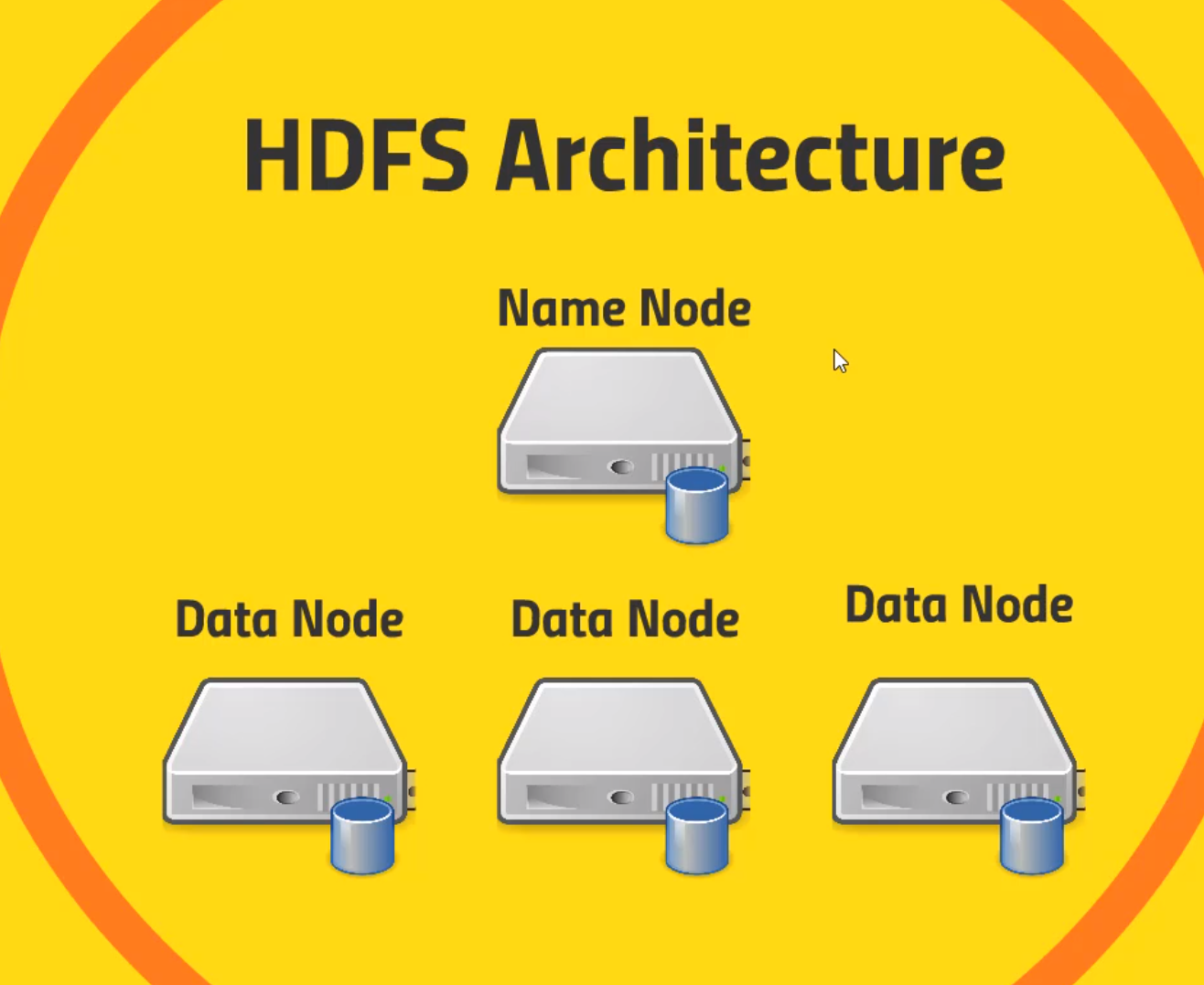
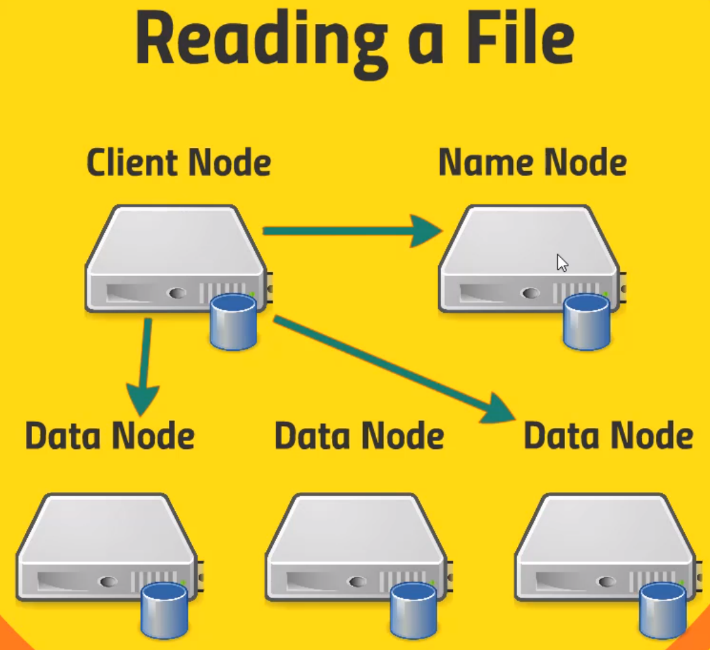
HDFS – Hadoop Distributed File System

Overview – Handles Big Files by breaking them into Blocks. 128 MB per block. Distribution of the processing. Parallel Processing.

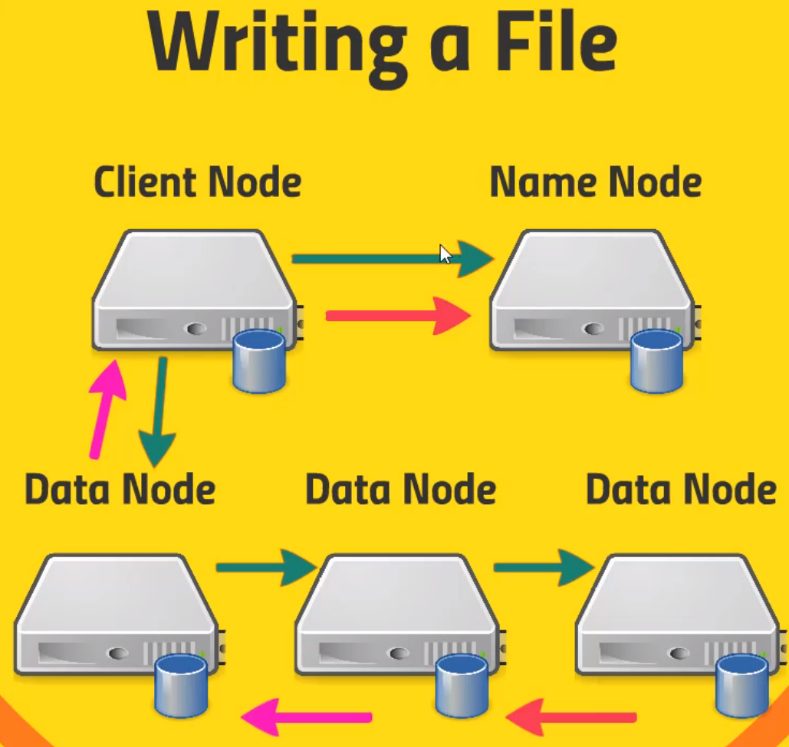
Stored across several commodity computer. Stores more than 1 copy of each block. It has a back up copy of that block.

Name node keeps tracks of where all those blocks live. Knows where to go to find every copy of every block. Edit logs.

Data Node – What store each block of each file. Client interact

Reading a File

Client will know where to get those files. Talk to Name Node and which are the most efficient data node to reach.

Writing a file

Data Nodes will talk to each other for replication of the data. Name Node keeps track.

Client Node talk to Data Node. And DataNodes interact with each other for Replication and storage.

What if a name node fails? Given that one name node is active at a given time.

Name Node Resilience

Back up Metadata – Namenode writes to local disk and NFS

Secondary namenode – Maintain merged copy from edit log you can restore from.

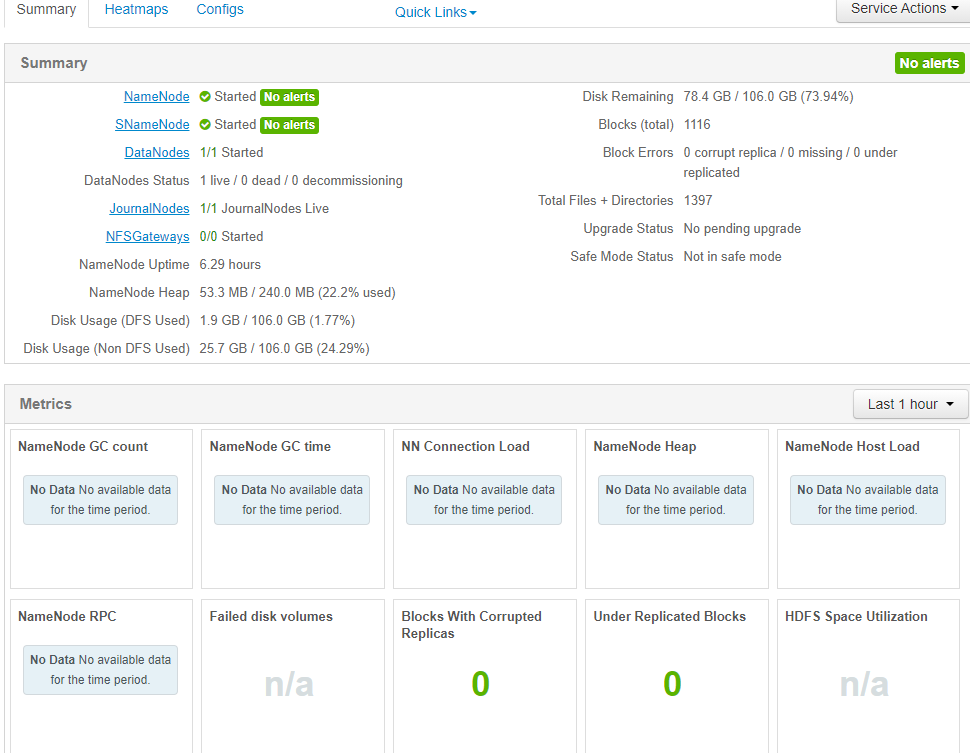
HDFS Federation – Each namenode manages a specific namespace volume. Different subdirectories, we can have separate namenodes. Spread out a lot of name node.

HDFS High Availability

* Hot standby namenode using shared edit log
* Zookeeper tracks active namenode
* Uses extreme measures to ensure only one namenode is used at a time.

Using HDFS

* UI (Ambari)
* Command-Line Interface
* HTTP/HDFS proxies (proxy server between HDFS machine)
* Java Interface (Application development)
* NFS Gateway (Network File System, Linux, Mounting remote file system on a server)

Click on Files View, Upload the data, Open, Rename, You can Concat them together. Start with u.data and concat with u.item. and delete to get rid of it. HTTP to HDFS Interface.

**PUTTY**

Using Command Line. Terminal to connect to. Use Putty. Download the Executable File. Run the executable file and connect to the port [maria\_dev@127.0.0.1](mailto:maria_dev@127.0.0.1), save it as HDP and Open it. Type in maria\_dev as password and you are in.

Type in ‘’hadoop fs -ls’’

‘’hadoop fs -mkdir ml-100k’’ to make a directory

Get some data in the local file directory by using ‘’wget’’

Ls to list

ls -la

Upload into Hadoop : ‘hadoop fs – copyFromLocal u.data ml-100k/u.data’

See files in directory “Hadoop fs -ls ml-100k”

Remove files : ‘hadoop fs -rm ml-100k/u.data’

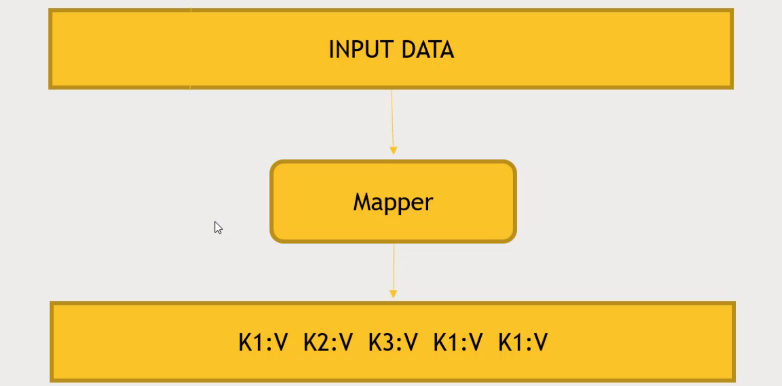
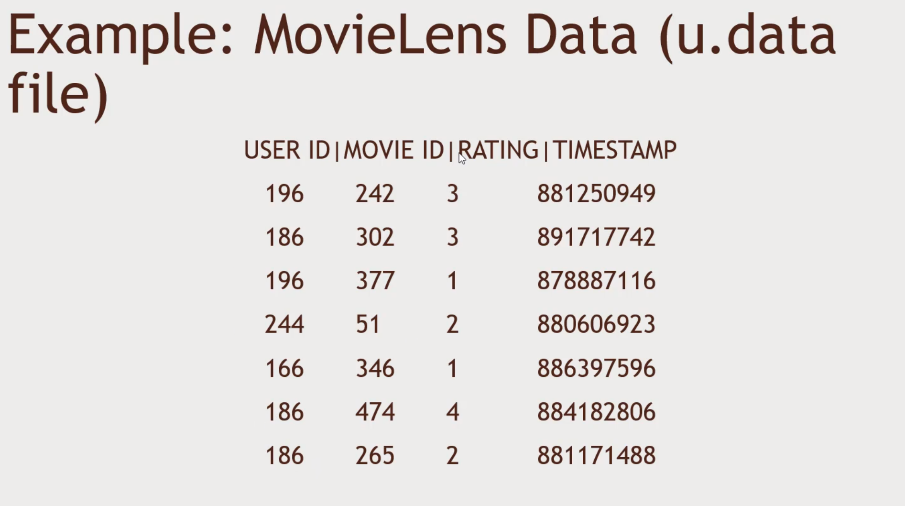
Remove directory: ‘hadoop fs -rmdir ml-100k’

Hadoop fs and Enter see what you can do in the Linux environment.

**MapReduce**

* Distributes the processing of data on your cluster
* Divides your data up into partitions that are Mapped(transformed) and Reduced(aggregated) by mapper and reducer functions you define.
* Resilient to failure – an application master monitors your mappers and reducers on each partition.

**Mapper converts raw source data into key/value pairs**

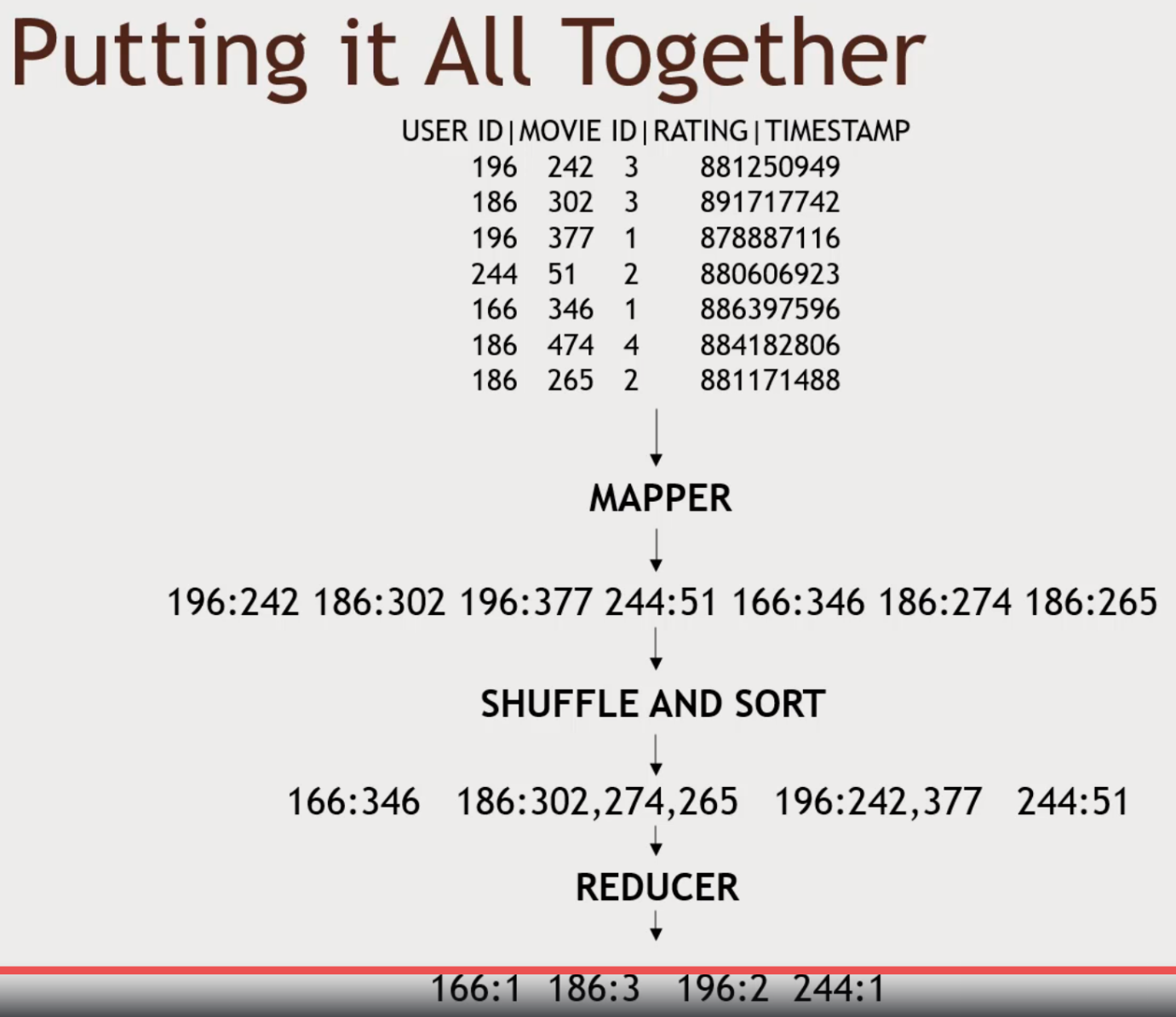
Key – user id, Values are number of movies

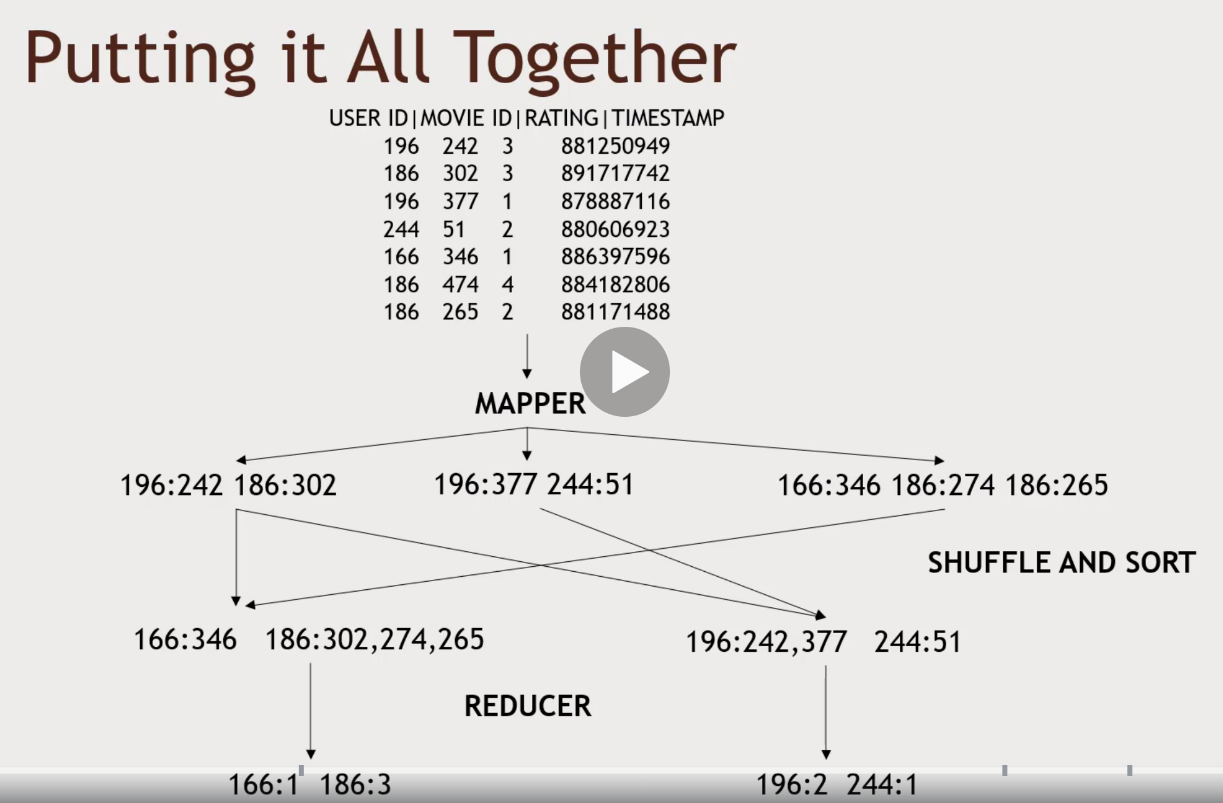
Mapper will reduce the User ID and Movie ID, 196:242, 186:302

Mapper is to extract and Organize what we care about, a bunch of key-value pairs.

MapReduce Sorts and Groups the Mapped Data (Shuffle and Sort). Aggregate all together so that you have a Unique Key.

Reducer processes each Key’s Values. We can reduce by len(movies) unique key and len(movies) combining how many values are associated with each key(user\_id)

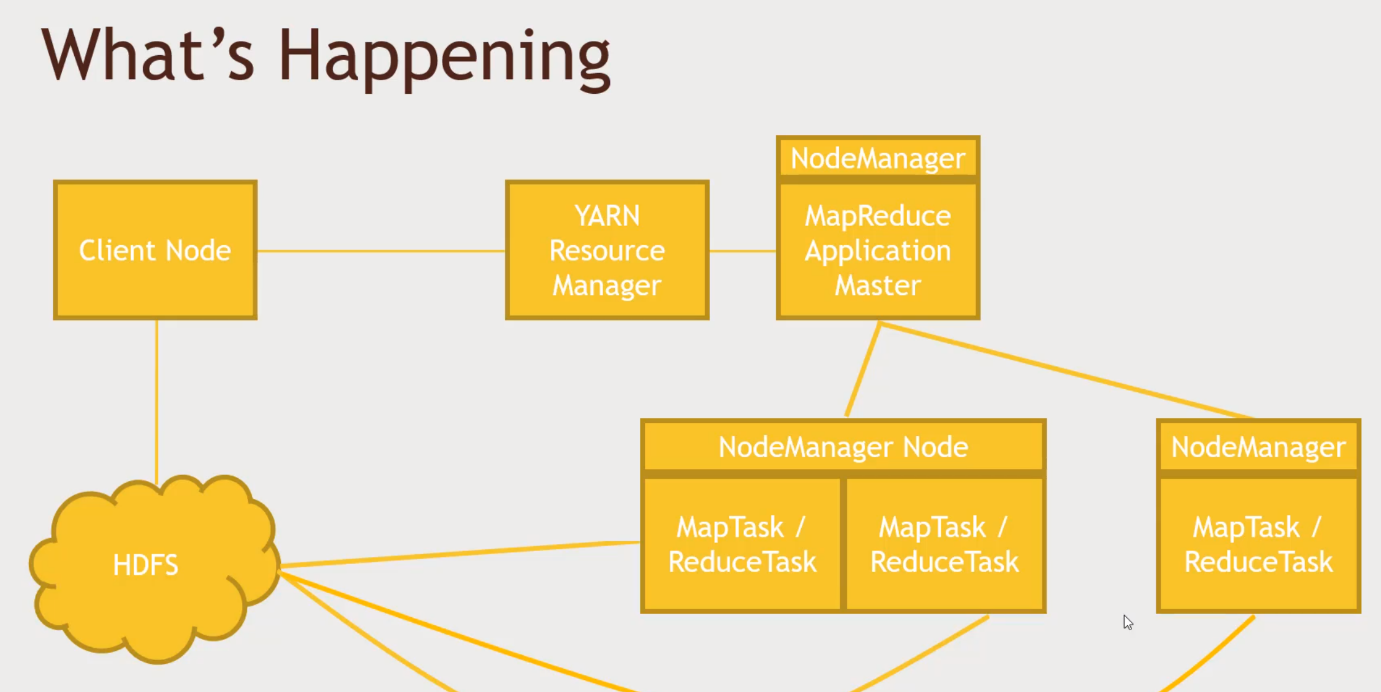
Actual Movies that are rated don’t matter, only care about the count.

MapReduce Distributes Processing

MapReduce map the data into 3 Nodes, for mapping, carve up your data into different partitions. Keep track of when its all done.

Shuffle and Sort Operation, get same information of same keys on multiple nodes. Aggregated together. MapReduce does this for you. Series of Merged Sort for all those information. Things getting shuffled around and sorted.

Sent to different reducers which are run in parallel which has a set of keys.

YARN – What nodes are available and etc.

Copy to HDFS.

MapReduce Application Master (Node Manager) – Responsible for keeping an eye on all the tasks. Talking to HDFS to receive the data that needs to be processed.

Client Node -> Resource manager -> Application Master -> Node manager individual PC

When your resources manager, where to run the map reduce, need to be close to where you input data is. Will be running on the same machine if its possible or it is as close to the network as possible.

Have multiple tasks running on the multiple computers.

Mappers and Reducers

MapReduce is natively Java

Streaming allows interfacing to other languages (ie Python).

Handling Failure

* Application Master monitors workers tasks for errors or hanging
  + Restarts as needed
  + Preferably on a different node
* What if the application master goes down?
  + YARN can try to restart it (Resource manager) will take over.
* What if an entire Node goes down?
  + This could be the application master
  + The resource manager will try to restart it
* What if the resource manager goes down?
  + Can set up ‘High availability’ (HA) using Zookeeper to have a hot standby

SuperSeeded into HIVE, Spark