 1.4.x和Tachyon 0.6.4版本兼容，而最新版的Tachyon 0.7.1和Spark 1.5.x兼容，目前最新版的Spark为1.4.1，所以下面的操作步骤全部是基于Tachyon 0.6.4平台的，Tachyon 0.6.4的搭建步骤和Tachyon 0.7.0类似。

废话不多说，开始介绍吧。我们先在HDFS上传一个文件，比如iteblog.txt，存放目录为/data：

```
[iteblog@www.iteblog.com hadoop]$  bin/hadoop fs -put iteblog.txt /data
```

启动Spark-shell

```
[iteblog@www.iteblog.com spark]$  bin/spark-shell
```

这时候我们可以通过Tachyon获取iteblog.txt文件，如下：

```
scala>  val s = sc.textFile("tachyon://localhost:19998/data/iteblog.txt")
15/08/31 14:15:24 INFO storage.MemoryStore: ensureFreeSpace(156896) called with curMem
=216700, maxMem=280248975
15/08/31 14:15:24 INFO storage.MemoryStore: Block broadcast_3 stored as values in memory
(estimated size 153.2 KB, free 266.9 MB)
15/08/31 14:15:24 INFO storage.MemoryStore: ensureFreeSpace(14945) called with curMem=
373596, maxMem=280248975
15/08/31 14:15:24 INFO storage.MemoryStore: Block broadcast_3_piece0 stored as bytes in m
emory (estimated size 14.6 KB, free 266.9 MB)
15/08/31 14:15:24 INFO storage.BlockManagerInfo: Added broadcast_3_piece0 in memory on l
ocalhost:55566 (size: 14.6 KB, free: 267.2 MB)
15/08/31 14:15:24 INFO spark.SparkContext: Created broadcast 3 from textFile at <console>:2
1
s: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[4] at textFile at <console>:21
```

其中tachyon://localhost:19998就是你Tachyon的通信地址。我们可以看到，ji/data/iteblog.txt对应的就是HDFS上的文件，这个是怎么获取到的呢？其实是在Tachyon的conf/tachyon-env.sh文件里面配置的，通过export TACHYON_UNDERFS_ADDRESS=hdfs://iteblog.com:8020配置，这个就是配置咱们HDFS集群的通信地址，这样我们就可以通过tachyon找到那个文件。好了，我们执行一下Action操作：

```
scala> s.count()
15/08/31 14:15:45 INFO : getFileStatus(/data/iteblog.txt): HDFS Path: hdfs://localhost:8020/data/iteblog.txt TPath: tachyon://localhost:19998/data/iteblog.txt
15/08/31 14:15:45 INFO : Loading to /data/iteblog.txt hdfs://localhost:8020/data/iteblog.txt
15/08/31 14:15:45 INFO : Loading: hdfs://localhost:8020/data/iteblog.txt
15/08/31 14:15:46 INFO : Create tachyon file /data/iteblog.txt/iteblog.txt with file id 15 and checkpoint location hdfs://localhost:8020/data/iteblog.txt
15/08/31 14:15:46 INFO : listStatus(tachyon://localhost:19998/data/iteblog.txt): HDFS Path: hdfs://localhost:8020/data/iteblog.txt
15/08/31 14:15:46 INFO : getFileStatus(tachyon://localhost:19998/data/iteblog.txt/iteblog.txt): HDFS Path: hdfs://localhost:8020/data/iteblog.txt/iteblog.txt TPath: tachyon://localhost:19998/data/iteblog.txt/iteblog.txt
15/08/31 14:15:46 INFO mapred.FileInputFormat: Total input paths to process : 1
15/08/31 14:15:46 INFO spark.SparkContext: Starting job: count at <console>:24
15/08/31 14:15:46 INFO scheduler.DAGScheduler: Got job 2 (count at <console>:24) with 2 output partitions (allowLocal=false)
15/08/31 14:15:46 INFO scheduler.DAGScheduler: Final stage: ResultStage 2(count at <console>:24)
15/08/31 14:15:46 INFO scheduler.DAGScheduler: Parents of final stage: List()
15/08/31 14:15:46 INFO scheduler.DAGScheduler: Missing parents: List()
15/08/31 14:15:46 INFO scheduler.DAGScheduler: Submitting ResultStage 2 (MapPartitionsRDD[4] at textFile at <console>:21), which has no missing parents
15/08/31 14:15:46 INFO storage.MemoryStore: ensureFreeSpace(2992) called with curMem=388541, maxMem=280248975
15/08/31 14:15:46 INFO storage.MemoryStore: Block broadcast_4 stored as values in memory (estimated size 2.9 KB, free 266.9 MB)
15/08/31 14:15:46 INFO storage.MemoryStore: ensureFreeSpace(1828) called with curMem=391533, maxMem=280248975
15/08/31 14:15:46 INFO storage.MemoryStore: Block broadcast_4_piece0 stored as bytes in memory (estimated size 1828.0 B, free 266.9 MB)
15/08/31 14:15:46 INFO storage.BlockManagerInfo: Added broadcast_4_piece0 in memory on localhost:55566 (size: 1828.0 B, free: 267.2 MB)
15/08/31 14:15:46 INFO spark.SparkContext: Created broadcast 4 from broadcast at DAGScheduler.scala:874
15/08/31 14:15:46 INFO scheduler.DAGScheduler: Submitting 2 missing tasks from ResultStage 2 (MapPartitionsRDD[4] at textFile at <console>:21)
15/08/31 14:15:46 INFO scheduler.TaskSchedulerImpl: Adding task set 2.0 with 2 tasks
15/08/31 14:15:46 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 2.0 (TID 4, localhost, PROCESS_LOCAL, 1427 bytes)
15/08/31 14:15:46 INFO executor.Executor: Running task 0.0 in stage 2.0 (TID 4)
```

```
15/08/31 14:15:46 INFO rdd.HadoopRDD: Input split: tachyon://localhost:19998/data/iteblog.txt/iteblog.txt:0+5840
15/08/31 14:15:46 INFO : open(tachyon://localhost:19998/data/iteblog.txt/iteblog.txt, 65536)
15/08/31 14:15:46 INFO : /mnt/ramdisk/tachyonworker/users/2/16106127360 was created!
15/08/31 14:15:46 INFO : Try to find remote worker and read block 16106127360 from 0, with len 11680
15/08/31 14:15:46 INFO : Block locations:[NetAddress(mHost:localhost, mPort:-1, mSecondaryPort:-1)]
15/08/31 14:15:46 INFO : Block locations:[NetAddress(mHost:localhost, mPort:-1, mSecondaryPort:-1)]
15/08/31 14:15:46 INFO : Opening stream from underlayer fs: hdfs://localhost:8020/data/iteblog.txt
15/08/31 14:15:46 INFO executor.Executor: Finished task 0.0 in stage 2.0 (TID 4). 1830 bytes result sent to driver
15/08/31 14:15:46 INFO scheduler.TaskSetManager: Starting task 1.0 in stage 2.0 (TID 5, localhost, PROCESS_LOCAL, 1427 bytes)
15/08/31 14:15:46 INFO executor.Executor: Running task 1.0 in stage 2.0 (TID 5)
15/08/31 14:15:46 INFO rdd.HadoopRDD: Input split: tachyon://localhost:19998/data/iteblog.txt/iteblog.txt:5840+5840
15/08/31 14:15:46 INFO : open(tachyon://localhost:19998/data/iteblog.txt/iteblog.txt, 65536)
15/08/31 14:15:46 INFO executor.Executor: Finished task 1.0 in stage 2.0 (TID 5). 1830 bytes result sent to driver
15/08/31 14:15:46 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 2.0 (TID 4) in 543 ms on localhost (1/2)
15/08/31 14:15:46 INFO scheduler.DAGScheduler: ResultStage 2 (count at <console>:24) finished in 0.555 s
15/08/31 14:15:46 INFO scheduler.DAGScheduler: Job 2 finished: count at <console>:24, took 0.651055 s
res2: Long = 212
```

这样我们获取到了iteblog.txt文件里面的行数。通过上面运行的日志我们知道，其实Tachyon本身将iteblog.txt文件加载到了内存，并存放到自身的文件系统里面：tachyon://localhost:19998/data/iteblog.txt/iteblog.txt，我们可以在Tachyon的 WEB UI界面（在我这路径为http://localhost:19999/browse?path=%2Fdata%2Fiteblog.txt%offset=0&limit=1）看到这个文件。

我们还可以将计算结果保存到Tachyon中：

```
scala> s.saveAsTextFile("tachyon://localhost:19998/iteblog")
```

我们同样可以在Tachyon的 WEB UI界面看到这个文件，并且在里面创建了iteblog文件夹，里面的

数据就是RDD的数据。仔细的读者还会发现，在HDFS上还生成了一个文件夹，里面存放了RDD的数据，如下：

```
[iteblog@www.iteblog.com hadoop]$  bin/hadoop fs -ls /tachyon/data
Found 6 items
-rwxrwxrwx   3 iteblog supergroup      13367 2015-08-31 14:02 /tachyon/data/11
-rwxrwxrwx   3 iteblog supergroup          0 2015-08-31 14:02 /tachyon/data/12
-rwxrwxrwx   3 iteblog supergroup       5890 2015-08-31 14:21 /tachyon/data/21
-rwxrwxrwx   3 iteblog supergroup       5790 2015-08-31 14:21 /tachyon/data/23
-rwxrwxrwx   3 iteblog supergroup          0 2015-08-31 14:21 /tachyon/data/24
-rwxrwxrwx   3 iteblog supergroup      13491 2015-08-31 14:02 /tachyon/data/9
```

其实这个路径是通过conf/tachyon-env.sh里面的-Dtachyon.data.folder=$TACHYON_UNDERFS_ADDRESS/tachyon/data配置的。

我们还可以将RDD cache到Tachyon上，通过设置缓存级别为StorageLevel.OFF_HEAP即可。这个可以减少GC的频率，并且减少executors占用的资源，最好的就是可以使得不同的Application之间可以共享RDD的数据。下面举个例子：

```
scala> import org.apache.spark.storage.StorageLevel
import org.apache.spark.storage.StorageLevel

scala> val data=sc.parallelize(List("www", "iteblog", "com"))
data: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[6] at parallelize at <console>:22

scala> val tmp = data.map(item => item +"good")
tmp: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[8] at map at <console>:24

scala> tmp.persist(StorageLevel.OFF_HEAP)
res9: tmp.type = MapPartitionsRDD[8] at map at <console>:24

scala> tmp.count
```

然后你就可以在Tachyon的 WEB UI界面上看到缓存的RDD存储目录。而且这些RDD是存放在内存文件系统中的。在使用过程中，我们可以通过spark.externalBlockStore.url参数设置Tachyon filesystem的URL，默认为tachyon://localhost:19998；通过spark.externalBlockStore.baseDir设置Tachyon File System中存放RDD的基路径，默认是System.getProperty("java.io.tmpdir")的值。

本博客文章除特别声明，全部都是原创！
原创文章版权归过往记忆大数据（过往记忆）所有，未经许可不得转载。
本文链接:【】（）