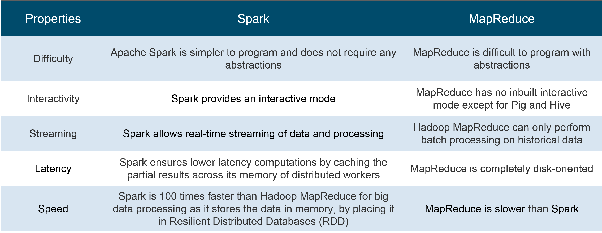
* what is the managed and external tables

|  |  |  |
| --- | --- | --- |
| Table type | Managed table | External table |
| Table create | No need to mention managed | Have to mentiona external key word |
| access | Only one user who created | All user in the cluster |
| Delete table | Table and metadata will be deleted | Only metadata will delete and actual data will be in hdfs only |

* what differ between spark and mapreduce



* waht is yarn

resource schuling

job scheduling

* what is flatmap its jest flat the data

flatmap is flat data ,split the line in to string

The flatMap operation takes a function that conceptually wants to consume one value and produce an arbitrary number of values

* what rdd

RDD(resilent distributed datasets)

its basic abstraction in spark that represent that coming into system in object form

rdd create from 2 ways

collection(arrays,list ,stock,vectors......)

datasets(csv file ,tsc,file ........ file formata data )

operation on rdd are 2 types

transformation:( convertion old rdd to new rdd applying functions)

actions: result bring data into system from all partitions data

features of rdd:

lazy evaluation

cacheable

fault tolarance

typeinfer

type of rdds 2 types

normalRDD

pairRDD

* can we retrieve the external table data from hdfs by again creating the external table

Yes we can retrive

* spark APIs

It provides high-level APIs in Java, Scala, Python and R,

java, scala, python and R

* in spark in memory process then how you optimise memory

In in-memory computation, the data is kept in *random access memory(****RAM****)* instead of some slow disk drives and is processed in parallel. Using this we can detect a pattern, analyze large data. This has become popular because it reduces the cost of memory. So, in-memory processing is economic for applications. The two main columns of in-memory computation are

* RAM storage
* Parallel distributed processing.

Optimisation in spark

memory management

data serialization

garbage collations

partitioning and parallism

action and transformation

caching and persisting

* what is the deference between data node and name node

|  |  |
| --- | --- |
| NameNode | DataNode |
| 1. NameNode is the centerpiece of [**HDFS**](http://data-flair.training/blogs/comprehensive-hdfs-guide-introduction-architecture-data-read-write-tutorial/). 2. NameNode is also known as the Master NameNode only stores the metadata of HDFS – the directory tree of all files in the file system, and tracks the files across the cluster. 3. NameNode does not store the actual data or the dataset. The data itself is actually stored in the DataNodes. 4. NameNode knows the list of the [**Blocks**](http://data-flair.training/blogs/data-blocks-hdfs-hadoop-distributed-file-system/) and its location for any given file in HDFS. With this information NameNode knows how to construct the file from blocks. 5. NameNode is so critical to HDFS and when the NameNode is down, HDFS/[**Hadoop cluster**](http://data-flair.training/blogs/installation-hadoop-3-x-ubuntu-pseudo-distributed-mode/) is inaccessible and considered down. 6. NameNode is a single point of failure in Hadoop cluster. 7. NameNode is usually configured with a lot of memory (RAM). Because the block locations are held in main memory | 1. DataNode is responsible for storing the actual data in HDFS. 2. DataNode is also known as the Slave 3. NameNode and DataNode are in constant communication. 4. When a DataNode starts up it announce itself to the NameNode along with the list of blocks it is responsible for. 5. When a DataNode is down, it does not affect the availability of data or the cluster. NameNode will arrange for replication for the blocks managed by the DataNode that is not available. 6. DataNode is usually configured with a lot of hard disk space. Because the actual data is stored in the DataNode. **DataNode periodically send HEARTBEATS to NameNode** |

* what rdd feature

features of rdd:

lazy evaluation

cacheable

fault tolarance

typeinfer

type of rdds 2 types

normalRDD

pairRDD