# Spark Architecture

## Block Manager

**Block Manager** is a key-value store for blocks of data in Spark. Block Manager acts as a local cache that runs on every node in Spark cluster, i.e. the driver and executors. It provides interface for uploading and fetching blocks both locally and remotely using various stores, i.e. memory, disk, and off-heap.

A BlockManager is a BlockDataManager, i.e. manages the storage for blocks that can

represent a cached RDD partition, an intermediate shuffle data, a broadcast data, etc. It is also a BlockEvictionHandler that drops a block from memory and storing it on a disk if applicable.（BlockManager extends BlockDataManager with BlockEvictionHandler）

**Cached blocks** are blocks with non-zero sum of memory and disk sizes.

It is created when a Spark application starts (as part of SparkEnv.create).

A BlockManager must be initialized before it is fully operable.

A BlockManager relies on the following services:

* RpcEnv
* BlockManagerMaster
* Serializer
* MemoryManager
* MapOutputTracker
* ShuffleManager
* BlockTransferService (BlockManager 之间数据 传输：getRemoteBytes)
* SecurityManager

## MapOutputTracker

Class that keeps track of the location of the map output of a stage. This is abstract because different versions of MapOutputTracker (driver and executor) use different HashMap to store its metadata.

* DAGScheduler向MapOutputTrackerMaster注册shuffle信息（newOrUsedShuffleStage方法）

MapOutputTrackerMaster.registerShuffle

* DAGScheduler 向MapOutputTrackerMaster注册mapoutput信息（handleExecutorLost）

MapOutputTrackerMaster.registerMapOutputs

* BlockStoreShuffleReader Read the combined key-values for this reduce task（）

org.apache.spark.MapOutputTracker.getMapSizesByExecutorId