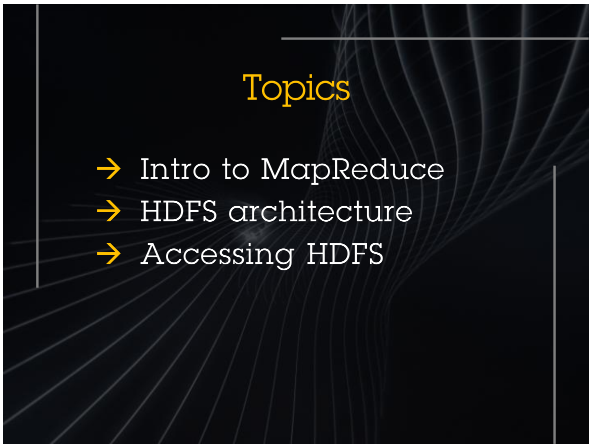
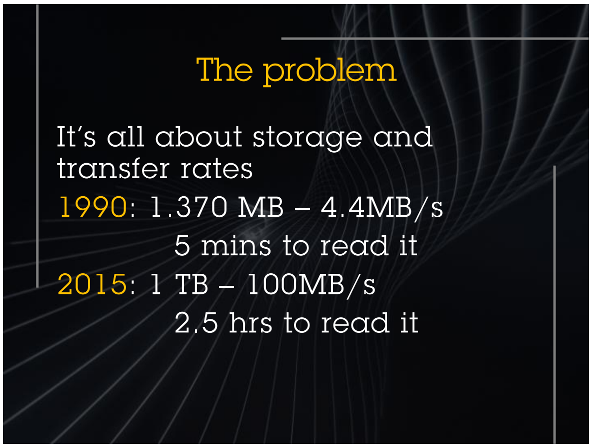
INTRODUZIONE AD HADOOP



Topics

- → Intro to MapReduce
- → HDFS architecture
- → Accessing HDFS

INTRO TO MAPREDUCE

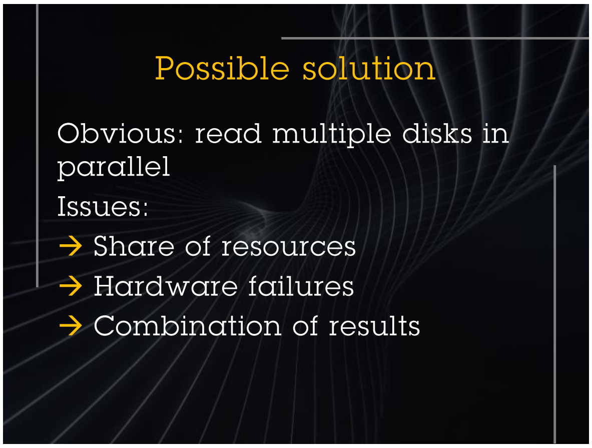


The problem

It's all about storage and transfer rates

1990: 1.370 MB – 4.4MB/s
5 mins to read it

2015: 1 TB – 100MB/s
2.5 hrs to read it

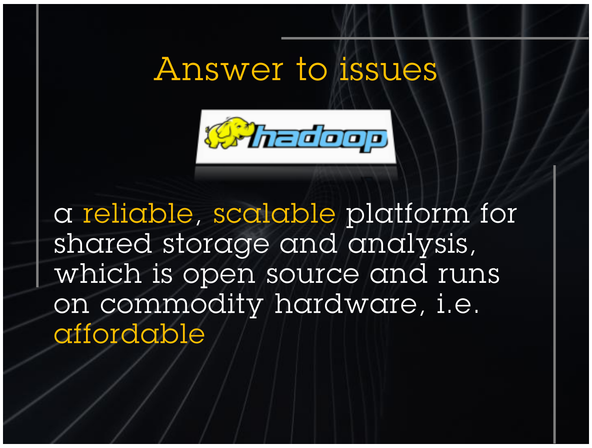


Possible solution

Obvious: read multiple disks in parallel

Issues:

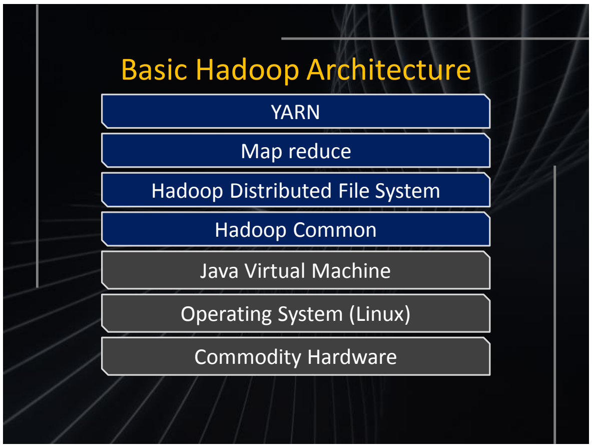
- → Share of resources
- → Hardware failures
- → Combination of results



Answer to issues

hadoop

a reliable, scalable platform for shared storage and analysis, which is open source and runs on commodity hardware, i.e. affordable



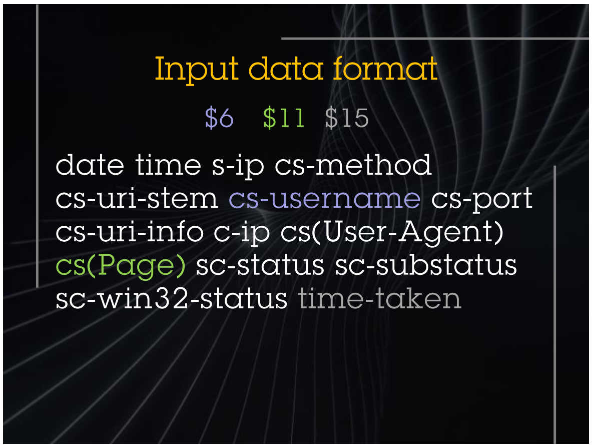
Basic Hadoop Architecture

- YARN
- Map reduce
- Hadoop Distributed File System
- Hadoop Common
- Java Virtual Machine
- Operating System (Linux)
- Commodity Hardware

Example: website logs

We want to identify the behaviour of the users of our website:

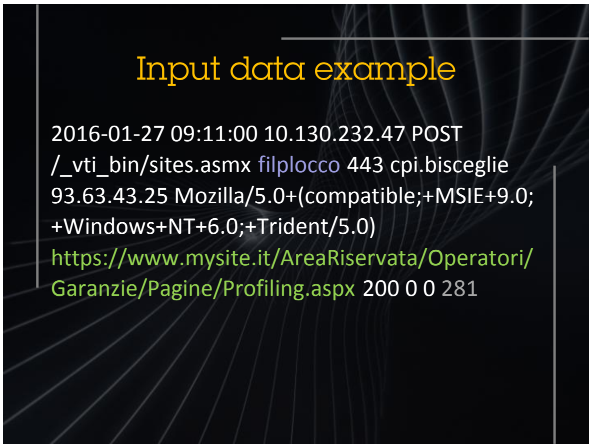
- time page impression
- number page impression
- etc…



Input data format

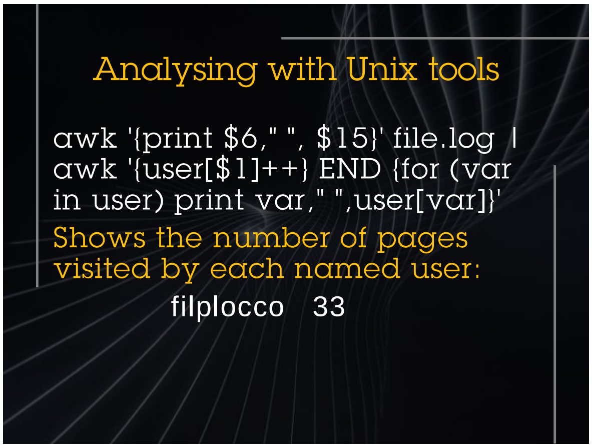
$6 $11 $15

date time s-ip cs-method cs-uri-stem cs-username cs-port cs-uri-info c-ip cs(User-Agent) cs(Page) sc-status sc-substatus sc-win32-status time-taken



Input data example

```
2016-01-27 09:11:00 10.130.232.47 POST
/_vti_bin/sites.asmx filplocco 443 cpi.bisceglie
93.63.43.25 Mozilla/5.0+(compatible;+MSIE+9.0;
+Windows+NT+6.0;+Trident/5.0)
https://www.mysite.it/AreaRiservata/Operatori/
Garanzie/Pagine/Profiling.aspx 200 0 0 281
```

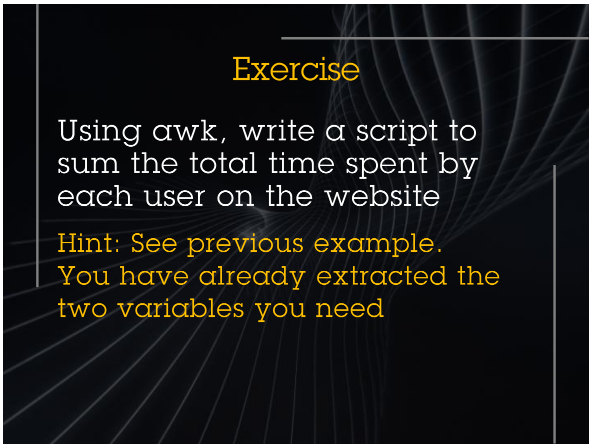


Analysing with Unix tools

```
awk '{print $6," ", $15}' file.log |
awk '{user[$1]++} END {for (var
in user) print var," ",user[var]}'
```

Shows the number of pages visited by each named user:

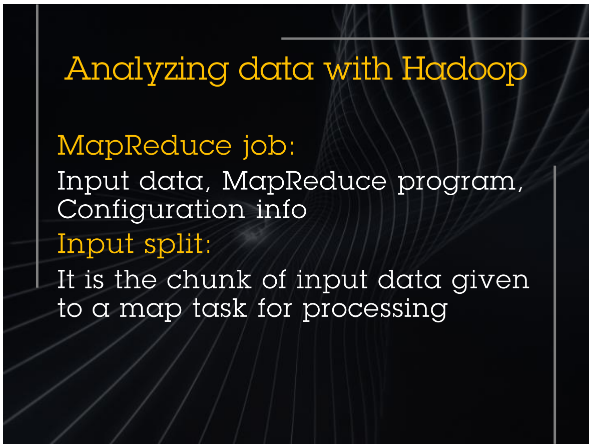
```
filplocco   33
```



Exercise

Using awk, write a script to sum the total time spent by each user on the website

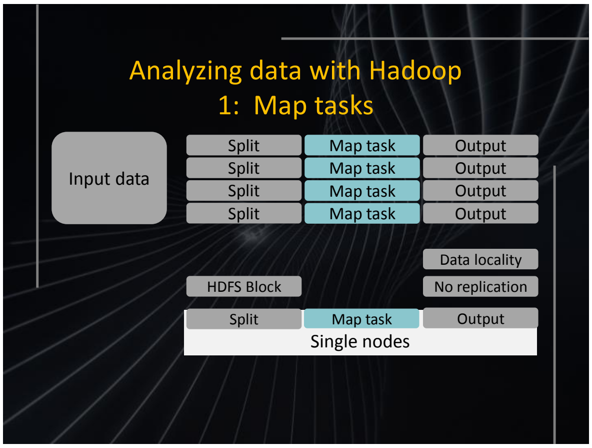
Hint: See previous example. You have already extracted the two variables you need



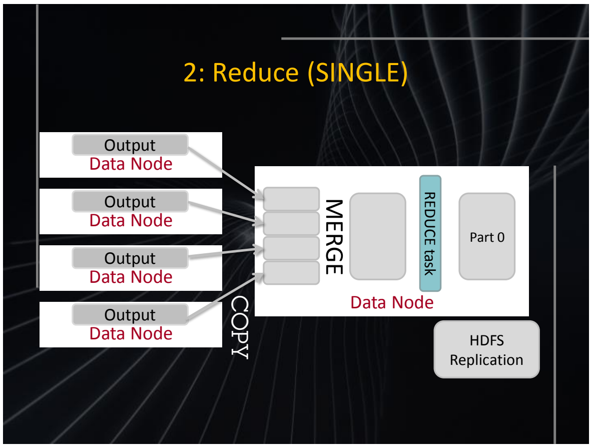
Analyzing data with Hadoop

MapReduce job:
Input data, MapReduce program, Configuration info

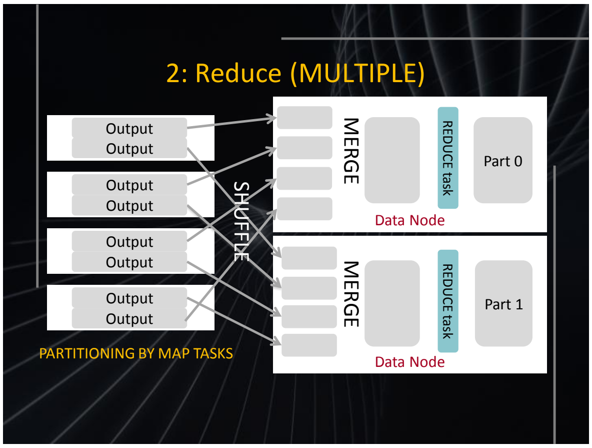
Input split:
It is the chunk of input data given to a map task for processing



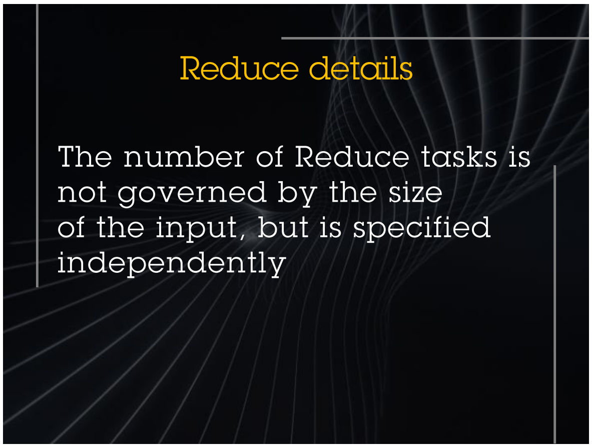
Analyzing data with Hadoop
1: Map tasks



2: Reduce (SINGLE)

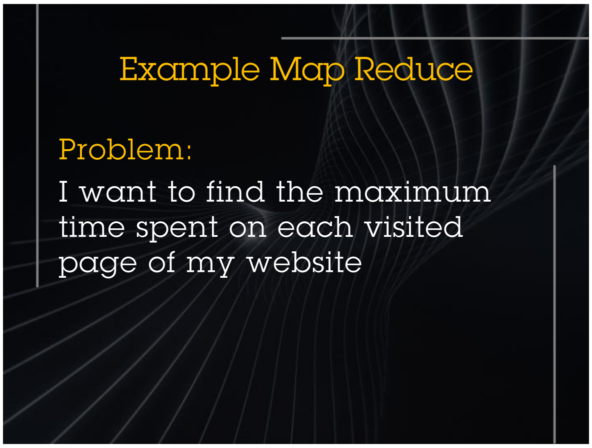


2: Reduce (MULTIPLE)



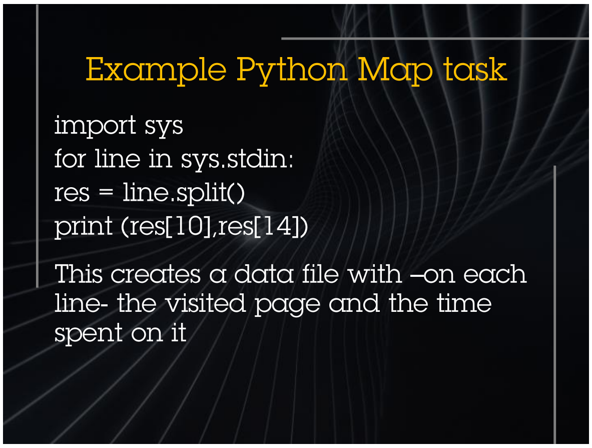
Reduce details

The number of Reduce tasks is not governed by the size of the input, but is specified independently



Example Map Reduce

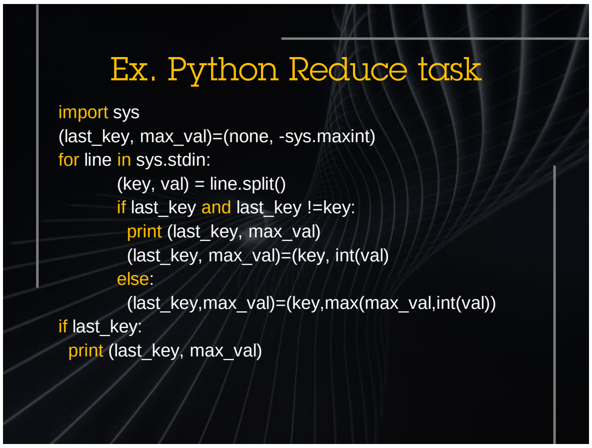
Problem:
I want to find the maximum time spent on each visited page of my website



Example Python Map task

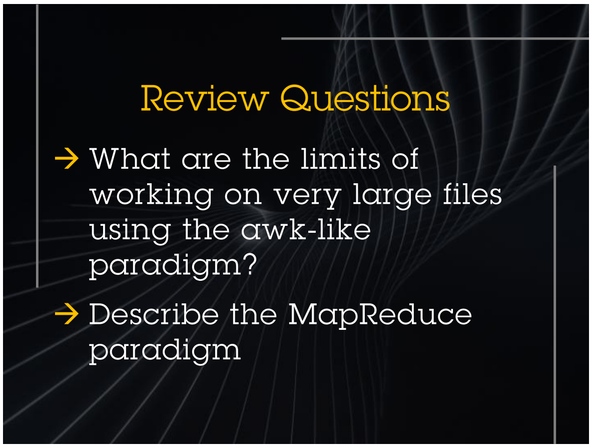
```
import sys
for line in sys.stdin:
res = line.split()
print (res[10],res[14])
```

This creates a data file with –on each line- the visited page and the time spent on it



Ex. Python Reduce task

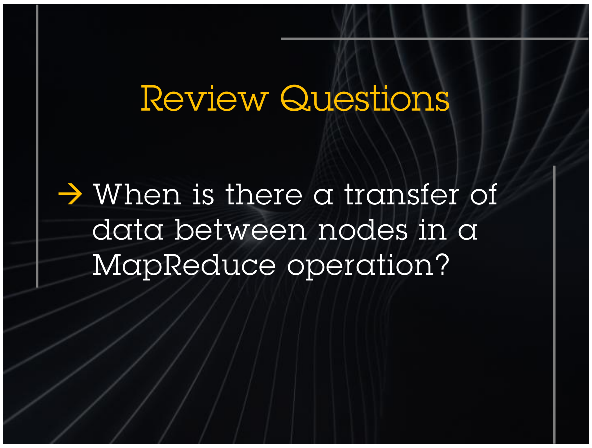
```
import sys
(last_key, max_val)=(none, -sys.maxint)
for line in sys.stdin:
        (key, val) = line.split()
        if last_key and last_key !=key:
          print (last_key, max_val)
          (last_key, max_val)=(key, int(val)
        else:
          (last_key,max_val)=(key,max(max_val,int(val))
if last_key:
  print (last_key, max_val)
```



Review Questions

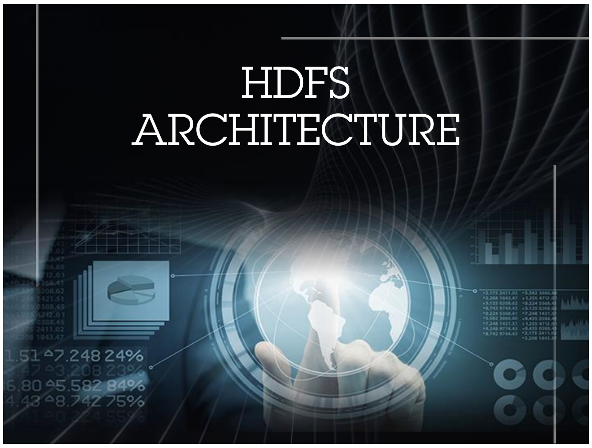
→ What are the limits of working on very large files using the awk-like paradigm?

→ Describe the MapReduce paradigm

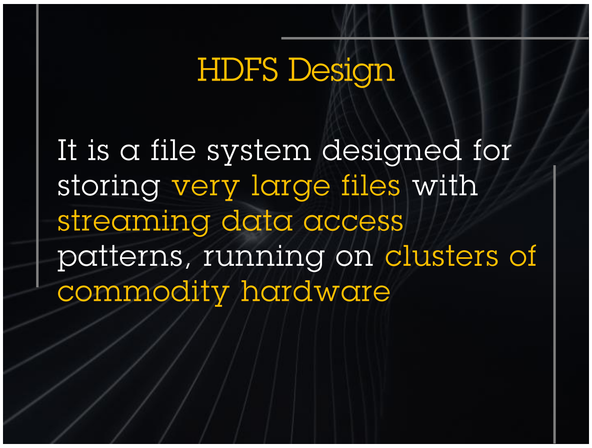


Review Questions

→ When is there a transfer of data between nodes in a MapReduce operation?

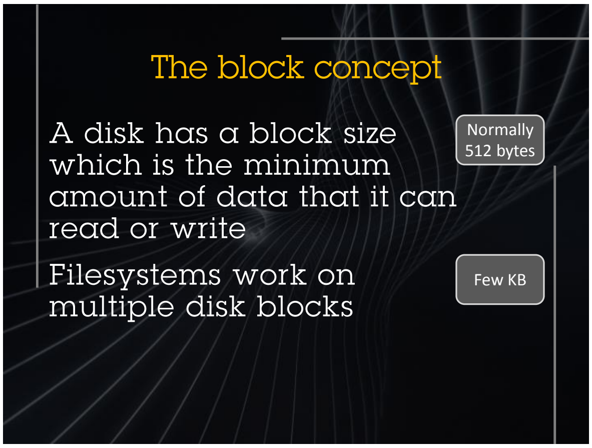


HDFS ARCHITECTURE



HDFS Design

It is a file system designed for storing very large files with streaming data access patterns, running on clusters of commodity hardware



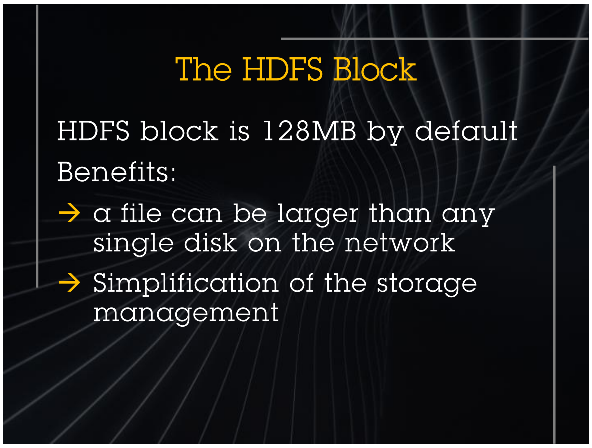
The block concept

A disk has a block size which is the minimum amount of data that it can read or write

Normally 512 bytes

Filesystems work on multiple disk blocks

Few KB



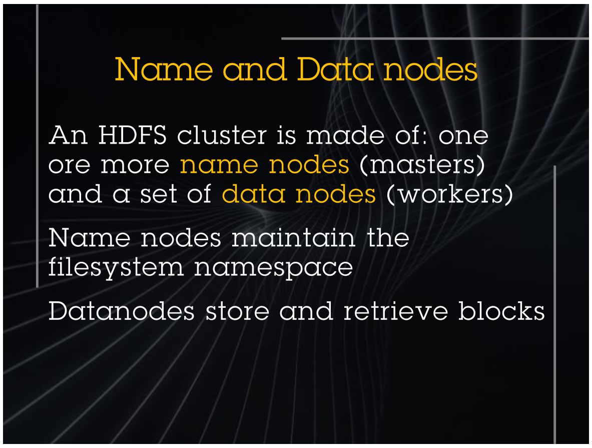
The HDFS Block

HDFS block is 128MB by default

Benefits:

→ a file can be larger than any single disk on the network

→ Simplification of the storage management

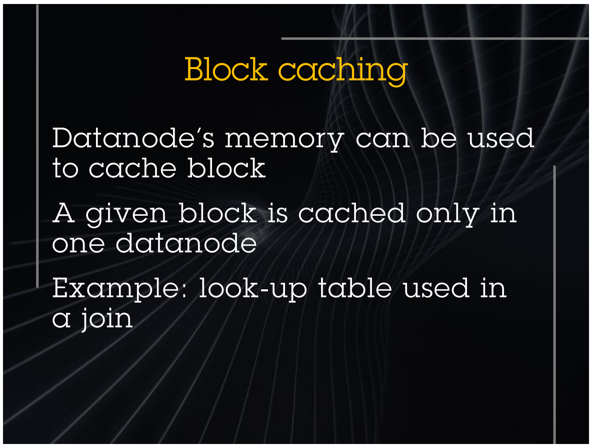


Name and Data nodes

An HDFS cluster is made of: one ore more name nodes (masters) and a set of data nodes (workers)

Name nodes maintain the filesystem namespace

Datanodes store and retrieve blocks

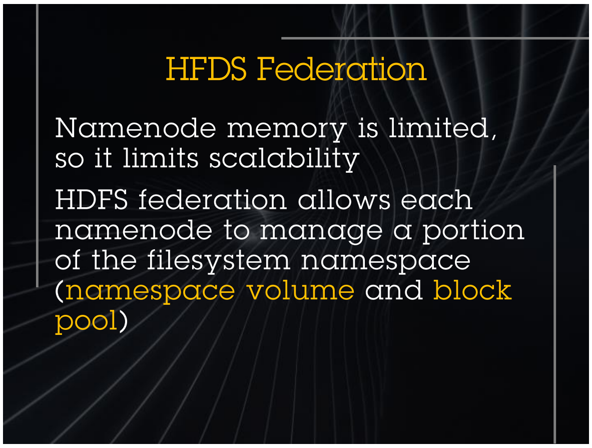


Block caching

Datanode's memory can be used to cache block

A given block is cached only in one datanode

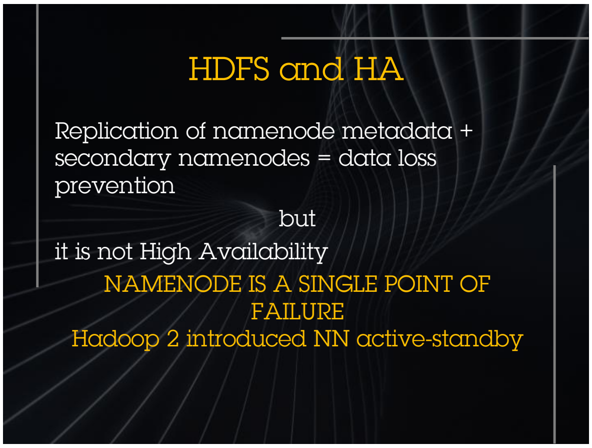
Example: look-up table used in a join



HFDS Federation

Namenode memory is limited, so it limits scalability

HDFS federation allows each namenode to manage a portion of the filesystem namespace (namespace volume and block pool)



HDFS and HA

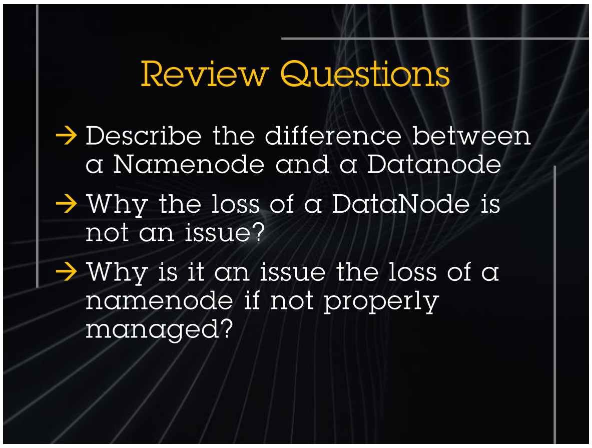
Replication of namenode metadata + secondary namenodes = data loss prevention

but

it is not High Availability

NAMENODE IS A SINGLE POINT OF FAILURE

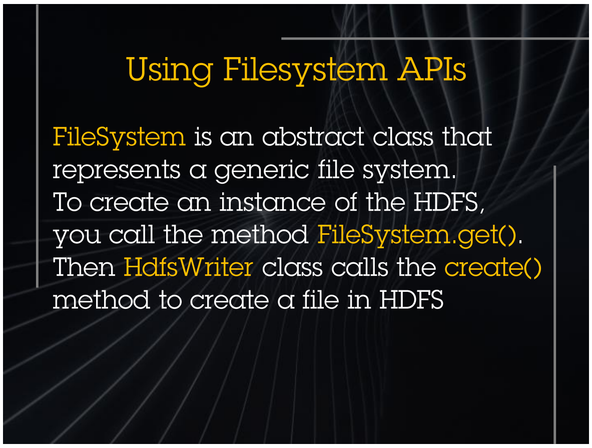
Hadoop 2 introduced NN active-standby



Review Questions

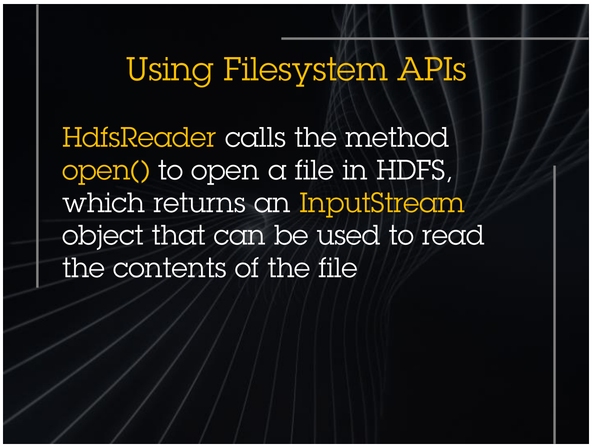
- → Describe the difference between a Namenode and a Datanode
- → Why the loss of a DataNode is not an issue?
- → Why is it an issue the loss of a namenode if not properly managed?

ACCESSING HDFS



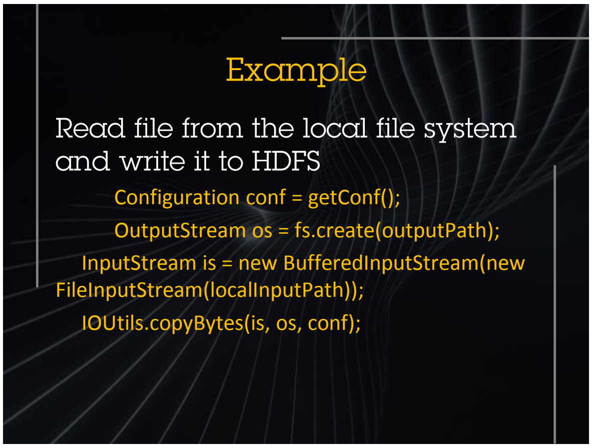
Using Filesystem APIs

FileSystem is an abstract class that represents a generic file system.
To create an instance of the HDFS, you call the method FileSystem.get().
Then HdfsWriter class calls the create() method to create a file in HDFS



Using Filesystem APIs

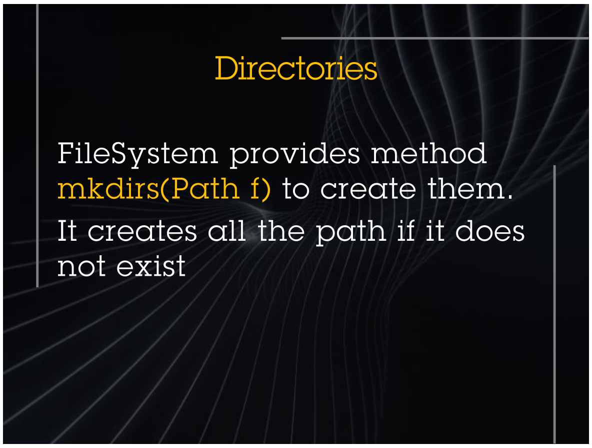
HdfsReader calls the method open() to open a file in HDFS, which returns an InputStream object that can be used to read the contents of the file



Example

Read file from the local file system and write it to HDFS

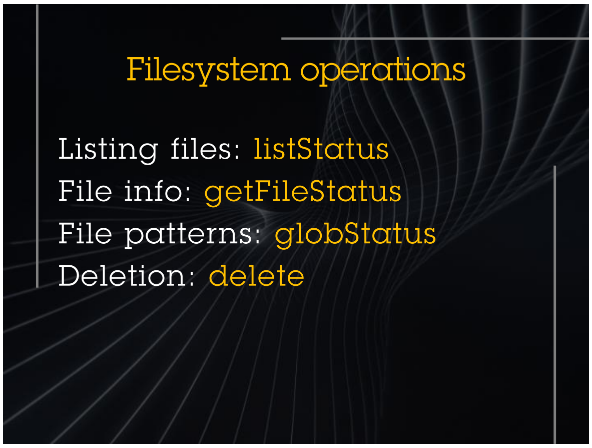
```
        Configuration conf = getConf();
        OutputStream os = fs.create(outputPath);
    InputStream is = new BufferedInputStream(new
FileInputStream(localInputPath));
    IOUtils.copyBytes(is, os, conf);
```



Directories

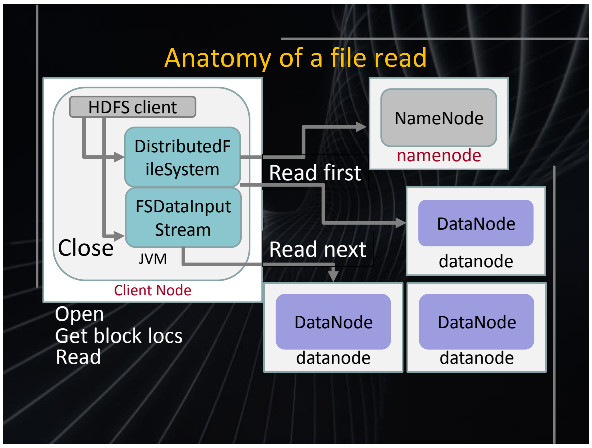
FileSystem provides method mkdirs(Path f) to create them.

It creates all the path if it does not exist



Filesystem operations

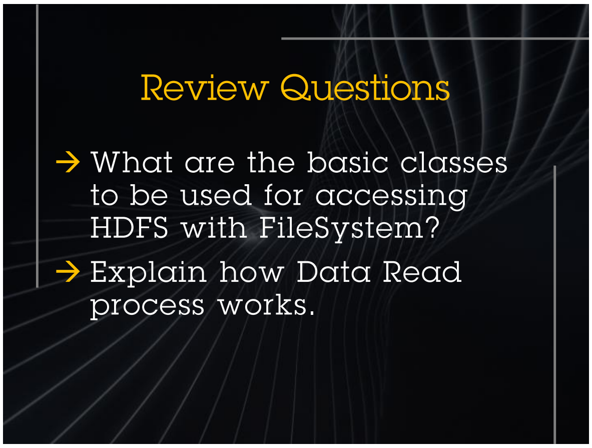
Listing files: listStatus
File info: getFileStatus
File patterns: globStatus
Deletion: delete



Anatomy of a file read

HDFS client
DistributedFileSystem
FSDataInputStream
Close
JVM
Client Node
NameNode
namenode
Read first
Read next
DataNode
datanode
DataNode
datanode
DataNode
datanode

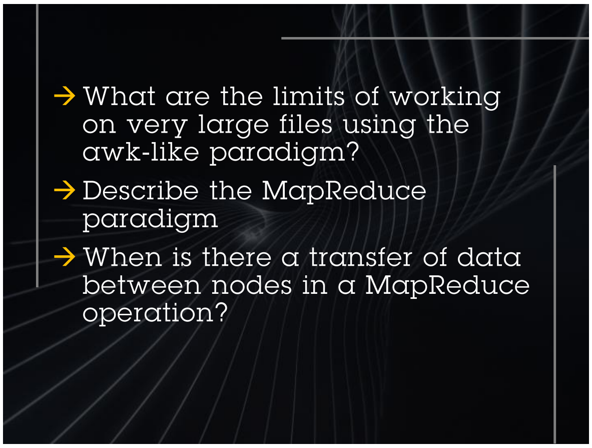
Open
Get block locs
Read



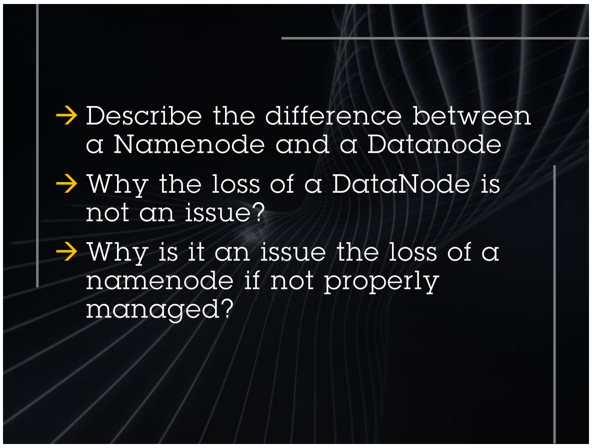
Review Questions

- → What are the basic classes to be used for accessing HDFS with FileSystem?
- → Explain how Data Read process works.

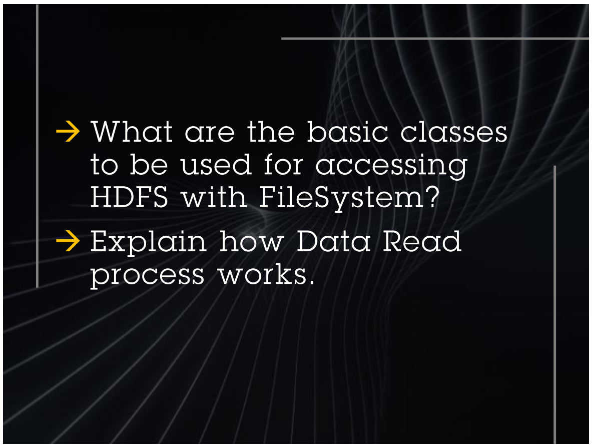
SUMMARY QUESTIONS



- → What are the limits of working on very large files using the awk-like paradigm?
- → Describe the MapReduce paradigm
- → When is there a transfer of data between nodes in a MapReduce operation?



- → Describe the difference between a Namenode and a Datanode
- → Why the loss of a DataNode is not an issue?
- → Why is it an issue the loss of a namenode if not properly managed?



- → What are the basic classes to be used for accessing HDFS with FileSystem?
- → Explain how Data Read process works.