Lecture - 3 — Advance DB

- HDFS - Hadoop Distributed file system.

Data sharding -
Data Cordinator - Data catalog.
- Has global picture about whats can take more data and cordinate the data.
- shard Manager -
Each client m/c running one shard managers

HDFS is one way of Adding Data sharding.

To run Distributed Job processing (Hadoop as YARN)
we need two Nodes.
- Master Node — resource mgr
  - task allocation — Allocate m/c to run your task
  - task scheduling
    - To schedule task on
- Excutor Node.

Next session - IntelliJ

3. Hadoop distributed file system.
- HDFS Data sharding

* Book - Hadoop - The definative Guide -

HDFS Data Sharding

Each file is partitioned into blocks which are send to different machines.
Data Cordinator is called namenode
namenode keeps track of which file is send to which location using catalog
To get the catalog we have to use metadata.
keeps the index and namenode updates the tree structure
- location of each block

namenode aka data cordinates
↓
has catalog
↓
keeps the track of data. in metadata.

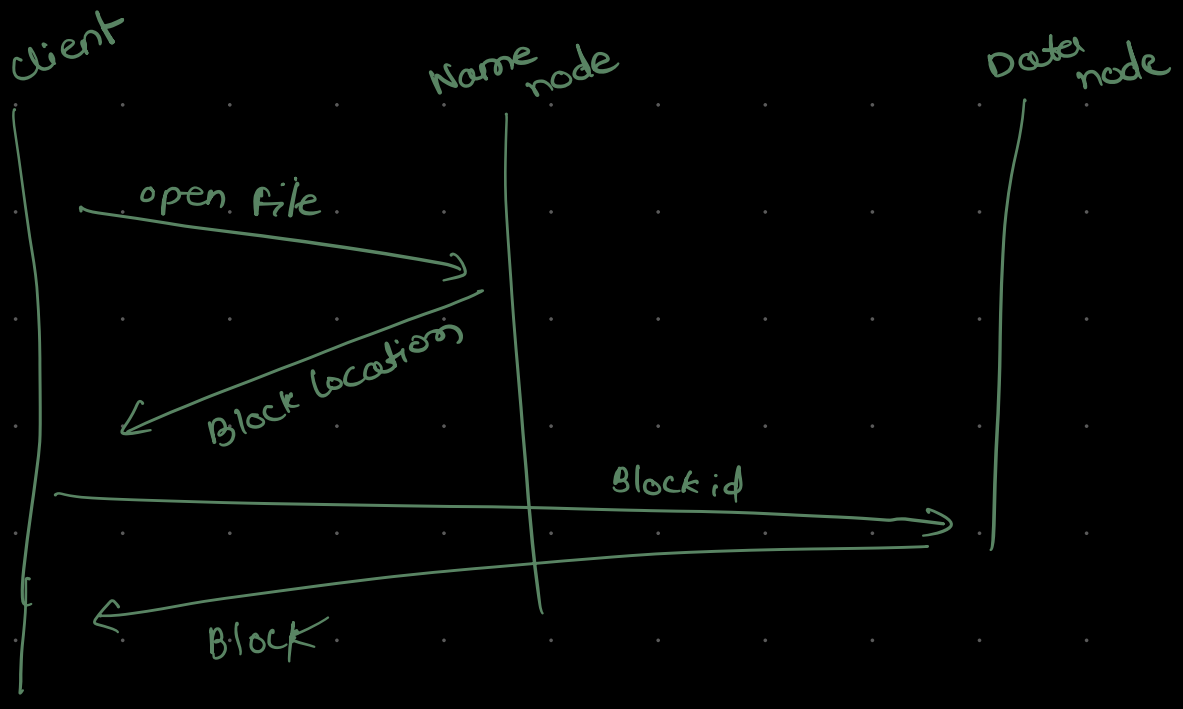
offset - (location of file in file)

- Every block contains
  - Block id
  - offset
  - Data length

– Block Data

In Hadoop block size is fixed.

you keep adding data to block until it's full



---

* namespace img file – it keeps the latest checkpoint – who take the checkpoint?

– Edit log file – is there as audit trail.. ←

– namenode only keeps data in location memory.

→ because it depends communication betn each ~~namenode~~ Datanode

– Data node – not that intelligent as namenode.

- assign blocks to local disk
- Blocks are identified using block id & pool id
- managment of block – read & append opn

– default block size – 128MB – is set higher to get effeciency.

reason →  ← disk

Seek time – machanical movement

- This seek time is always constant as it's machanical movment
- data transfer time is electronic so it depends on data.

to reduce the seek time we use High size data blocks.

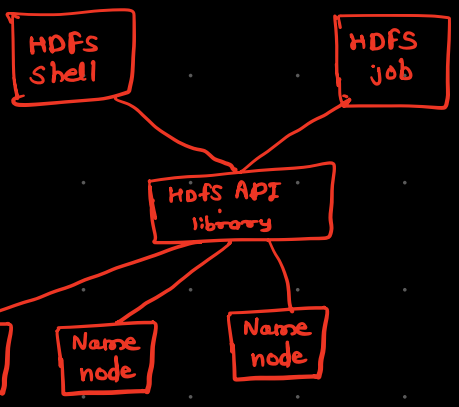
- why not make it even large block size ?
  if we make blocks of higher size data shorading will take the toll as our main objective is to distribute data as much as we can.

— Write once
Read - multiple time
— As blocks [vvvv] when writing cannot be done again but we can read data multiple time.

lecture - 4

Oct. 7th friday

6th week - midterm



you can create your own job using java/ python

We can re-configure the block size less than 128MB — so new block's will be new size
So hadoop will have two diff block size.

also there is way to change the block size!

How to use java to create API

Use cases of file-level API

Coding session - Applicable in done of intellij

→ Create project > Select Java SDK > maven

create configuration object
create file system obj
create file system
test it with adding record
prof. added 15 with loop

Input string / output string

Midterm will have question's on block size, blk id, and other foooo

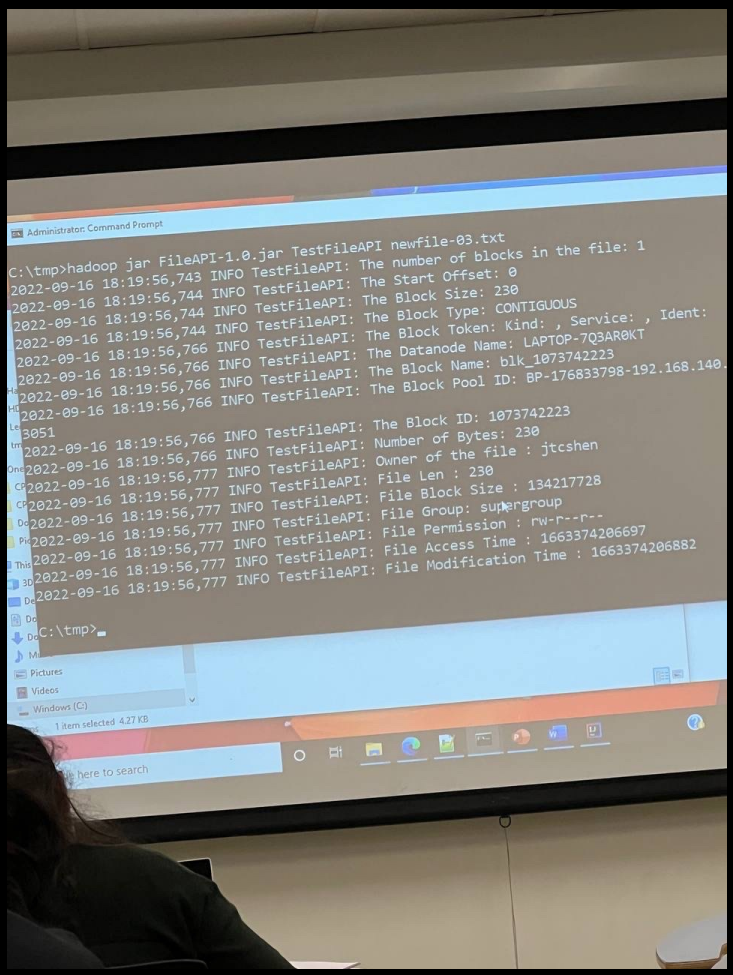
image

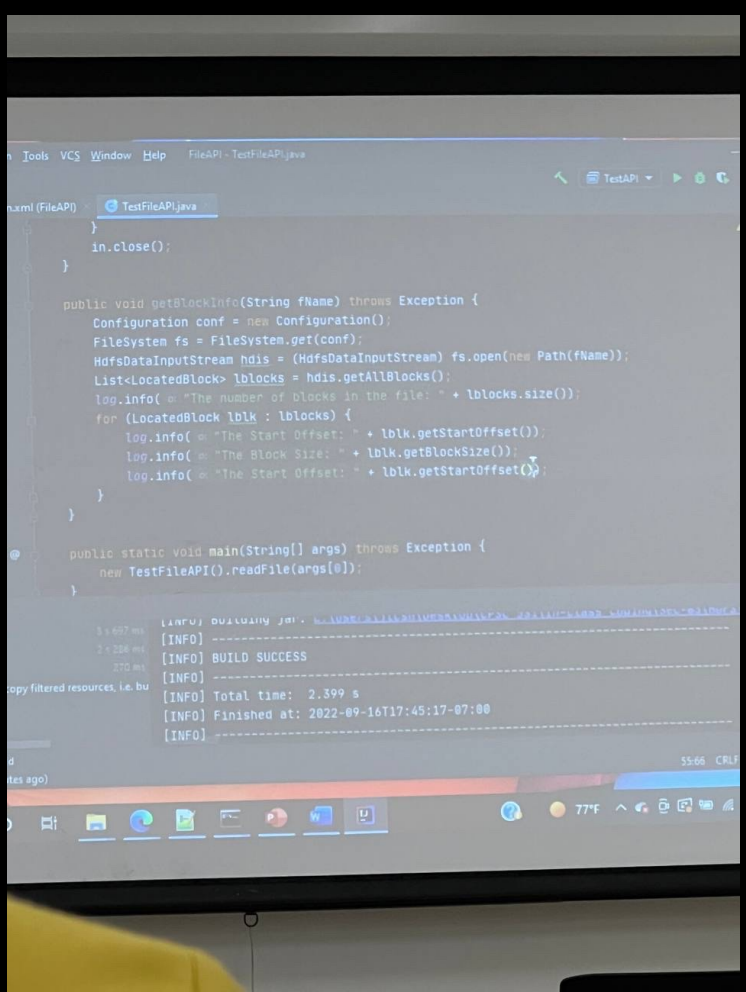
get file status
to get metadata of file

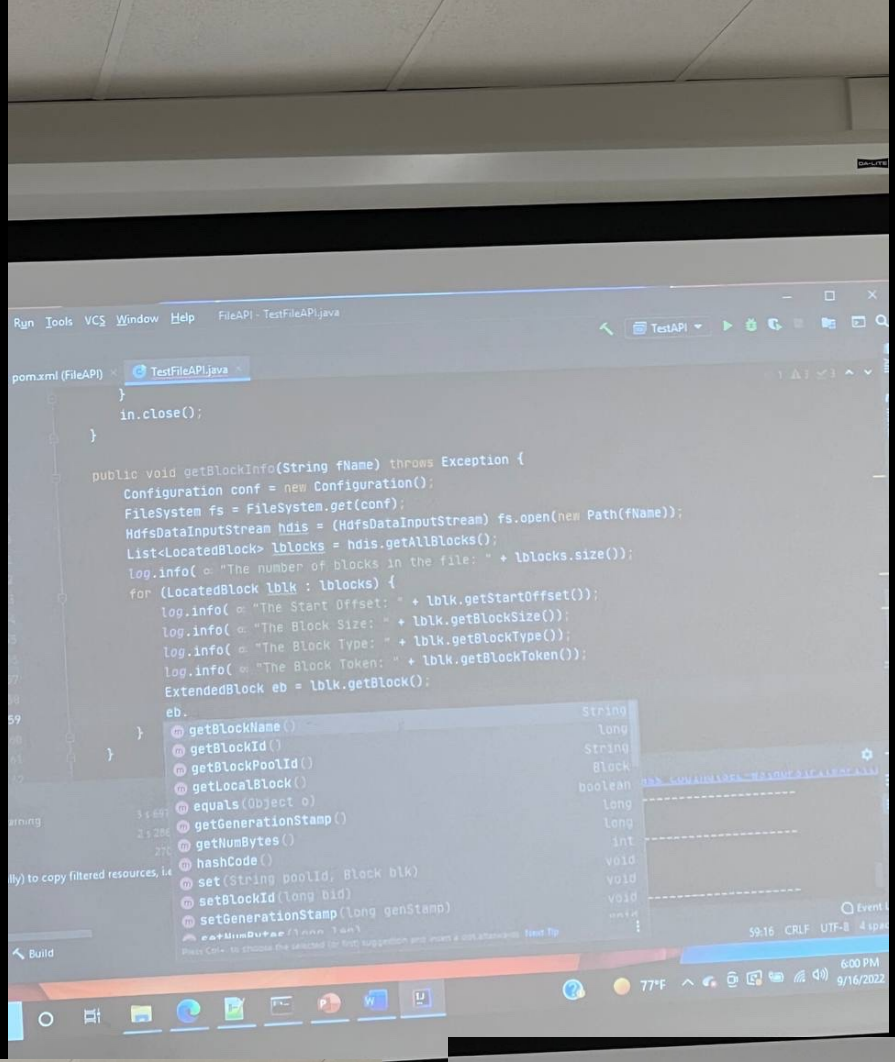
Namenode has
① file metadata
② block location
has following
What is blockpool id
W — Starting offset
— file counts

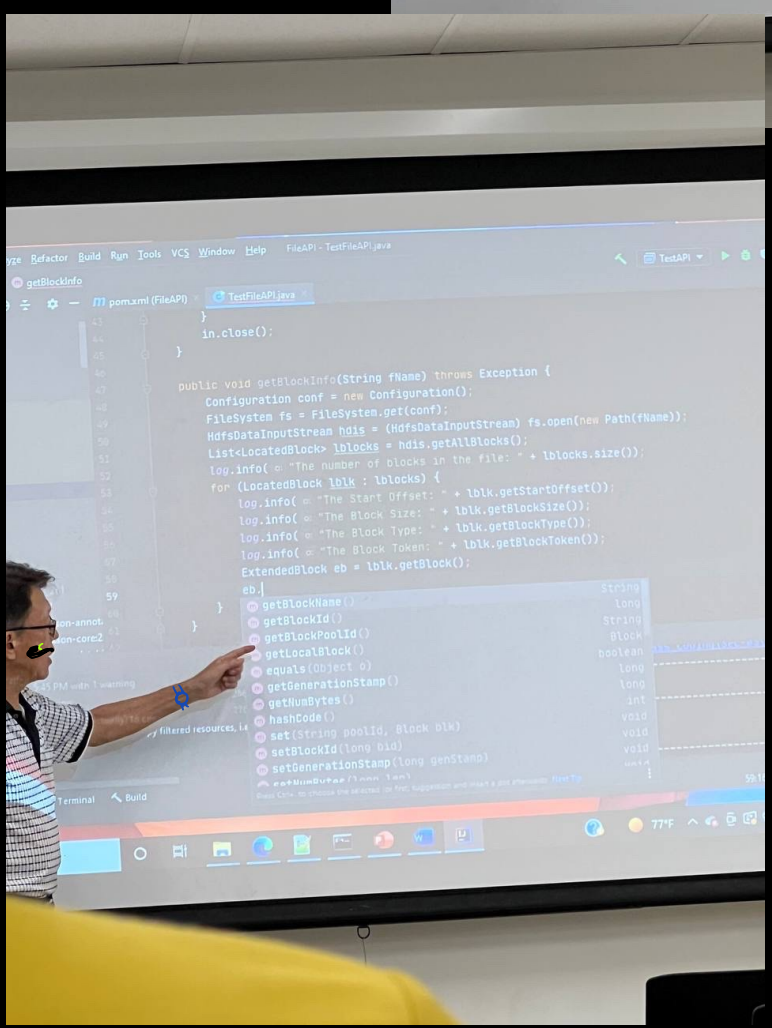
getblock info()
why we need this info?
we can Access the data physically in case of disaster recovery
can decrypt the data with block reader or seek

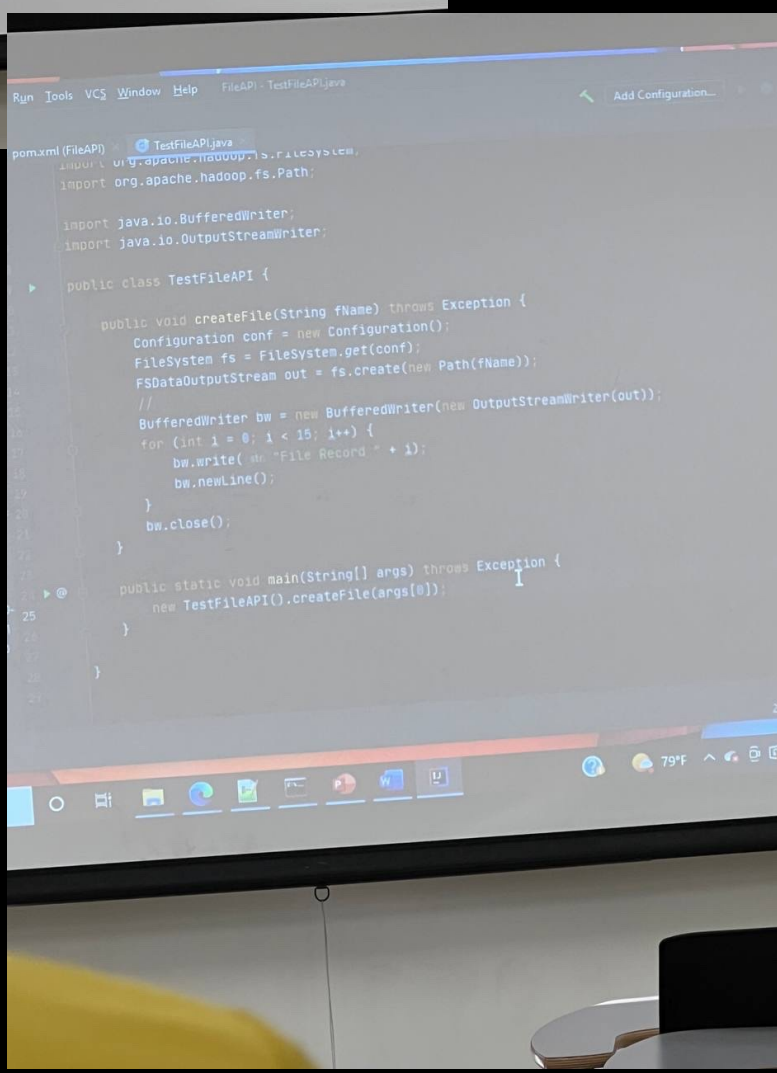
Block reader
can read Seperate data





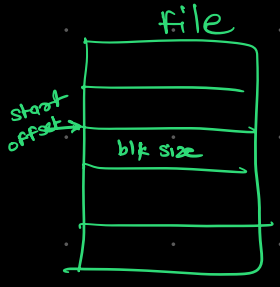

Run Tools VCS Window Help FileAPI - TestFileAPI.java
TestAPI
pom.xml (FileAPI) TestFileAPI.java
in.close();
}
public void getBlockInfo(String fName) throws Exception {
Configuration conf = new Configuration();
FileSystem fs = FileSystem.get(conf);
HdfsDataInputStream hdis = (HdfsDataInputStream) fs.open(new Path(fName));
List<LocatedBlock> lblocks = hdis.getAllBlocks();
log.info(o: "The number of blocks in the file: " + lblocks.size());
for (LocatedBlock lblk : lblocks) {
log.info(o: "The Start Offset: " + lblk.getStartOffset());
log.info(o: "The Block Size: " + lblk.getBlockSize());
log.info(o: "The Block Type: " + lblk.getBlockType());
log.info(o: "The Block Token: " + lblk.getBlockToken());
ExtendedBlock eb = lblk.getBlock();
eb.
getBlockName()
getBlockId()
getBlockPoolId()
getLocalBlock()
equals(Object o)
getGenerationStamp()
getNumBytes()
hashCode()
set(String poolId, Block blk)
setBlockId(long bid)
setGenerationStamp(long genStamp)
Build
77°F
6:00 PM
9/16/2022


getBlockInfo
pom.xml (FileAPI) TestFileAPI.java
in.close();
}
public void getBlockInfo(String fName) throws Exception {
Configuration conf = new Configuration();
FileSystem fs = FileSystem.get(conf);
HdfsDataInputStream hdis = (HdfsDataInputStream) fs.open(new Path(fName));
List<LocatedBlock> lblocks = hdis.getAllBlocks();
log.info(o: "The number of blocks in the file: " + lblocks.size());
for (LocatedBlock lblk : lblocks) {
log.info(o: "The Start Offset: " + lblk.getStartOffset());
log.info(o: "The Block Size: " + lblk.getBlockSize());
log.info(o: "The Block Type: " + lblk.getBlockType());
log.info(o: "The Block Token: " + lblk.getBlockToken());
ExtendedBlock eb = lblk.getBlock();
eb.
getBlockName()
getBlockId()
getBlockPoolId()
getLocalBlock()
equals(Object o)
getGenerationStamp()
getNumBytes()
hashCode()
set(String poolId, Block blk)
setBlockId(long bid)
setGenerationStamp(long genStamp)
Terminal
Build
77°F


Run Tools VCS Window Help FileAPI - TestFileAPI.java
Add Configuration...
pom.xml (FileAPI) TestFileAPI.java
import org.apache.hadoop.fs.Path;
import java.io.BufferedWriter;
import java.io.OutputStreamWriter;
public class TestFileAPI {
public void createFile(String fName) throws Exception {
Configuration conf = new Configuration();
FileSystem fs = FileSystem.get(conf);
FSDataOutputStream out = fs.create(new Path(fName));
//
BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(out));
for (int i = 0; i < 15; i++) {
bw.write(str: "File Record " + i);
bw.newLine();
}
bw.close();
}
public static void main(String[] args) throws Exception {
new TestFileAPI().createFile(args[0]);
}
}
79°F

Midterm – 21 oct

lecture-5



offset

Seek method to look for data

Exercise – How seek works?

In hadoop we can get tree structure of files using:
→ using getmetadata directory

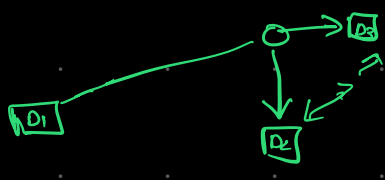
we add code → create jar file for the same

To get the root directory we just add "/" at the end

- When namenode crashes
  - Metadata are not accessible
  - Blocks are not accessible
- Datanode crash
  - partial dataloss

Hadoop created a mechanism to protect from such situations
① Block redudancy
② Namenode redudancy

Data locality – How close data is to you.



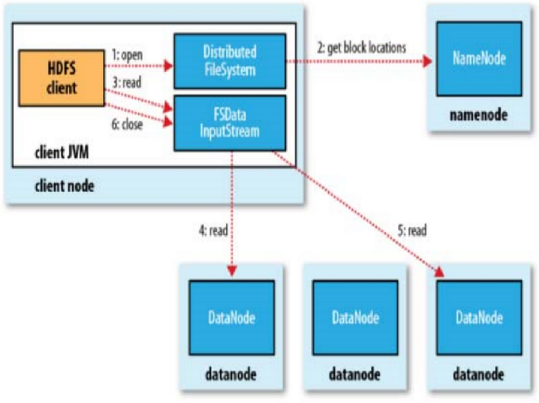
= Low data locality

Data close = Data locality
Data locality can be defined using time to reach the Datanode.

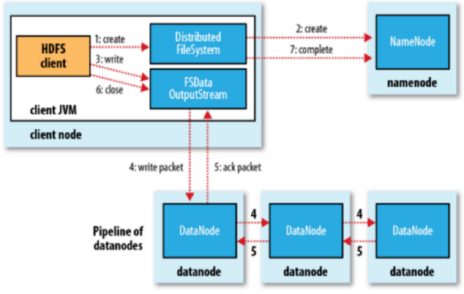
Map reduce has more issue of data locality

↳ Part of Block redudancy.

Implementation of File Read Operati



Implementation of File Write Operation



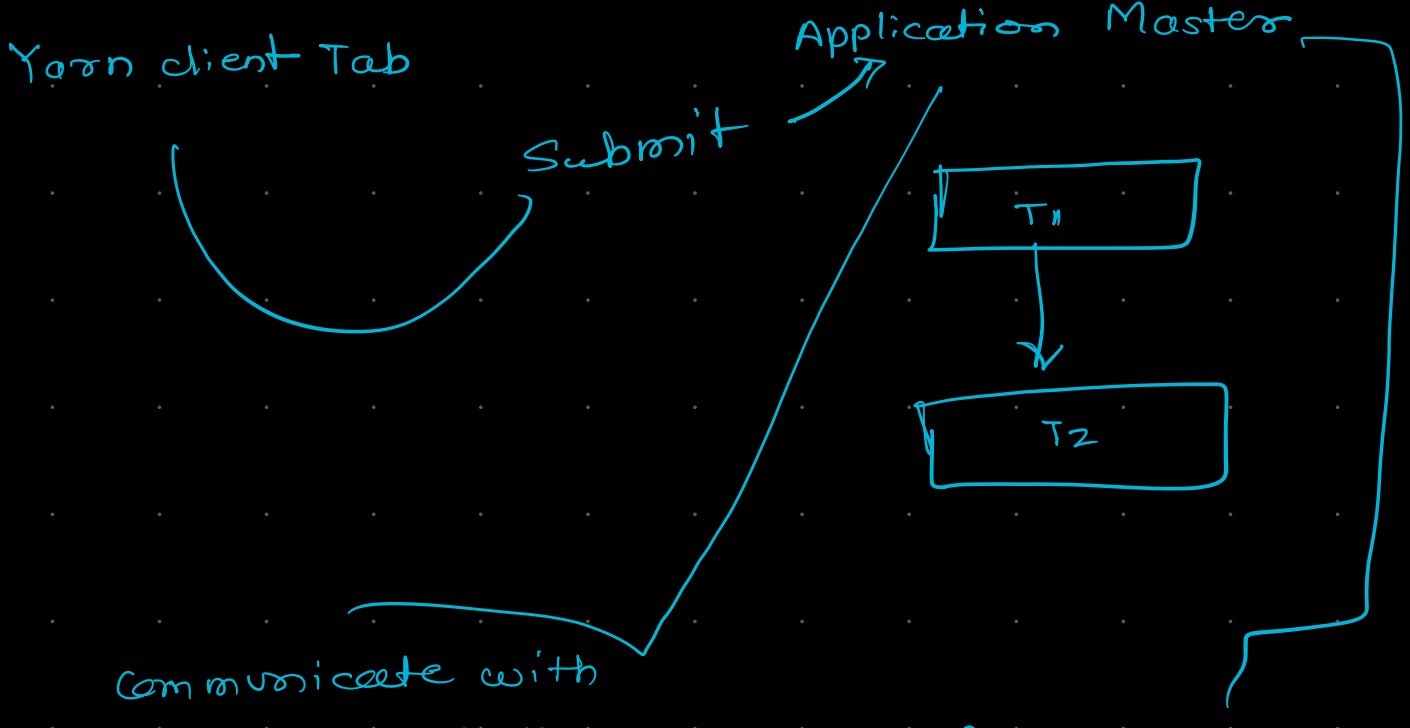
Namenode Redudancy

New chapter

YARN API

Sept 30

YARN



Job scheduling –

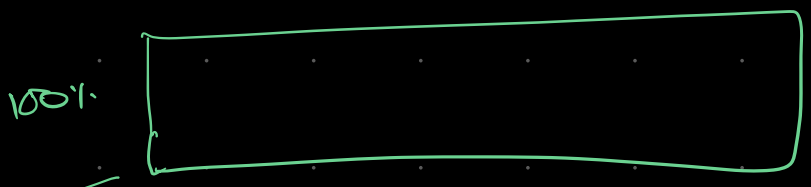
- Scheduling of resources at the resource manager point.

Capacity Scheduling –

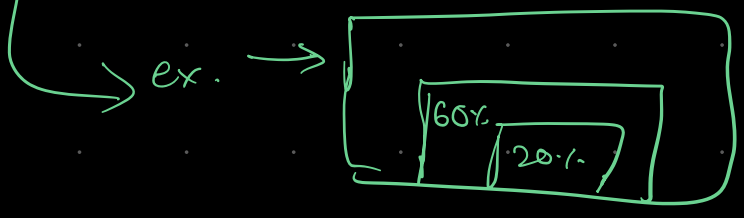
– show running jobs then complete thoes first with priority

–

Dynamic resource allocation



So we YARN scheduling fairness it will not prioritize anyone He will just schedule some part of resource to others.



HDFS

└ redudancy.

— what hapn if failure on node manager?

resource manager has to detect that first & He has to take another container

if task fails Application manager responsibility to detect the error & recover from it.

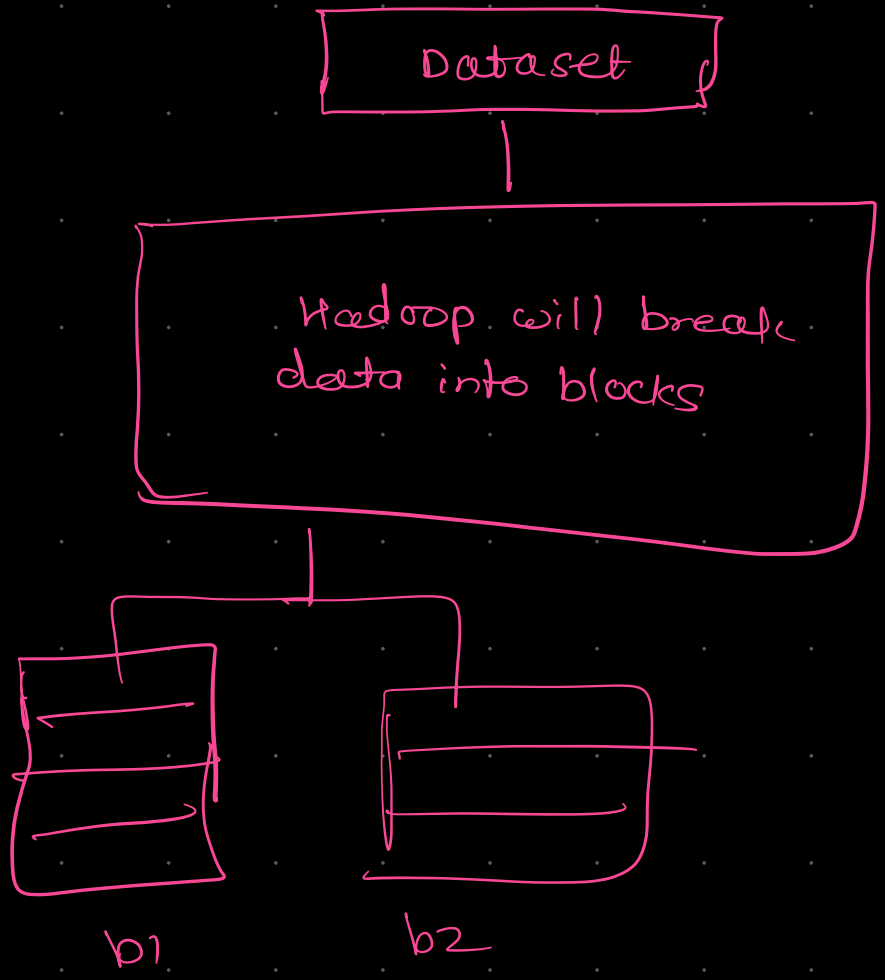
Map reduce

Using YARN

How we develop Appln using map reduce

implementing Map reduce
- why we use map reduce

Group the data and do aggrigate work



to group the data - redistribute the data we get key attribute value

- with map reduce we can do aggrigate operation in parallel foshion

Map reduce major issue -

↳ Redistribution

- Reduce task -

Writable class is depended on

- compare to – mainly used for sorting purpose
- meditator *
- write

— Write a map reduce App
group students using gender
using composite key

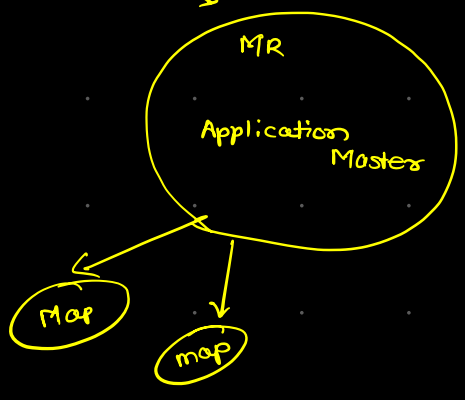
```
select gender, level, AVG(gpa)
from student
group by gender level
```

Apache Spark

difference between MP & Spark

Map reduce

- Write Map Reduce
- Create Map reduce client job
- Build jar
- Hadoop jar < jar file > < Main class >



- Mp uses maps based on need (on request)
- for Spark even before initialization it runs the executors (based on event)
- ex. 10 running but only 4 can be used for the task.
  - underutilized.
  - Spark has to take care of utilization

Spark

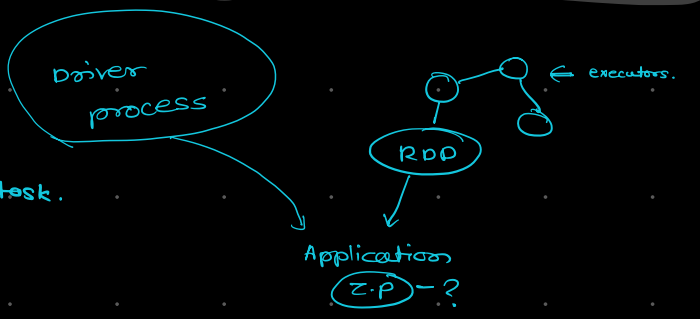
- Write Spark code -
  - Create Application content object
  - Create RDD (Resiliant Distributed dataset)
  - Create job
  - Submit job
- Build the jar file use
- Spark - submit

Driver process

→ Here rather than map executors are there which executes the task.

→ How many executors need to start

Based on formula



Biggest diff RDD is partitioned

IF we run YARN cluster mode then Driver process run AS YARN Appln Master.

client mode then driver process is not relied on YARN Appn in cluster mode it is depended

- for deployment Spark is more flexible.

- On request demand

- Spark runs on memory
- On event

We can assign no. of executors using command as executors start at the start

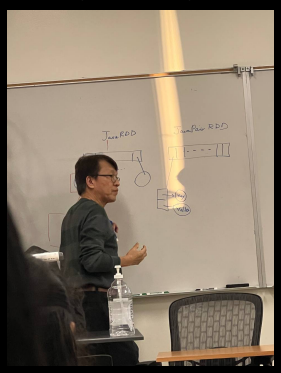
— SO Spark has
  - Core
  - Memory
  - No. of executors

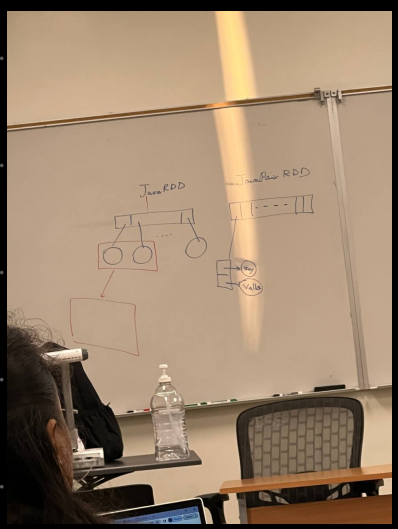
SO based on executors we give how many cores, memory to use but it has to be in limit. We have to make sure to utilize it accordingly.

— As memory is limiting factor for each task
— We can run more cores but each will have less memory.

Another way to allocate Resource → one core per machine!

RDD programming





Java RDD

Java Pair RDD
- key
- value

Spark - SQL Spark

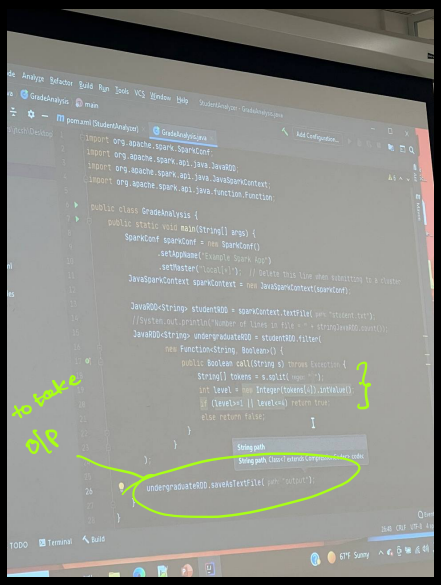
- In Spark we create multiple key value pair in order to not use same key everywhere.
  ↓
- Hands on sessions!
  Spark folder Demo!
  ↓
  intellij
  ↓
  copy dependencies! (Spark / core)
  ↓
  invalide & restart to download dependencies
  ↓

In this example we are filtering student level from 1 to 4

level = tokens[4] but have to convert it to integer object.

We use if else & return Boolen we don't use while loop here.

•



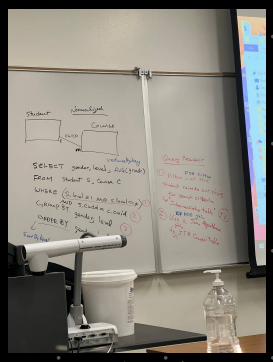
We get two output because of partitions in Spark.

We can also manually say How many partitions We need.
It can be seen in professors code!

LOL — In Spark 0 partition means o/p will be in 1.

What We are trying Here is get data filter it
Then we filter it then we join (See example Below)

—



To define key, value pair in Java

JavapairRDD < String, String>