Speculation Execution Model:

JT assigns task to TT, if TT unable to respond within the given time(Heart-beat time) then JT assigns the task to another TT. This is called Speculative Execution.

Speculative Execution comes into picture only when TT is unable to respond within heart-beat time.

If TT is responding and saying that the task is about to be failed, then JT asks to re-try for 4 times, after 4 attempts, the job is named as "failed"

Default re-try attempts= 4

But we can increase /decrease this re-try attempts

for that---------->goto hadoop/etc/conf/mapred-site.xml

<configuration>

<property>

<name>mapred.map.max.attempts</name>

<value> 6 </value>

</property>

</configuration>

---------------------------------------------------------------------------------------------------------------------------------------------------------------

Speculative Job:

Speculative Task:

S1------->B1-------->m1---------->92%

S2------>B2--------->m2---------->4%

S3------>B3--------->m3----------->85%

S4------>B1

S5------>B2--------->m2 is initiated--------->

S6------>B3

If Multiple mappers(m1,m2,m3) are running parallelly , then if any one mapper(m2) is performing badly then JT initates the same mapper task(m2)in background in anothe slave(s5) containing blockB2 .

Among S2 and S5 ,which one executes first ,that o/p is collected and the other task will be terminated.

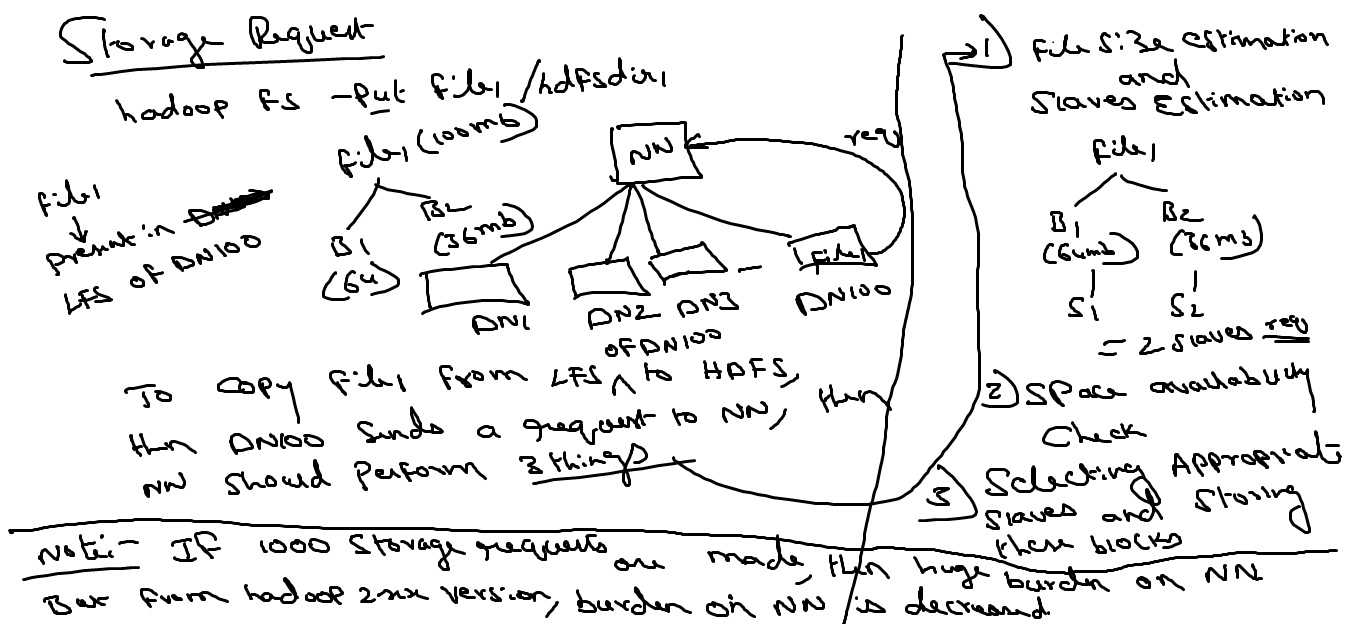
The Job that is using this speculation is called Speculative Job.

The Task(m2) which is duplicated in the background is called Speculative Task.

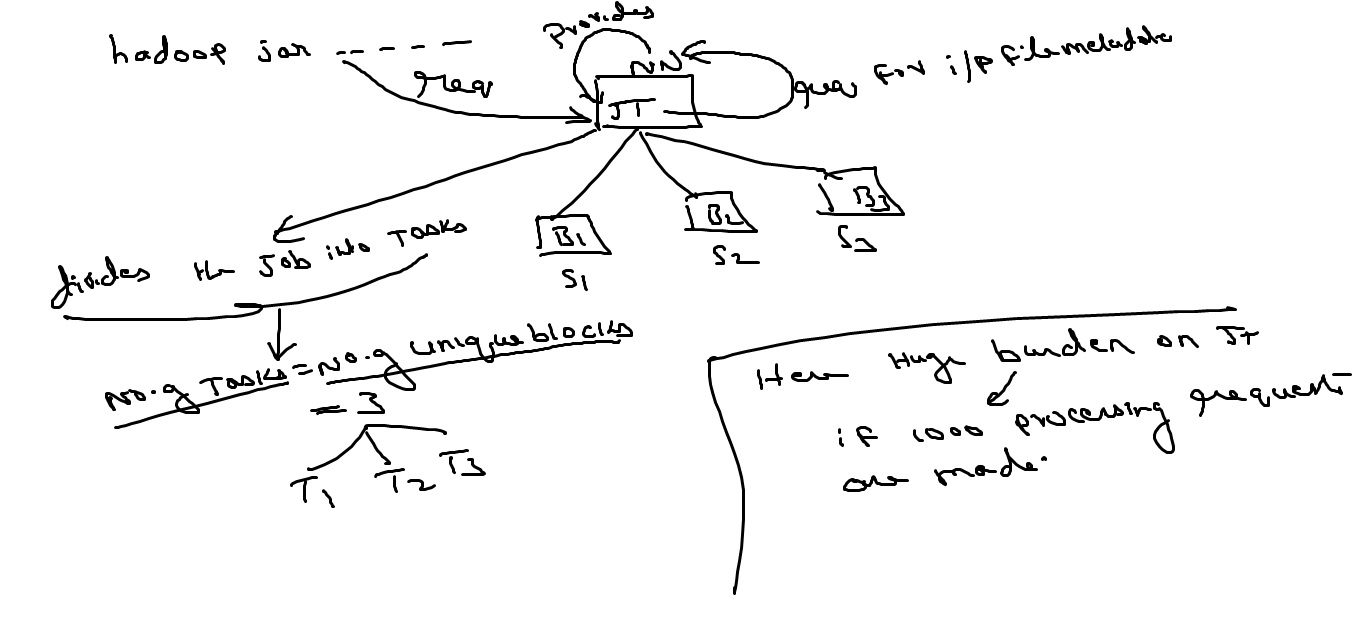
-------------------------------------------------------------------------------------------------------------------------------------------------------------------

Submitting Storage and Processing Requests:

hadoop fs -put file1 /hdfsdir1-------------------->storage request (ref diagram yarn2.jpg)



Processing request



-----------------------------------------------------------------------------------------------------------------------------------------------------------------

TT Responsibilities :

1)Executing the tasks given by JT.

2)once Task execution is completed,TT sends ACK to JT

3)TT sends heart-beats to JT for every 3secs.

If the heart-beats are missing b/w JT and TT then JT thinks that TT is down and allocates the same task to another TT.

-------------------------------------------------------------------------------------------------------------------------------------------------------------------

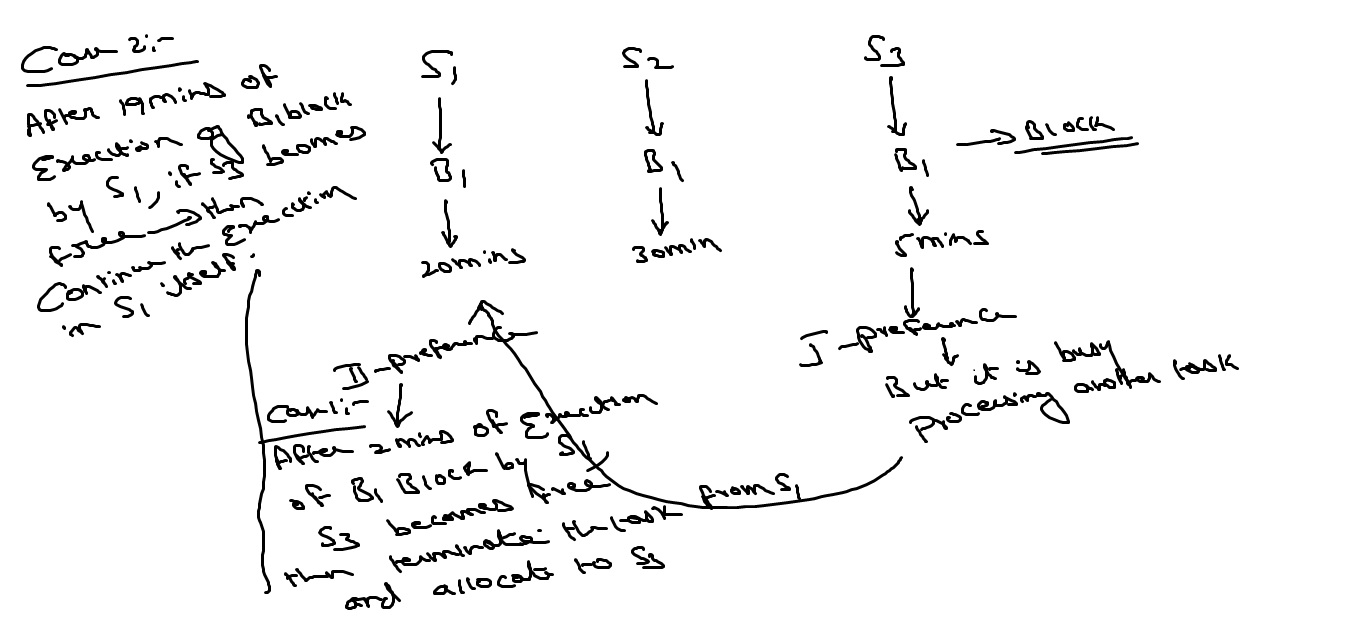
Q)what are the 3 situations where JT reassigns the task to another slave.

1)If Active slave is down( Fault Tolerence)

2)If any slave is performing badly

3)If the Expected slave becomes free after some time.

JT can estimate the time taken by each slave to process a task based on the slave's configuration (Ref diagram Yarn3.jpg)



-----------------------------------------------------------------------------------------------------------------------------------------------------------------

Q) what happens if JT is down and Name node is up --------->All processing operations are down (partial cluster sevices are down) but still storage operations can be performd

-New Jobs cannot be submitted

-Running jobs will be terminated.

-Queued jobs also will be terminated.

---------------------------------------------------------------------------------------------------------------------------------------------------------------

Q)what happens if NN is down---------->All storage and processing Operations are down.

here why processing operations are down---------> bcoz JT to process requires i/p file metadata which is available with NN but NN is down ,so processing operations are also down.

-New Jobs cannot be submitted

-Running jobs will be terminated.

-Queued jobs also will be terminated.

-----------------------------------------------------------------------------------------------------------------------------------------------------------------

storage New Jobs Running Jobs Queued

jobs

1)JT - Down Yes No No No

NN - Up

2)NN - Down No No No No

JT - Up

in First case partial cluster services are down

in second case entire cluster services are down

so here SPOF(single-point-of-failure) is at NameNode i.e if NN is down then everything is down ,i.e entire cluster services are down

But from Hadoop 2.x version ,It is SPOF-free Architecture

--------------------------------------------------------------------------------------------------------------------------------------------------------------------

YARN (Yet Another Resource Negotiator)

Before hadoop 2.x version,SPOF is at NameNode i.e if Namenode is down then everything is down

from Hadoop 2.x version, we got a new feature called HA(High Availability) i.e we can have

multiple Name Nodes

Before Hadoop 2.x ----------------------->NN down----------->SNN

from HAdoop 2.x--------------------------> NN1 NN2 NN3

Active passive passive

if down --------> Active Passive

if down ----------->Active

At a point of time,only one Namenode will be active,the others will be passive.

PAssive Namenodes will keep collecting data from active Namenode.

Suddendly if any Active NAmenode is down,then any of the passive Namenode becomes active

--------------------------------------------------------------------------------------------------------------------------------------

Q) How NN2 comes to know that NN1 is down

Ans) based on heartbeat signals .

--------------------------------------------------------------------------------------------------------------------------------------

Differences B/w

Hadoop 0.xx/1.xx Hadoop 2.xx

------------------------ --------------------------------------------

1) Here SNN is not 100% backup node. , 1)Here we have multiple Name Nodes

Here SNN cannot write or update If one NN1is down, then immediately

into fsimage and edits. the other NN2 will become active.

Here NN2 is 100% backup node for NN1

Here NN2 can write or update into

fsimage and edits.

Here shared metadata

(Ref diagram yarn4.jpg)

2)Here no metadata sharing 2)Here shared Metadata

3)Here No HA(High Availability) 3)Here we have High Availability(HA)

4)SPOF is at NameNode 4)SPOF-free Architecture

4)Default Block size=64mb 4)Default Blocksize=128mb

5)Storage Request: 5)Storage Request:

hadoop fs -command name hdfs dfs -command name

ex:hadoop fs -put file1 /hdfsdir1 ex:hdfs dfs -put file1 /hdfsdir1

6)Processing Request: 6)Processing Request:

hadoop jar jarfilename \ yarn jar jarfilename \

package.drivername \ package.drivername \

inputpath \ inputpath \

outputpath outputpath

7)Much burden on JobTracker 7)Here no burden bcoz , JT divided into bcoz should take care of 2 parts

2 things 1)Resource Manager ------>for

Resource mangement

1)Resource Management

2)Task monitoring 2)Application Master------->for Task

monitoring

NodeManager------------>nothing

but Task Tracker

8)stating services: 8)Starting services:

start-dfs.sh start-dfs.sh

start-mapred.sh start-yarn.sh

-----------------------------------------------------------------------------------------------------------------------------------------------------------------

YARN Components:

1)Resource Manager-------------------------------->

2)Application Manager

3)Resource Scheduler -------------------->all these 4 components runs at Master level

4)Job History Server--------------------------------->

5)Application Master-------------------------------->

6)Node Manager ------------>all these 3 components runs at slave level.

7)Container ---------------------------------->

-----------------------------------------------------------------------------------------------------------------------------------------------------------

1)Resource Manager : Responsible for Resource Management

It has got 2 sub-services

i)Application Manager

ii)Resource Scheduler

2)Job History server : maintains track of history of all jobs.

3)Application Master : Responsible for task monitoring i.e responsible for dividing job into tasks

for 1 job-------->one AM is assigned

for 3 jobs------->3 AMs

for 10jobs------->10 AMs

so here no burden

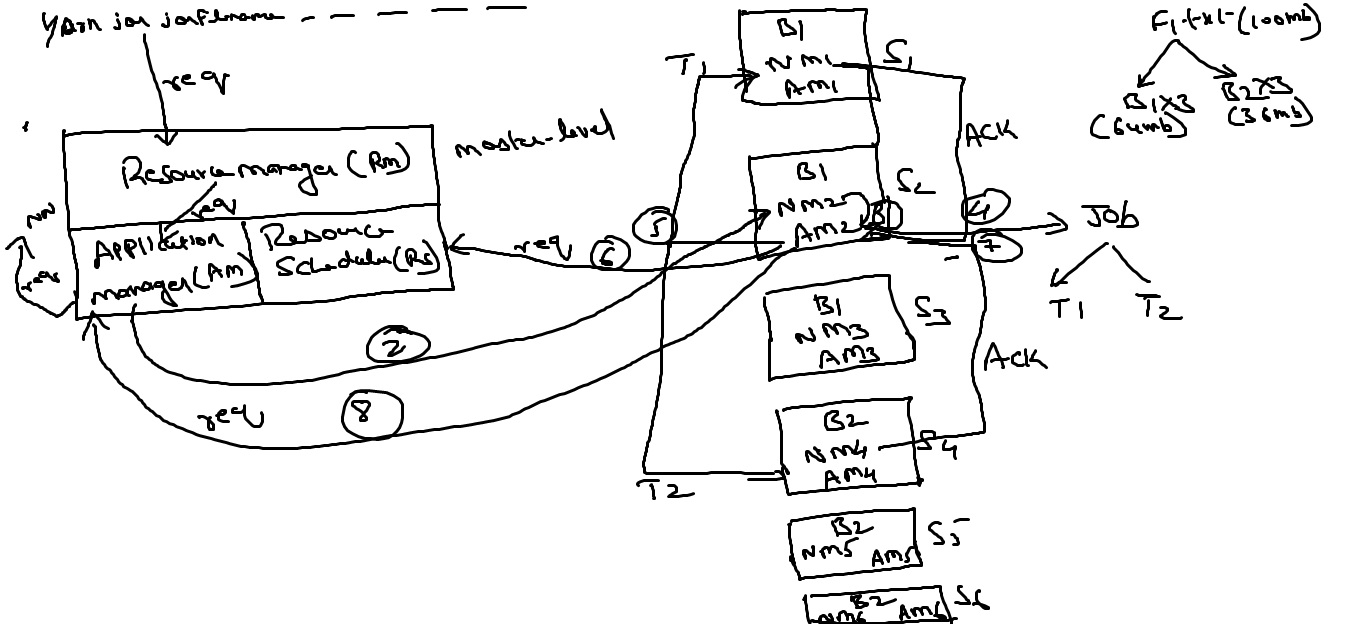
4)Node Manager : Responsible for execution of tasks

similar to task tracker

5)Container :set of Hardware resources required for executing a task.

------------------------------------------------------------------------------------------------------------------------------------------------------

YARN Architecture : (Ref diagram yarn5.jpg)



step1 :whenever any processing request is made, then the requested is forwarded to Resource Manager(RM),then RM forwards the req to Application Manager(AM), then AM requests NN for i/p file metadata.

NN supplies the metada.

step 2: AM selects the slave(s2) which is near, now communication b/w AM and Nodemanager(NM2) is established

step 3:AM asks NM2 to start Application Master(Am2), now communication b/w AM and Am2 is established and ApplicationManager(AM) will hand over the entire job to AM2

step 4: ApplicationMAster(Am2) divides the job into tasks(T1,T2) based on blocks(B1,B2).

Note : To execute thse 2 tasks(T1,T2), 2 containers(c1,c2).

step 5:ApplicationMAster(Am2) selects appropriate Nodemanagers(NM1,NM4) for executing the 2 tasks(T1,T2)

step 6:ApplicationMaster(Am2) sends a request to ResourceScheduler(RS) to grant 2 containers(C1,C2) for executing the 2 tasks(T1,T2) by NodeManagers --->NM1 and NM4.

Resource Schedulers grants the required containers

step 7: ApplicationMAster(Am2) asks NM1 and Nm4 to start the containers(c1,c2).

After starting the containers ,NM1 and NM4 sends ACK to AM2 saying that the containers are started

step 8: ApplicationMAster(Am2) sends a req to Application Manager(AM) asking for execution of tasks(T1,T2)

once AM approves then Am2 orders NM1 and Nm4 to to start the execution of tasks

step 9: once tasks(T1,T2) executions are completed then NM1 and NM4 sends ACK to Am2 and

Am2 sends ACK to ApplicationManager(AM) and AM sends ACK to ResourceManager(RM).

step 10 : once Entire job execution completes then the containers(C1,C2) are deallocated by ResourceScheduler(RS).

If one more job, i.e job2, then ApplicationManager hand over that job to another ApplicationMAster(Am3)

If 5 jobs------------->means AM hand over these 5 jobs to 5 ApplicationMasters such as Am1,Am2,Am3,Am4,Am5

so here the burden is decreased on ApplicationManager.

--------------------------------------------------------------------------------------------------------------------------------------------------------------

Fault-Tolerence:

Case 1: If Container(C1) is down

then ApplicationMaster(Am2) asks NM1 to re-start the container(C1) if still problem then Am2 requests

RS for a another container

Case 2: If NodeManager(NM1) is down

then ApplicationMaster(Am2) assigns the task T1 to another NodeManager(NM3) containing block B1 in that slave.

Case 3: If ApplicationMaster(Am2) is down

then all NodeManagers(NM1,NM4) will report or sends ACK to ApplicationManager(AM) directly.

Case 4:If ApplicationManager(AM) is down

then ApplicationMaster(Am2) will report or sends ACK to ResourceManager(RM)

Case 5: If ResourceScheduler(RS) is down

then ApplicationMaster(Am2) will report or sends ACK to ResourceManager(RM)

Case 6:If Resource Manager is down

then we got HighAvailabilty i.e NN2 becomes active.