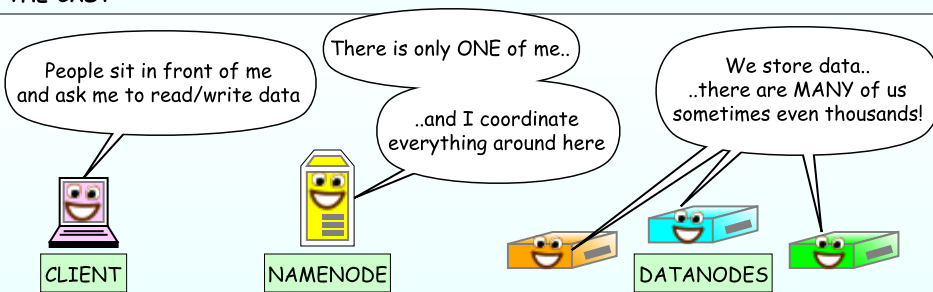


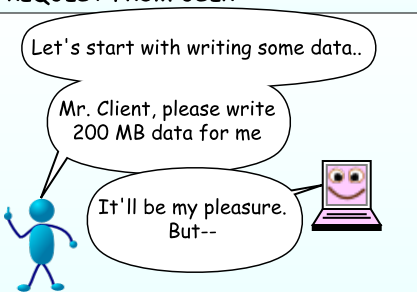
HADOOP DISTRIBUTED FILE SYSTEM (HDFS)

THE CAST

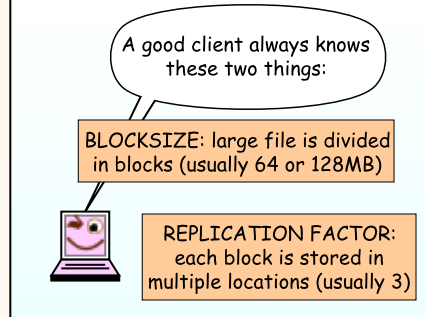
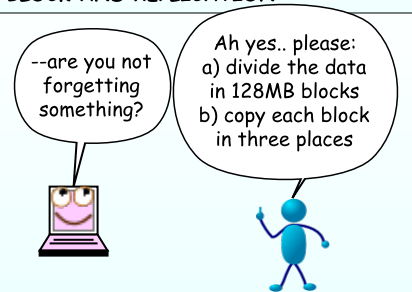


WRITING DATA IN HDFS CLUSTER

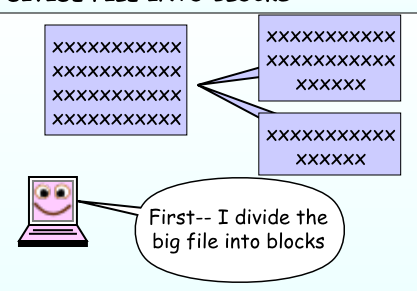
REQUEST FROM USER



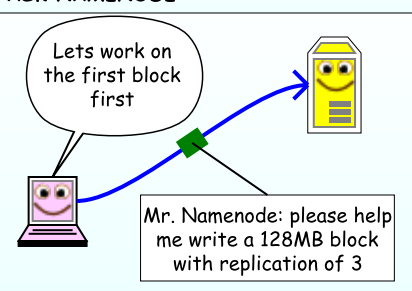
BLOCK AND REPLICATION



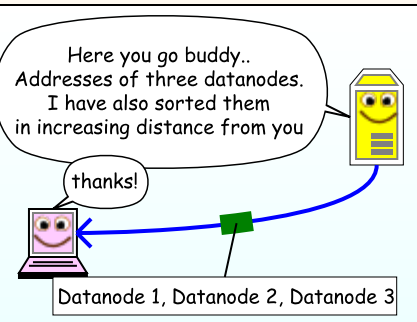
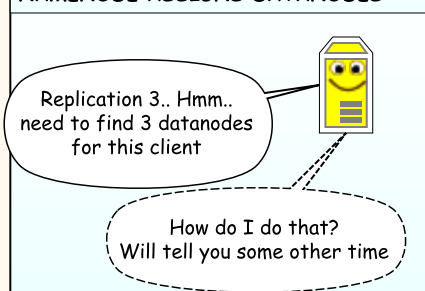
DIVIDE FILE INTO BLOCKS



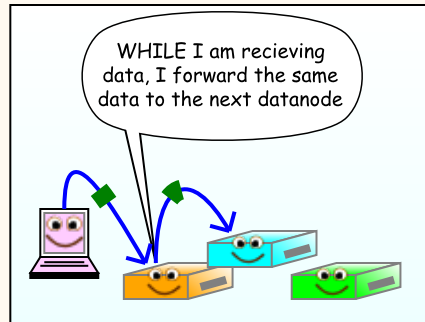
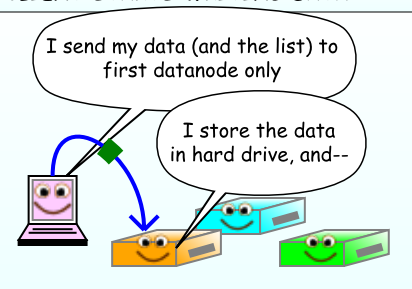
ASK NAMENODE

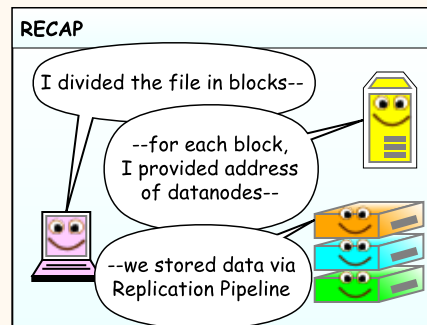
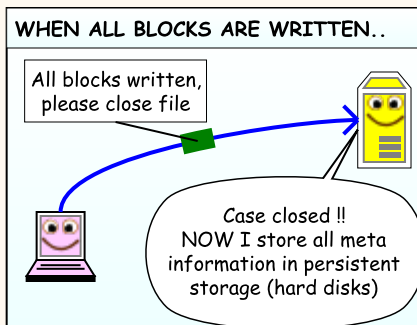
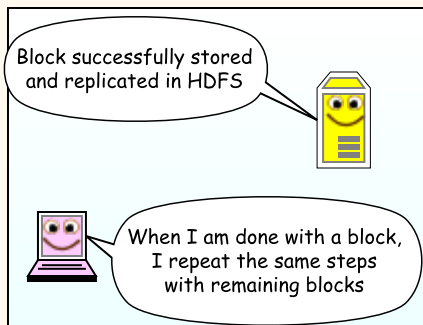
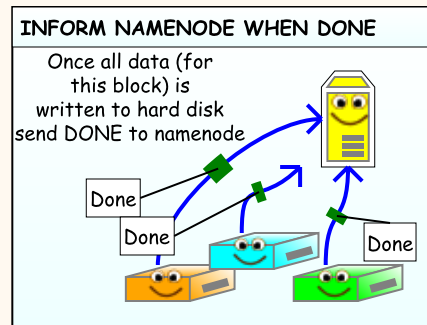
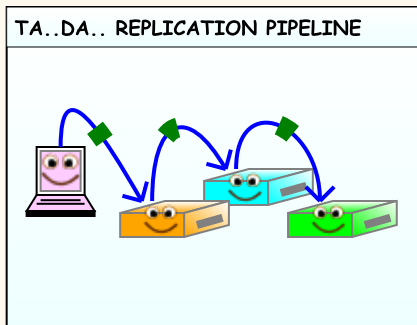
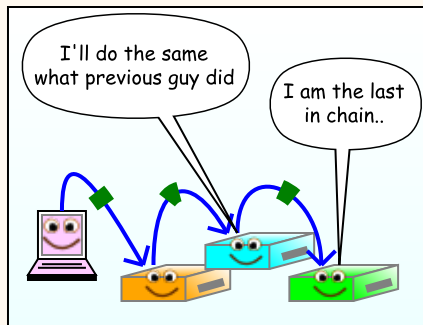


NAMENODE ASSIGNS DATANODES

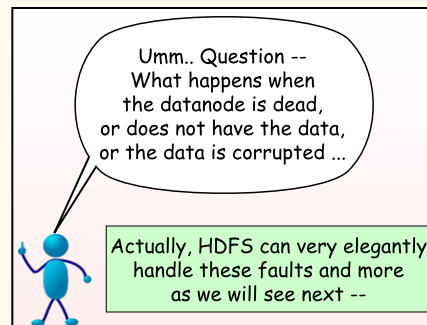
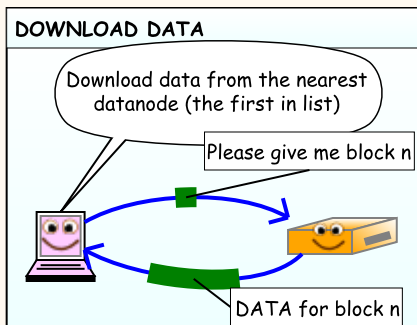
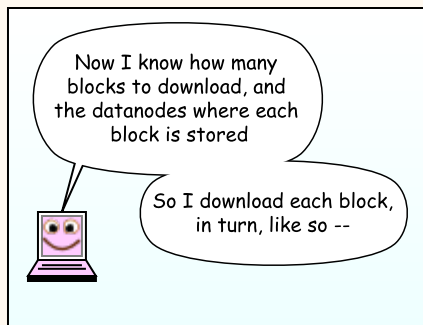
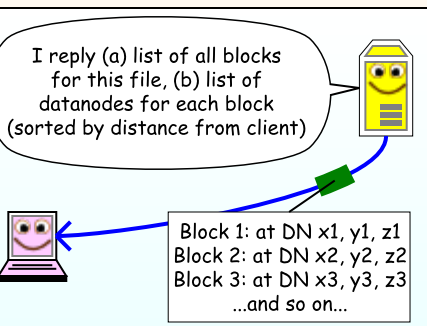
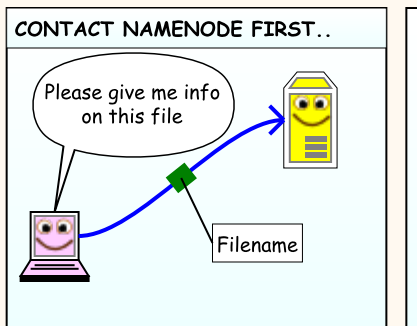
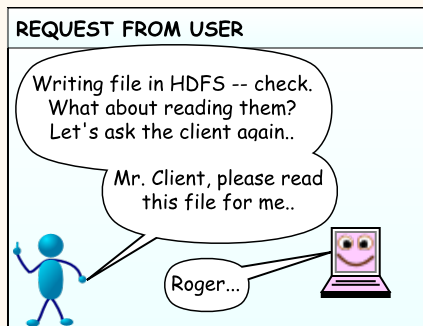


CLIENT STARTS WRITING DATA





READING DATA IN HDFS CLUSTER



FAULT TOLERANCE IN HDFS. PART I: TYPES OF FAULTS AND THEIR DETECTION

FAULT I: NODE FAILURE

There are typically three kinds of faults:
The first is **NODE FAILURE**

Goodbye,
cruel world



DETECTION #1: NODE FAILURES

NOTE:
If Namenode is dead,
the entire cluster is dead!
Namenode is the **SINGLE**
POINT OF FAILURE



Instead, let's focus on
how datanode failures
are detected

FAULT II: COMMUNICATION FAILURE

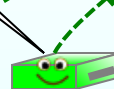
Second is **COMMUNICATION FAILURE**
(cannot send and receive data)

where IS everybody?



DETECTING DATANODE FAILURE

We send **HEARTBEAT**
message every 3 seconds.
This is our way of
saying we are alive



FAULT III: DATA CORRUPTION

Third is **DATA CORRUPTION**

Data can be corrupted while
sending over network



Data on Disk



Or corrupted while it is
stored in hard disks

DETECTION #2: NETWORK FAILURES

Whenever data is sent,
an **ACK** is replied by the receiver



Data

ACK



If the **ACK** is not received (after several
retries), the sender assumes that the host
is dead, or the network has failed

DETECTION #3: CORRUPTED DATA

Checksum is sent along with
transmitted data

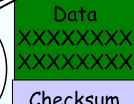


Checksum

Data



Moreover, when I store
data in hard disks,
I also store the checksum



DETECTING CORRUPTED HARD DRIVES

Periodically, all datanodes
send **BLOCKREPORT** to the namenode



List of all
blocks I have



Before sending block report
I check if checksums are ok.
I don't send info for
blocks that are corrupted

I have
four blocks

I thought he had five
blocks.. so one
block is corrupted



RECAP: HEARTBEAT MESSAGES AND BLOCK REPORTS

We send heartbeats every
3 seconds to say we are alive

HEARTBEAT

BLOCK
REPORT



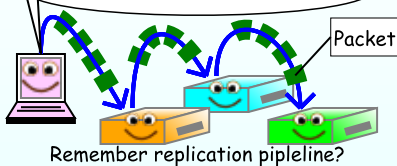
We send block reports
and we skip blocks
that are corrupted

(which is how the
namenode will know
which blocks are lost)

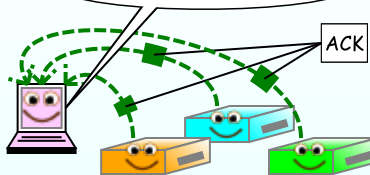
FAULT TOLERANCE IN HDFS. PART II: HANDLING READING AND WRITING FAILURES

HANDLING WRITE FAILURES

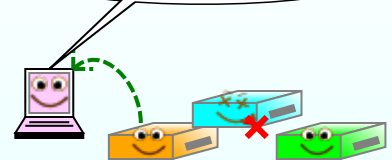
One thing I should have said earlier..
I write the block in smaller data units (usually 64KB) called "packets"



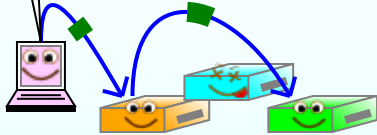
Moreover, each datanode replies back an ACK for each packet to confirm that they got it



So, if I don't get ACKs from some datanode, I know it is dead.
I adjust the pipeline to skip him

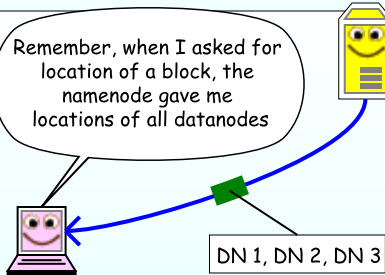


Here's the adjusted pipeline.
Note that the block will be "under replicated", but the namenode will take care of that later on

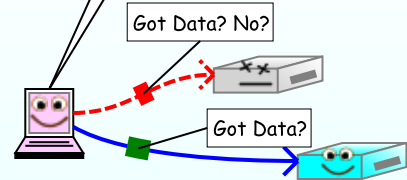


HANDLING READ FAILURES

Remember, when I asked for location of a block, the namenode gave me locations of all datanodes



If one datanode is dead, I read from the others in the list



FAULT TOLERANCE IN HDFS. PART III: HANDLING DATANODE FAILURES

First-- I must tell you about the two tables I keep..



List of Blocks
Block 1 - stored at DN1, DN2, DN3
Block 2 - stored at DN1, DN4, DN5

List of Datanodes
Datanode 1 - has block 1, 2, ..
Datanode 2 - has block 1, 5, ..

I continuously update these two tables--

If I find a block on a datanode is corrupted, I update first table (by removing bad DN from block's list)

And if I find that a datanode has died, I update both tables

UNDER REPLICATED BLOCKS

I scan the first list (list of blocks) periodically, and see if there are blocks that are not replicated properly

These are called "under replicated" blocks

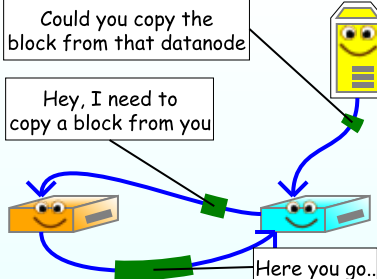
For all under-replicated blocks, I ask other datanodes to copy them from datanodes that have the replica

like so --



Could you copy the block from that datanode

Hey, I need to copy a block from you



Umm.. one more question: All of this works if there is atleast one valid copy of the block somewhere.. right?

That's correct. HDFS cannot guarantee that atleast one replica will always survive. But it tries it best by smartly selecting replica locations, as we will see next --



REPLICA PLACEMENT STRATEGY

Remember I promised to tell you how I select datanode locations for storing the replicas of a block?

Hang tight.. here it goes..

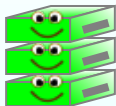


RACKS AND DATANODES

The cluster is divided into RACKS
Each rack has multiple datanodes



Rack 1



Rack 2



Rack 3

SELECTING FIRST REPLICA LOCATION

First replica location is simple:

If the writer is a member of cluster,
it is selected as first replica

Otherwise some random
datanode is selected



NEXT TWO REPLICA LOCATIONS

Pick a different rack than first replica's
Select two different datanode on that rack

first replica

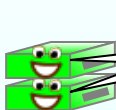
next two replicas



SUBSEQUENT REPLICA LOCATIONS

Pick any random datanode,
if it satisfies these two conditions:

Only one replica per datanode



Max two
replicas
per rack

Please note the fine print: sometimes
those two conditions cannot be satisfied,
in which case they are .. ahem.. ignored
(convenient eh?)

Also, HDFS allows you use your
own placement algorithm.
So if you know a better
algorithm, don't be shy now...



WHERE TO GO FROM HERE?

I do a lot of other things
as well.. read more
about me at websites and books..



Or best of all,
install and run HDFS
and see for yourself!!



We do more than store data.
We can run "Map-Reduce" jobs
Read about map reduce
in our next comics



THE
END

