

一：BlockManager运行实例观察

**从Application启动的角度来观察BlockManager**

1. 在Application启动的时候还在SparkEnv中注册BlockManager以及MapOutputTracker,其中
2. **BlockManagerMaster**: 对整个集群的Block数据进行管理的(Block是Spark处理数据的单位)
3. **MapOutputTrackerMaster**: 跟踪所有的Mapper的输出的



1. **BlockManagerMasterEndpoint**：本身是一个消息体，会负责通过远程消息通信的方式去管理所有的节点BlockManager。构造BlockManagerMaster的时候内部会构造BlockManagerMasterEndpoint.

**val** blockManagerMaster = **new** BlockManagerMaster(registerOrLookupEndpoint(  
 BlockManagerMaster.*DRIVER\_ENDPOINT\_NAME*,  
 **new** BlockManagerMasterEndpoint(rpcEnv, isLocal, conf, listenerBus)),  
 conf, isDriver)

1. 每启动一个ExecutorBackend(Executor是进程的对象，ExecutorBackend才是进程代表)都会实例化BlockManager并通过远程通信的方式注册给BlockManagerMaster；实质上是Executor中的BlockManager在启动的时候注册给了Driver上的BlockManagerMasterEndpoint**。BlockManagerMasterEndpoint是Driver中的一个对象而已，无法对一个对象做HA，从集群的运营角度来讲，可以对进程做HA**
2. **在实际表现出来的是BlockManagerMaster，其内部是通过BlockManagerMasterEndpoint具体去完成任务的。BlockManagerMaster只是Driver中的一个对象。**
3. MemoryStore：是BlockManager中专门负责内存数据存储和读写的类。**它的存储单位是以block为单位的。**
4. **DiskStore**：是BlockManager中专门负责基于磁盘的数据存储和读写的类。

*/\*\*  
 \* Stores BlockManager blocks on disk.  
 \*/***private**[spark] **class** DiskStore(blockManager: BlockManager, diskManager: DiskBlockManager)

1. **DiskBlockManager**：管理Logical Block(逻辑级别的，程序运行级别的) 与Disk上的Physical Block之间的映射关系并负责磁盘的文件的创建，读写等；

*/\*\*  
 \* Creates and maintains the logical mapping between logical blocks and physical on-disk  
 \* locations. By default, one block is mapped to one file with a name given by its BlockId.  
 \* However, it is also possible to have a block map to only a segment of a file, by calling  
 \* mapBlockToFileSegment().  
 \*  
 \* Block files are hashed among the directories listed in spark.local.dir (or in  
 \* SPARK\_LOCAL\_DIRS, if it's set).  
 \*/*

**从job运行的角度来观察BlockManager：**

**BlockManagerInfo**：是Driver中为了管理所有ExecutorBackend中的BlockManager上面的所有元数据而设立的。

1. 首先通过MemoryStore来存储广播变量；
2. 在Driver中是通过BlockManagerInfo来管理集群中每个ExecutorBackend中的BlockManager中的元数据信息。
3. 当改变了具体的ExecutorBackend上的Block信息后就必须发消息给Driver中的BlocalManagerMaster来更新相应的BlockManagerInfo；



1. 当执行第二个Stage的时候，第二个Stage会向Driver中的MapOutputTrackMasterEndpoint发消息，请求上一个Stage中相应的输出，此时MapOutputTrackMaster会把上一个Stage的输出数据的元数据信息发送给当前请求的Stage.



补讲：

Spark-shell详解

1. Spark启动的env



在sparkEnv中注册mapOutputTracker

// Have to assign trackerActor after initialization as MapOutputTrackerActor  
// requires the MapOutputTracker itself  
mapOutputTracker.*trackerEndpoint* = registerOrLookupEndpoint(MapOutputTracker.*ENDPOINT\_NAME*,  
 **new** MapOutputTrackerMasterEndpoint(  
 rpcEnv, mapOutputTracker.asInstanceOf[MapOutputTrackerMaster], conf))

BlockManagerMaster被注册

**val** blockManagerMaster = **new** BlockManagerMaster(registerOrLookupEndpoint(  
 BlockManagerMaster.*DRIVER\_ENDPOINT\_NAME*,  
 **new** BlockManagerMasterEndpoint(rpcEnv, isLocal, conf, listenerBus)),  
 conf, isDriver)

管理磁盘存储的，在本地/tmp目录下创建文件以blockmgr- 开头的文件



