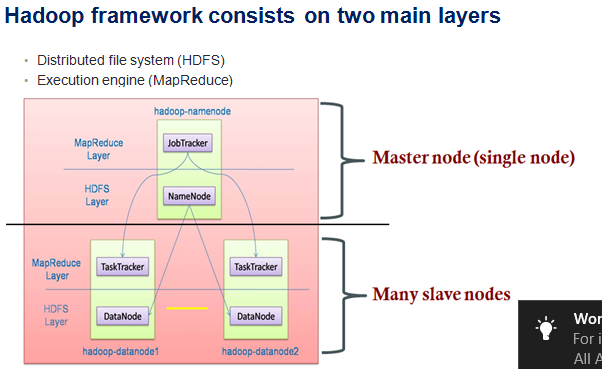
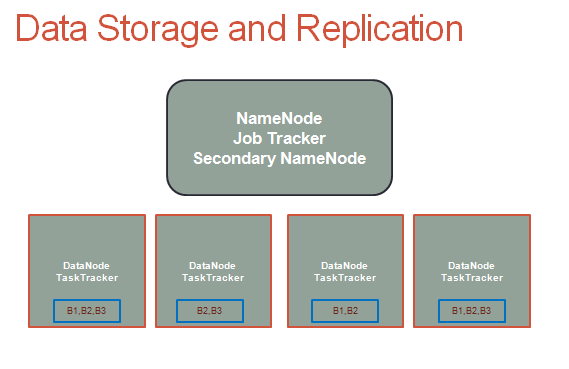
**Hadoop:**

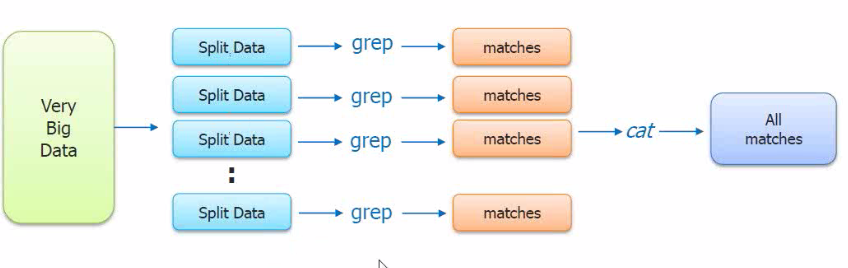


Hadoop is not fit for small files because namenode will get impacted.

**Data replication:**



This is the usecase of tranditional way of extracting data. Data would be very big and first chunk it by number of line then process it parallelly.



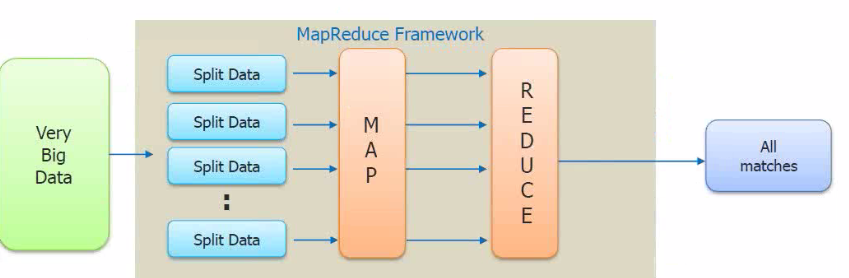
Disadvantage:

- Splitting the data

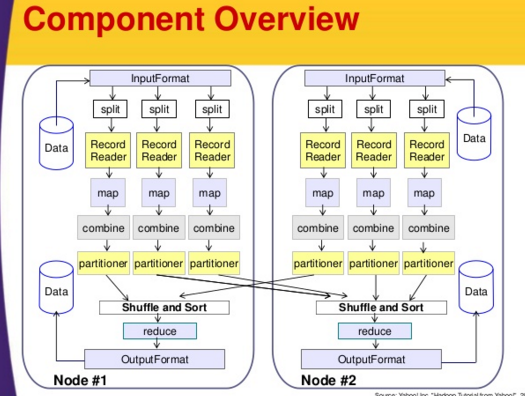
- Complex query

- If any slit job fail then again to process it

- Aggregation



**MR Job flow:**



**InputFormarts**

TextInputFormart

KeyValueInputFormart

NLineInputFormat

SequenceFileInputFormart

**xmls:**

Core - Core-site.xml > Contains Namenode information

HDFS - hdfs-site.xml > Contains files related information - blocks, replication

Yarn - yarn-site.xml > Contains Resource MN, Node MN information

MapReduce - mapred-site.xml > Contains Job tracker and task tracker info.

Hadoop job run:

Hadoop jar <jar filename> <program\_name> <input\_path> <output\_path>

**Map:**

Map Task - Ideally number of Blocks should equal to number of mappers.

1-Get the data from input File

2-Process and implement the logic

3-output (Intermediate output / local disk )

**Reduce: (Hadoop will colect all the maps output into a reducer)**

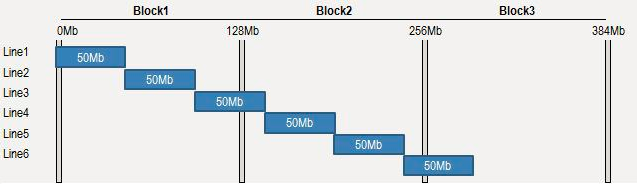
1- Read the data from map task

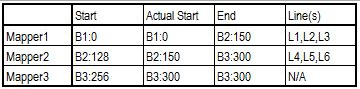
2- Process (Aggregation logic)

3- Output (Final output - hdfs - blocks/replication applies)

**InputSplit Vs BlockSize:**

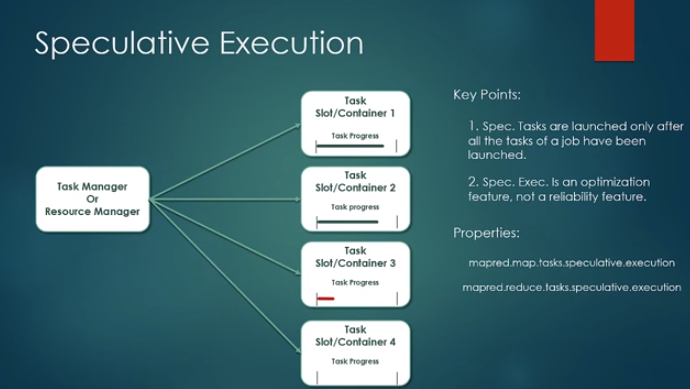
File is composed on 6 lines of 50Mb each



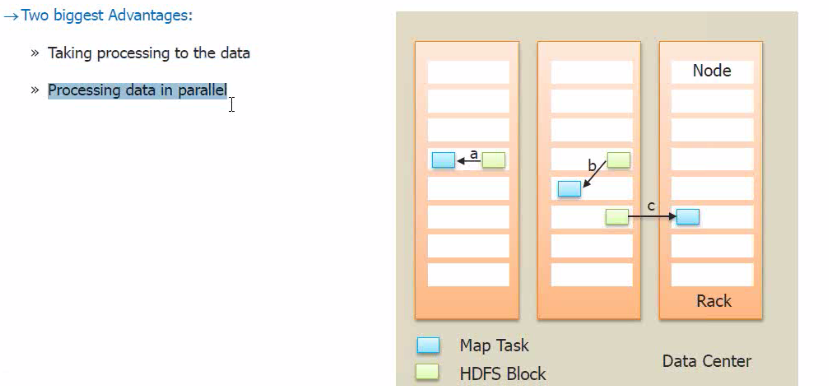


**MAP and Reduce Phase:**

**Speculative Execution:**

****

**Why MapReduce:**

****

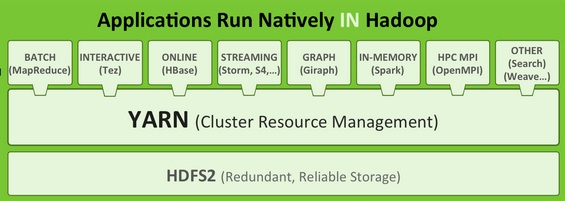
Data locality / Rack locality and Off rack (inter rack network transfer)

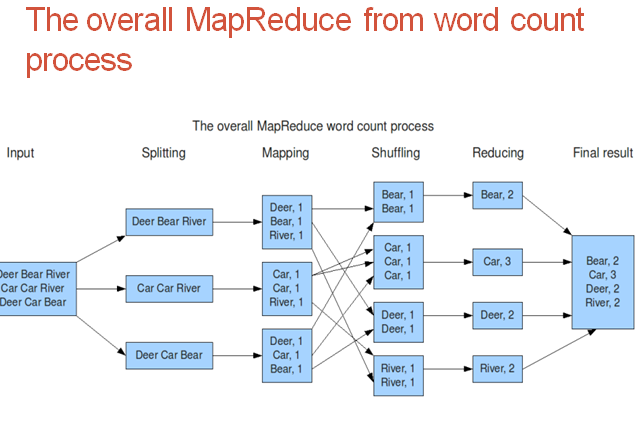
how multinode custer look like:

**Why YARN:**

**Enter Apache Hadoop YARN**

**The fundamental idea of YARN is to split up the two major responsibilities of the JobTracker i.e. resource management and job scheduling/monitoring, into separate daemons: a global ResourceManager and per-application ApplicationMaster (AM).**

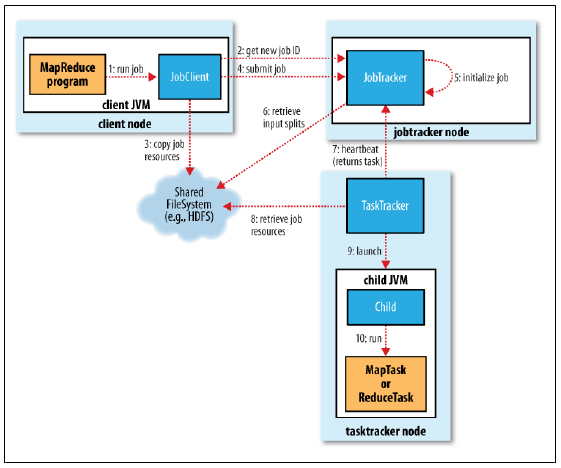
****



**MRv1:**

**It has 4 important components.**

1. The Client
2. The Job Tracker
3. The Task Tracker
4. A Distributed File System HDFS

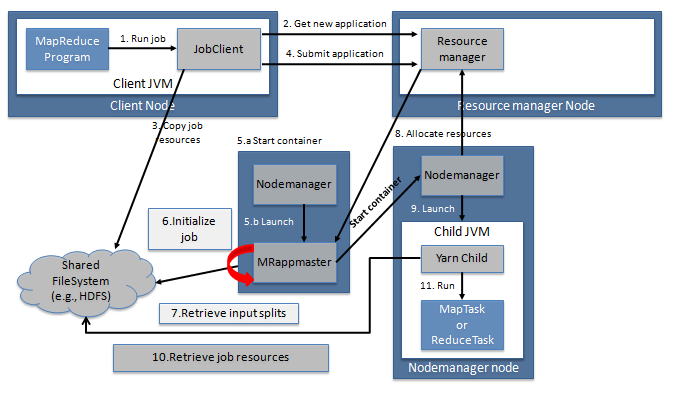


**MRv2:**

YARN meets the scalability shortcomings of “classic” MapReduce by splitting the responsibilities of the jobtracker into separate entities.

Yarn has 5 important concepts:

1. The Client.
2. Resource manager
3. Node Managers
4. MapReduce application master
5. HDFS



**Differences:**

* + **In Mapreduce 1 jobtracker used to launch tasktrackers to run the jobs and also keep a track of progress by each task tracker.**
  + **In Mapreduce 2 (yarn) , this dual functioning of jobtracker is split.**