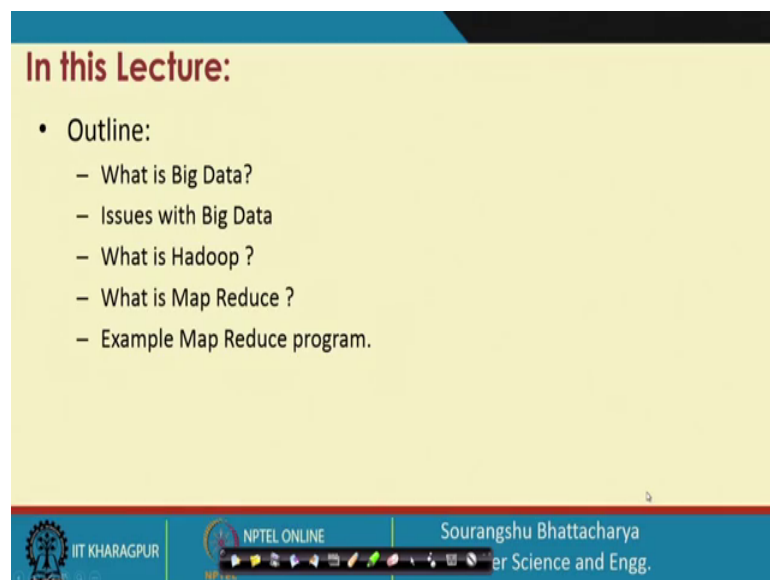


**Scalable Data Science**  
**Prof. Sourangshu Bhattacharya**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture -26**  
**Map Reduce and Hadoop**

Hello, welcome to NPTEL course on Scalable Data Science. Today, we are going to discuss Map Reduce and Hadoop. I am Sourangshu Bhattacharya, department of Computer Science and Engineering, IIT Kharagpur.

(Refer Slide Time: 00:34).



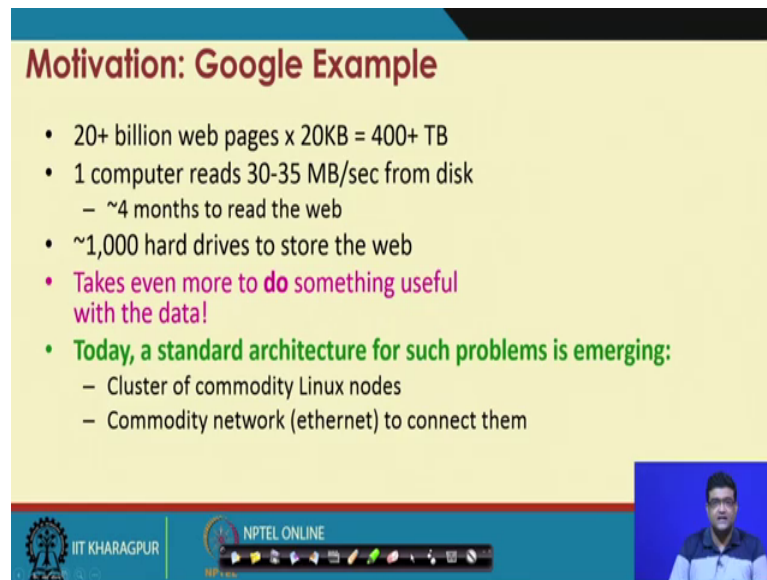
**In this Lecture:**

- Outline:
  - What is Big Data?
  - Issues with Big Data
  - What is Hadoop ?
  - What is Map Reduce ?
  - Example Map Reduce program.

The slide features a yellow background with a blue header and footer. The footer contains the IIT Kharagpur logo, NPTEL ONLINE text, and the name Sourangshu Bhattacharya, Department of Computer Science and Engineering.

So, in this lecture, we will cover the idea of Big Data. What is big data, , what are the issues with big data; then, we will discuss what is the, what is Hadoop, and we will also discuss, what is Map Reduce, and then we will see an example-Map Reduce Program and how it functions .

(Refer Slide Time: 01:01).



**Motivation: Google Example**

- 20+ billion web pages x 20KB = 400+ TB
- 1 computer reads 30-35 MB/sec from disk
  - ~4 months to read the web
- ~1,000 hard drives to store the web
- Takes even more to do something useful with the data!
- Today, a standard architecture for such problems is emerging:
  - Cluster of commodity Linux nodes
  - Commodity network (ethernet) to connect them

The slide includes logos for IIT KHARAGPUR and NPTEL ONLINE, and a small video inset of a man speaking in the bottom right corner.

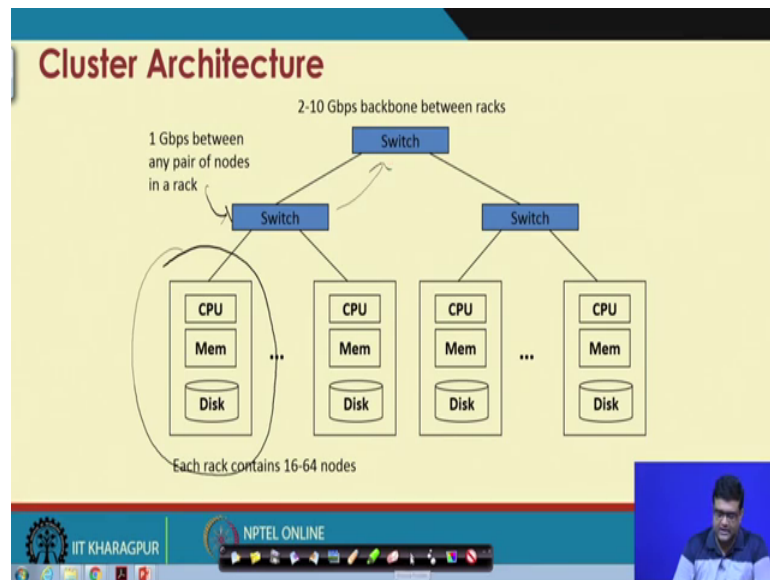
So, consider this, example of Google. So, Google has roughly 20 billion web pages to index. Let us say, it might be more, but let us assume that it has 20 billion web pages to index. Each web page is of 20 kilo bytes size. So, it has to index more than 400 tera bytes of data.

Now, consider this, one computer if you have one hard disc in a computer, it reads at the rate of 30-35, 35 megabytes per second. So, this means that, in order to index 20 billion web pages on one computer, you would take roughly 4 months. In 4 months, many of the web pages would have changed. So, you again have to re index them. So, this solution is not feasible.

So what is, what do you do? So, what Google does is it uses many-many hard drives on many-many computers to store the data? But it takes so, it, but it takes more time or more resources to do something useful with the data.

So, what is the solution? So, you know that, today, a standard architecture, for solving that kind of problems is a Cluster of commodity machines. Instead of, having one very large computer, you have a Cluster of community computers, and these are connected over, let us say, a commodity network, something like an Ethernet network.

(Refer Slide Time: 03:37).



So, instead of, using one large computer, we use many small computers so, to solve this problem. Now, what is the architecture for this? The architecture for this is something like this; you have typically 1 Gbps switches. So, you have racks where racks are kept. So, these are. So, this is one rack typically in one rack, you can have between 16-64 nodes.

These racks are connected by a first level switch. So, each of the computers in a rack are connected and each rack is connected by a first level switch and these switches are further connected by a second level switch. So, this is the cluster architecture that is typically followed in, in many of the data, data centres that one uses.

(Refer Slide Time: 04:36).

**Large-scale Computing**

- Large-scale computing for data mining problems on commodity hardware
- Challenges:
  - How do you distribute computation?
  - How can we make it easy to write distributed programs?
  - Machines fail:
    - One server may stay up 3 years (1,000 days)
    - If you have 1,000 servers, expect to loose 1/day
    - People estimated Google had ~1M machines in 2011
      - 1,000 machines fail every day!

IIT KHARAGPUR NPTEL ONLINE

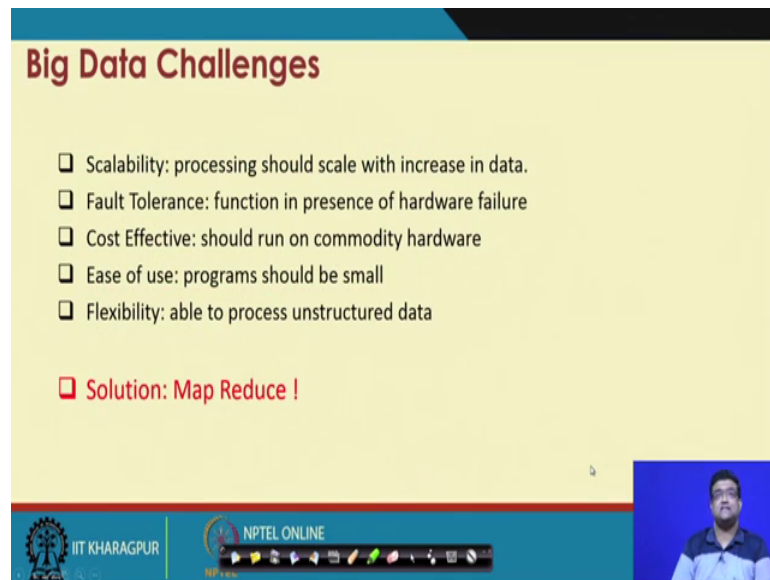
So, now this way of computing over very large data, using commodity hardware, throws a certain challenge. So, what is the challenge? The first challenge is, how do you distribute the computation; says the how do in which machine? So, if you have thousands of servers; in which machine which indexing should go on?

So, this is the first challenge, the second challenge is, how can you easily write such distributed program, because writing distributed programs can take a lot of time and may require highly expert programmers., and the third challenge is, that when you have many many servers, the some of the servers can fail. So, if we do some back of the pump calculation.

So, let us say, we assume that one server can stay up for 3 years. So, after 3 years, it will fail. As in one way or the other it will fail in roughly 1000 days which is 3 years.

So, if you have 1000 servers, on an average you can expect, that one server will fail every day. So, for example, in case of Google, people expect that they had about 1M machines. So, that means, about 1000 machine fail every day. So, whatever program one writes, whatever distributed program one writes, has to be tolerant to this failure. So, with these ideas in mind, so or these challenges in mind, so, what are the challenges? Just to reiterate the challenges.

(Refer Slide Time: 06:43).



**Big Data Challenges**

- Scalability: processing should scale with increase in data.
- Fault Tolerance: function in presence of hardware failure
- Cost Effective: should run on commodity hardware
- Ease of use: programs should be small
- Flexibility: able to process unstructured data

**Solution: Map Reduce !**

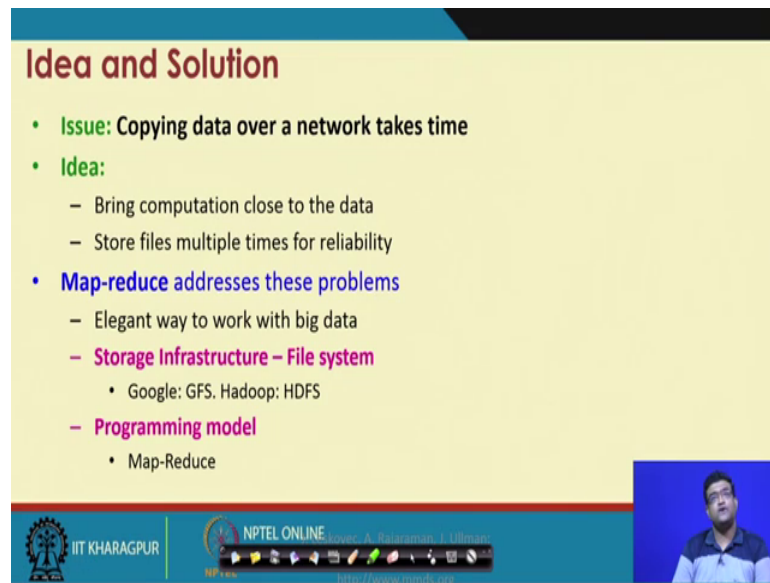
IIT KHARAGPUR | NPTEL ONLINE

The first challenge is the scalability challenge. So, the processing should scale with the increase, in the data and this scalability in order to solve this scalability challenge we started using Clusters of commodity machines. That induced a second challenge which is the fault tolerance challenge. So, the program that you write should function even in presence of hardware failure.

The third challenge is the cost effectiveness. So, it should be able to run on commodity hardware or you know like desktop computers and things like this. The fourth challenge is ease of use. So, the programs writing program should not take many many days or very expert knowledge. So, the program should be easily writable.

And last is, there should be flexibility. So, for example, these, one should be able to process, unstructured data like web pages or you know, news articles, take things like this using this framework.

(Refer Slide Time: 08:30).



The slide, titled "Idea and Solution", is presented on a yellow background. It contains the following content:

- **Issue:** Copying data over a network takes time
- **Idea:**
  - Bring computation close to the data
  - Store files multiple times for reliability
- **Map-reduce** addresses these problems
  - Elegant way to work with big data
  - **Storage Infrastructure – File system**
    - Google: GFS. Hadoop: HDFS
  - **Programming model**
    - Map-Reduce

The slide footer includes the IIT Kharagpur logo, the NPTEL ONLINE logo, and a small video inset of a presenter in the bottom right corner.

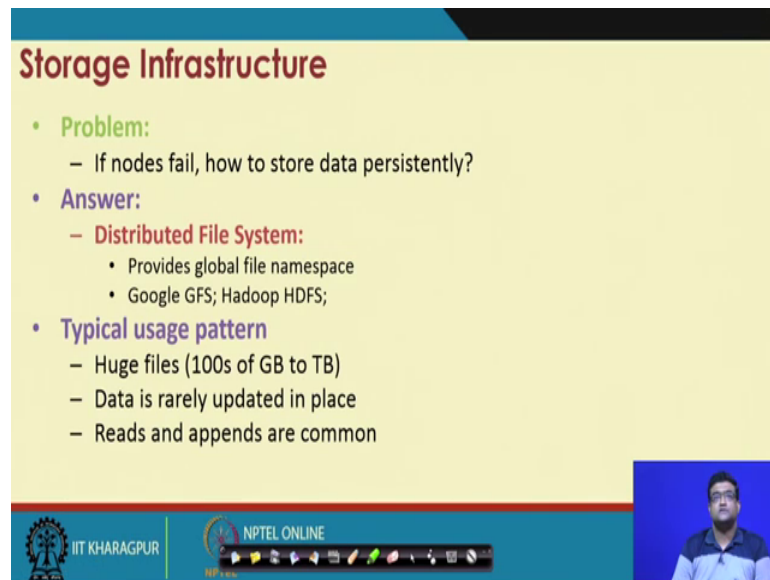
So, in order to address all these challenges the map reduce framework for computing with big data was born. So, the first issue that this Map Reduce framework faced was that, if we have a really large amount of data, copying this large amount of data over network, will take a lot of time and a lot of bandwidth.

So, what is the solution to that; the solution is instead of, copying the data over the network, you bring the computation close to the data, which is on whichever machine the data, a particular portion of the data is stored, you could do the computation for that particular portion of the data on that machine, and you store files in multiple places,.

So, you make a copy of the files or the data which is stored in these files, in multiple machines. This helps both improve reliability and also the paradigm of computing something locally. So, there is a high chance, that you can get a local copy of the data, on a particular machine.

So, map reduce is uses the following way to address these problems. So, the first is the storage infrastructure which is typically a distributed file system. So, in case of Google internally Google uses something, called the Google file system. The open source framework, Hadoop uses Hadoop distributed file system, which we will discuss in detail in the next lecture. So, for storage you use a distributed file system, for programming you use what is called the map reduce programming paradigm, . We will discuss this next.

(Refer Slide Time: 10:50).



**Storage Infrastructure**

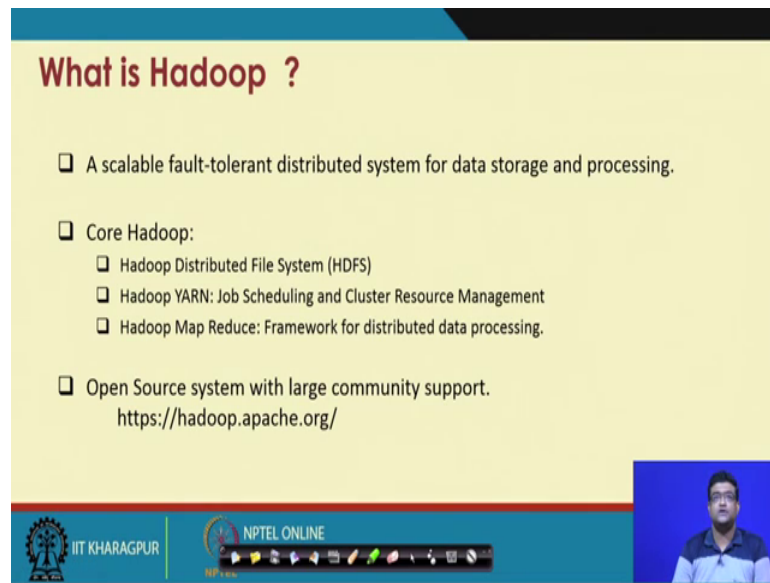
- **Problem:**
  - If nodes fail, how to store data persistently?
- **Answer:**
  - **Distributed File System:**
    - Provides global file namespace
    - Google GFS; Hadoop HDFS;
- **Typical usage pattern**
  - Huge files (100s of GB to TB)
  - Data is rarely updated in place
  - Reads and appends are common

IIT KHARAGPUR | NPTEL ONLINE

So then second problem is what, if nodes fail?. So, if nodes fail the data stored on these nodes, may be lost, or. So, the challenge is how to store the data persistently even though certain nodes fail, and the answer is to have a distributed file system which provides global namespace, but it provides ability to copy. So, it stores data redundantly, in multiple machines and it provides ability to copy this data from one machine to another seamlessly additionally

So, the typical usage pattern, for this kind of a distributed file system, is that you have huge files. So, and data is rarely updated, but data is very frequently read and very frequently appended or new data new records of data are added very very frequently.

(Refer Slide Time: 12:35).



**What is Hadoop ?**

- ❑ A scalable fault-tolerant distributed system for data storage and processing.
- ❑ Core Hadoop:
  - ❑ Hadoop Distributed File System (HDFS)
  - ❑ Hadoop YARN: Job Scheduling and Cluster Resource Management
  - ❑ Hadoop Map Reduce: Framework for distributed data processing.
- ❑ Open Source system with large community support.  
<https://hadoop.apache.org/>

The slide also features logos for IIT KHARAGPUR and NPTEL ONLINE, along with a small video inset of a man speaking in the bottom right corner.

So, what is Hadoop? So, Hadoop is a scalable, fault tolerant, distributed system for processing storage and processing of big data. So, it addresses all the challenges of big data that we discussed earlier.

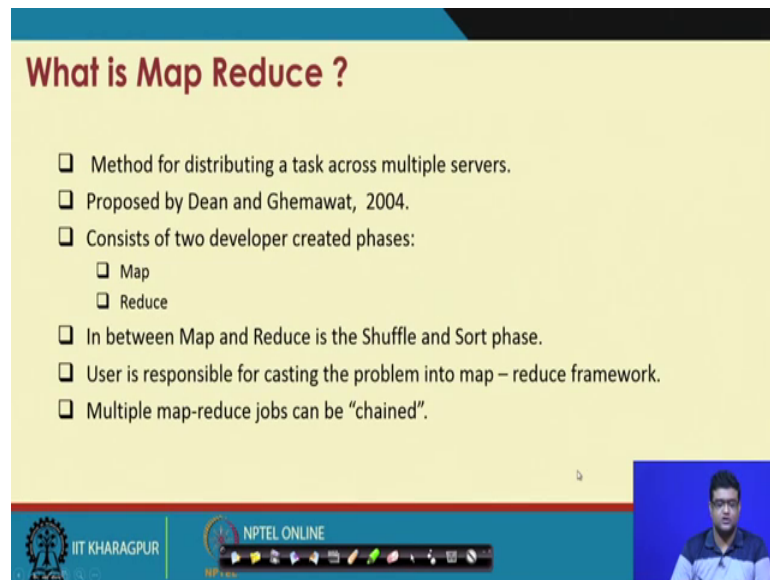
So, Core Hadoop has the following three components. The first component is called the Hadoop distributed file system which allows you to store the data in a distributed manner and over a Cluster of commodity servers and it also satisfies all the properties of fault tolerance and redundancy.

Next component is the Hadoop yarn which is the job scheduler and the Cluster resource manager. So, this schedules the map reduce jobs on appropriate computers and it also keep tracks of where a particular job is based scheduled, and where which jobs are running.

And finally, the third component is the Hadoop Map Reduce component which is a framework for distributed data processing which we will discuss next and Hadoop is an open source community project.



(Refer Slide Time: 14:22).



**What is Map Reduce ?**

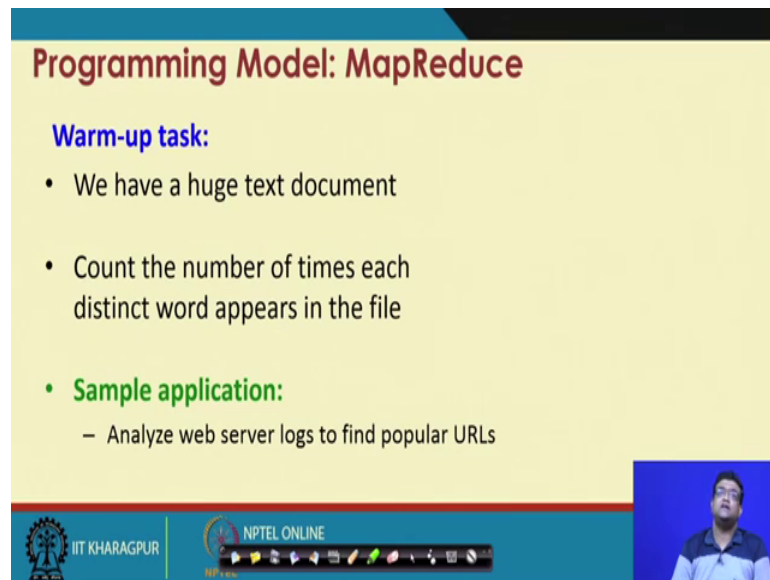
- Method for distributing a task across multiple servers.
- Proposed by Dean and Ghemawat, 2004.
- Consists of two developer created phases:
  - Map
  - Reduce
- In between Map and Reduce is the Shuffle and Sort phase.
- User is responsible for casting the problem into map – reduce framework.
- Multiple map-reduce jobs can be “chained”.

IIT KHARAGPUR | NPTEL ONLINE

So, next is, what is map reduce; so, map reduce is a programming paradigm. It is a programming paradigm, which is used for distributing a task across multiple servers. It was proposed by Dean and Ghemawat it 2004. So, it consists of two developer created phases the map and the reduce

In between the map and the reduce phase, there is a shuffle and a sort phase. So, the user, given a problem, the user is responsible for casting the problem into this map reduce framework. So, this is where the map reduce programming paradigm comes into picture which, we will learn in this lecture. And also note that, typically multiple map reduce jobs can be chained or scheduled to execute one after the other.

(Refer Slide Time: 15:43).



**Programming Model: MapReduce**

**Warm-up task:**

- We have a huge text document
- Count the number of times each distinct word appears in the file

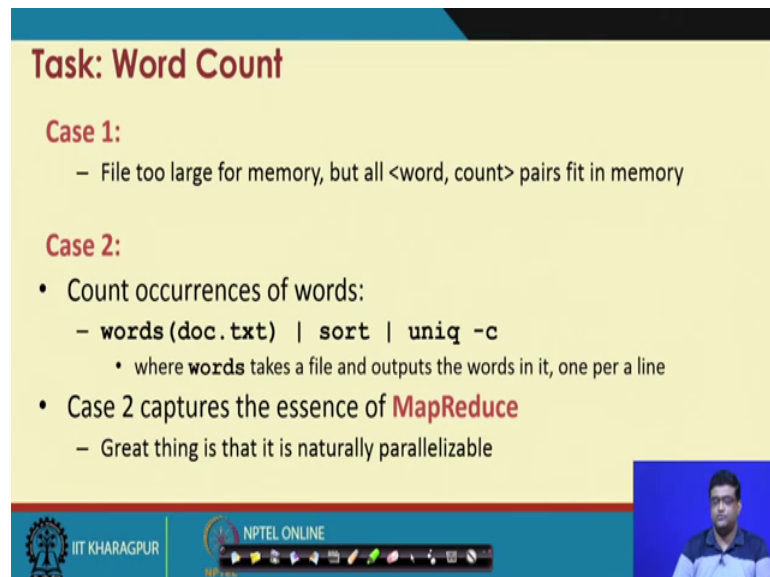
**Sample application:**

- Analyze web server logs to find popular URLs

IIT KHARAGPUR | NPTEL ONLINE

So, let us consider a huge text document. So, in order to describe the map reduce programming model; let us consider, a huge text document and your task is to count the number of times each distinct word appears in the file. So, for example, an example application could be that you analyse web server logs to find popular URLs.

(Refer Slide Time: 16:16).



**Task: Word Count**

**Case 1:**

- File too large for memory, but all <word, count> pairs fit in memory

**Case 2:**

- Count occurrences of words:
  - `words(doc.txt) | sort | uniq -c`
    - where `words` takes a file and outputs the words in it, one per a line
- Case 2 captures the essence of **MapReduce**
  - Great thing is that it is naturally parallelizable

IIT KHARAGPUR | NPTEL ONLINE

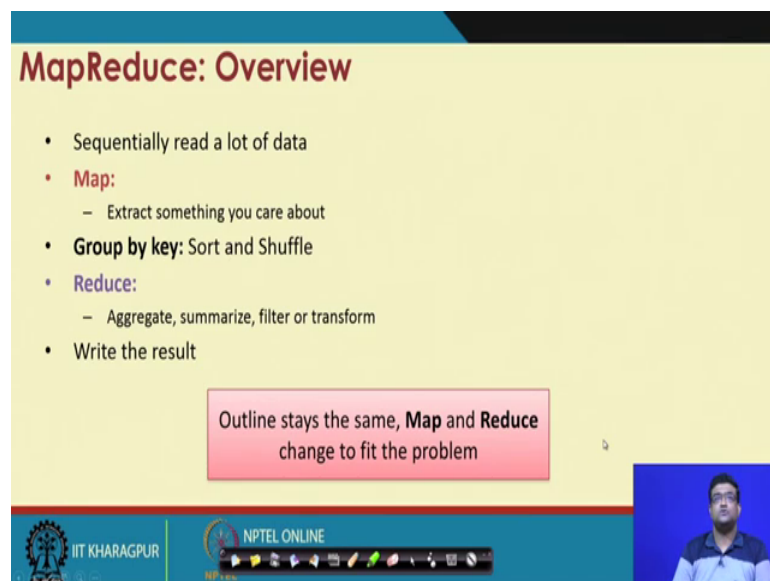
So, you can have two situations. The first situation is that, even though the file is too large, you store, you can store all the words in memory. So, you store the count

corresponding to every word in memory and you update these counts, you make one pass through the file and you update these counts in memory.

So, this, so this creates many problems. So, typically it is not possible to store all the words for example, by words in the previous application it could be all the URL. So, you may not be able to store all the URLs in the memory of a single computer. Another problem is, if you are using multiple computers how do you ensure that the counts are consistent across multiple computers.

So, the second approach is the following approach that you count for each word, you create a stream of words, and then you sort this stream using the `uniq -c` sort commands and then you count the number of `uniq -c` words in this stream. So, this approach is more similar to the map reduce approach of doing things.

(Refer Slide Time: 18:04).



The slide is titled "MapReduce: Overview" and lists the following steps:

- Sequentially read a lot of data
- **Map:**
  - Extract something you care about
- **Group by key:** Sort and Shuffle
- **Reduce:**
  - Aggregate, summarize, filter or transform
- Write the result

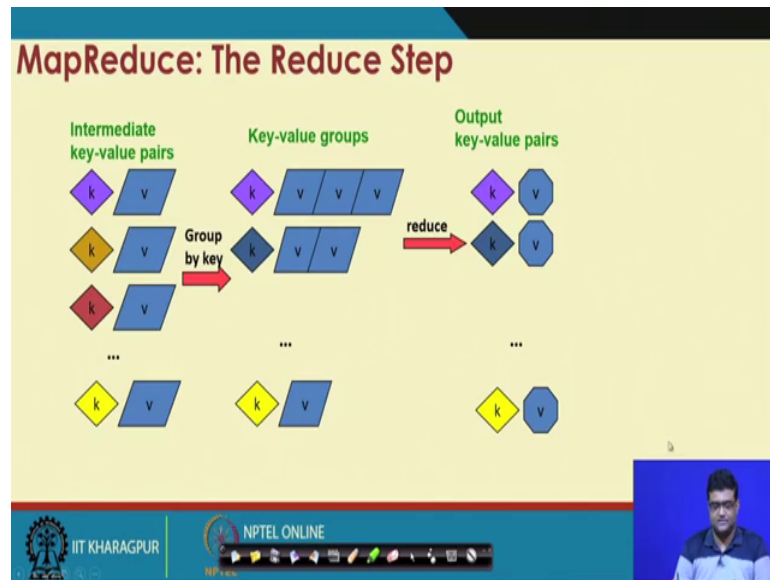
A pink box in the center of the slide contains the text: "Outline stays the same, **Map** and **Reduce** change to fit the problem".

The slide footer includes the IIT KHARAGPUR logo, the NPTEL ONLINE logo, and a small video feed of the presenter.

So, what is the idea; the idea is that you have two phases. One is the Map phase and one is the Reduced phase. So, in the map phase you extract something that you care about. So, in this case you as you read the document you extract the words and the counts of those words, and in the reduce phase you aggregate or summarise. So, the reduce phase, assume for example, that in this case, all the words appear together. So, that you can easily aggregate or summarise.

So, this task of making all the words together is handled by the sort or the shuffle phase. Now, this is just one problem. So, the outline remains the same, but you can change it to fit it to many problems.

(Refer Slide Time: 19:07).




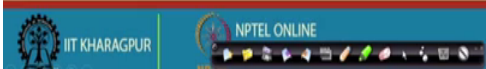
So, this is the functioning of the map reduce framework . So, as you can see. So, on the left here, you have many intermediate values. So, you have many intermediate values here which are the keys and the values and what the group by key or the sort and Shuffle shuffle phase does, is it brings together, records all records with the same key.

So, here you can see all the records with the same key are together and the values are, for each of those records are stored in a list and then finally, the reduce phase generates, one value for each of the keys or more formally for any map reduce program, there are two methods which the programmer must specify.

(Refer Slide Time: 20:10).

### More Specifically

- **Input:** a set of key-value pairs
- Programmer specifies two methods:
  - **Map(k, v)** →  $\langle k', v' \rangle^*$ 
    - Takes a key-value pair and outputs a set of key-value pairs
      - E.g., key is the filename, value is a single line in the file
    - There is one Map call for every (k,v) pair
  - **Reduce(k', <v'>\*)** →  $\langle k', v'' \rangle^*$ 
    - All values v' with same key k' are reduced together and processed in v' order
    - There is one Reduce function call per unique key k'



The first is the mapper method. The input to which is a key value pair and the output is a list of key value pairs. This mapper output is sorted and shuffled and fit into the reducer input. So, the reducer input, is again a key and a list of values pair. This is in map reduce.

So, as you get for each key a list of values you can; so, the reducer function aggregates the list of values and produces a new key and a value. So, this is the, example of word count or this is showing the example a running of word count for using the map reduce. So, here the mapper function, takes the record or takes the document.

(Refer Slide Time: 21:10).

### MapReduce: Word Counting

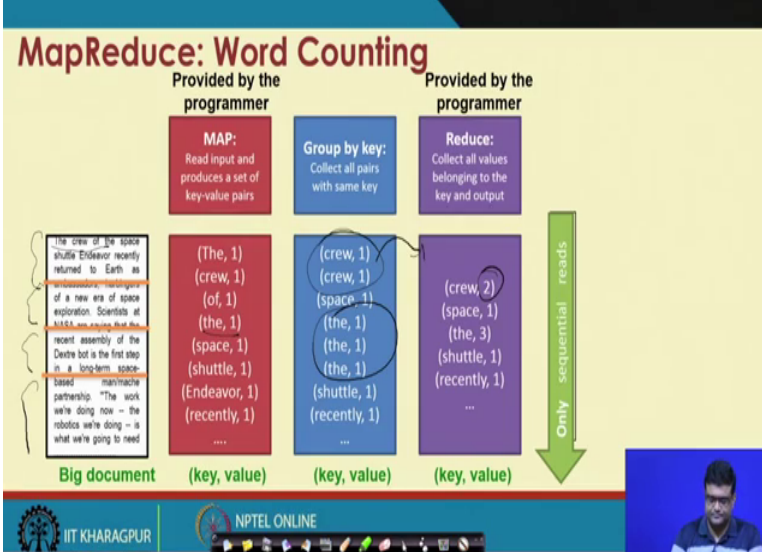
Provided by the programmer

**MAP:**  
Read input and produces a set of key-value pairs



**Group by key:**  
Collect all pairs with same key

Provided by the programmer

**Reduce:**  
Collect all values belonging to the key and output



Big document (key, value) (key, value) (key, value)

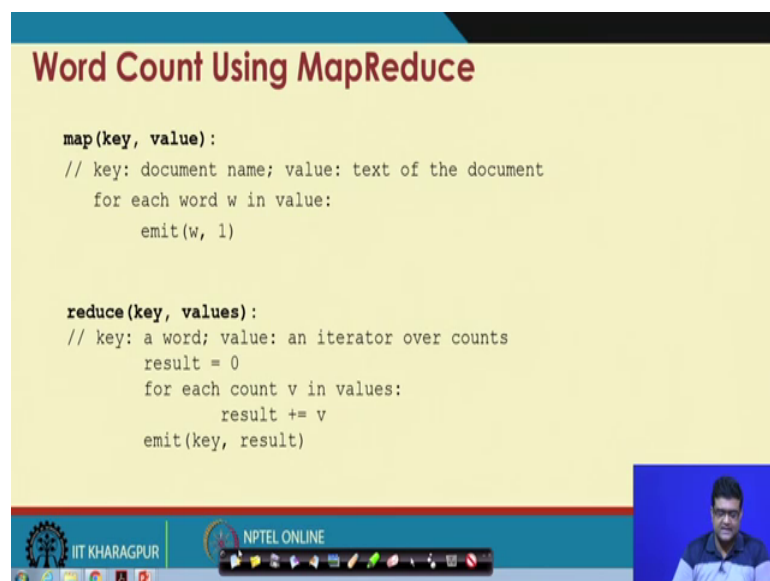


The documents it is typically broken down into records. So, for example, here you can see that, this is the first record, this is the second record, this is the third record, and this is the fourth record for this particular application. It does not matter, how the document is broken down so, long as, a word appears in one document. Sometimes the documents are broken down, using new line or something like that.

So, then the mapper function produces, the key value pair, for each document, but note that the keys can appear in any order. For example, the comma 1, crew comma 1 is coming from the first record and then there is a, the in the second record, somewhere. So, it is appearing at a later point it.

Now, the shuffle or the group by key phase brings, together all the crew brings together all the crew, and all the words, all the occurrences of those words together. So, then the reducer input record, gets this as input and it computes that it aggregates the two 1s in the crew to come up with this 2.

(Refer Slide Time: 23:19).



### Word Count Using MapReduce

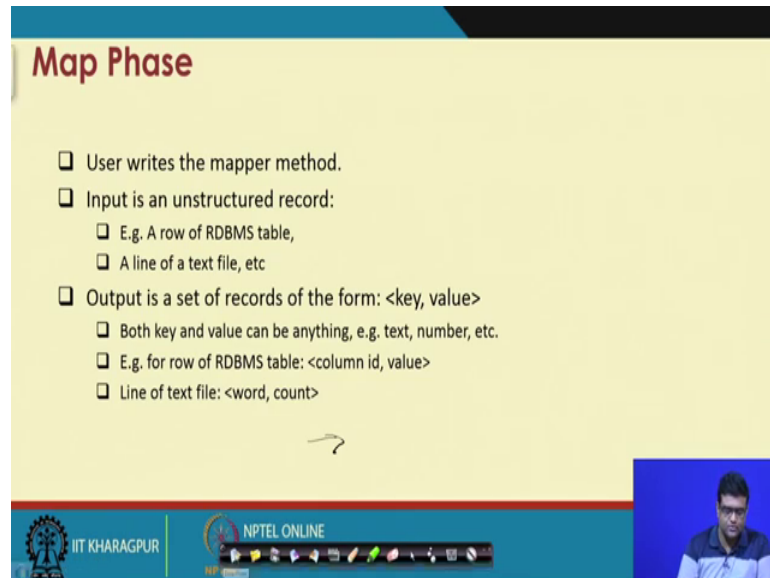
```
map(key, value):  
  // key: document name; value: text of the document  
  for each word w in value:  
    emit(w, 1)  
  
reduce(key, values):  
  // key: a word; value: an iterator over counts  
  result = 0  
  for each count v in values:  
    result += v  
  emit(key, result)
```

IIT KHARAGPUR NPTEL ONLINE

So, this is the map reduce program. So, once you get the document, the mapper record is emit w comma 1 and for each word in the, in the current value which contains the actual text, for the mapper record and the key is simply ignored. Key is something like a document name.

For, the reducer record, the key is a word which is very important and the value is the count over the, rather the value is a list of counts of this particular words. And in this case, all you have to do is iterate over all the values which come with this particular key and you just have to add them to the result and then you emit the sum. So, this is your word count program in map reduce.

(Refer Slide Time: 24:30).



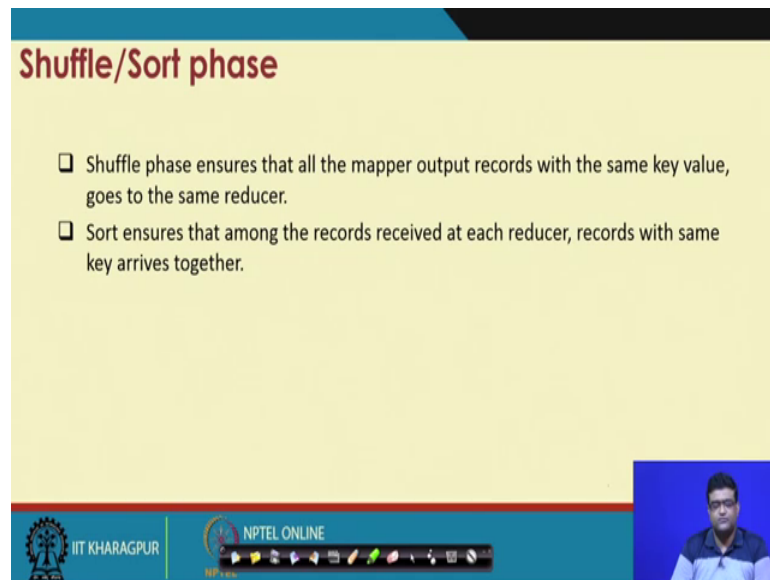
**Map Phase**

- User writes the mapper method.
- Input is an unstructured record:
  - E.g. A row of RDBMS table,
  - A line of a text file, etc
- Output is a set of records of the form: <key, value>
  - Both key and value can be anything, e.g. text, number, etc.
  - E.g. for row of RDBMS table: <column id, value>
  - Line of text file: <word, count>

The slide also features a video inset of a presenter in the bottom right corner and logos for IIT KHARAGPUR and NPTEL ONLINE at the bottom.

So, just to summarise, the user writes the mapper and the reducer method. The input is typically, input to a mapper is typically unstructured record for example, a row of RDBMS table or a line of text file etcetera. Output is a set of records of form key value pair where both key and value can be anything. And so, for example, in case of word count the output record was word comma count.

(Refer Slide Time: 25:24)



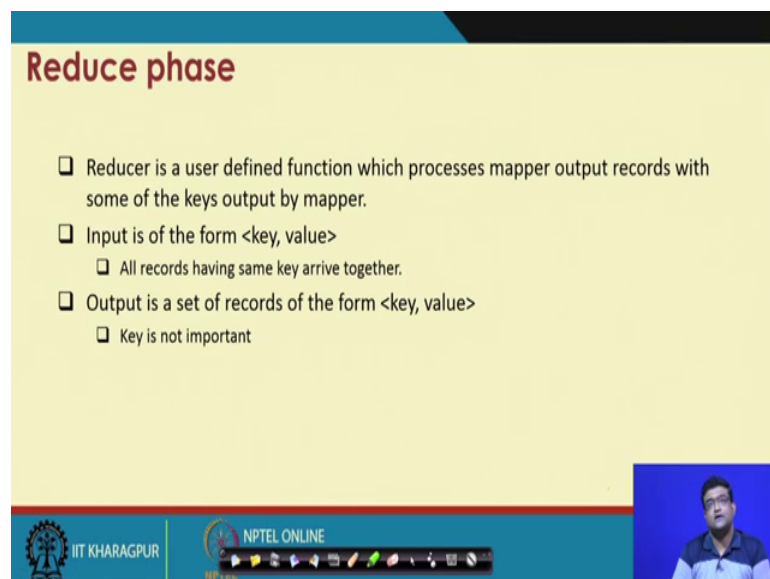
### Shuffle/Sort phase

- ❑ Shuffle phase ensures that all the mapper output records with the same key value, goes to the same reducer.
- ❑ Sort ensures that among the records received at each reducer, records with same key arrives together.

IIT KHARAGPUR | NPTEL ONLINE

The shuffle phase ensures that the mapper output records with the same key value goes to the same reducer or goes into the same reducer record. And sort ensures that, the records which with the same key arrive together.

(Refer Slide Time: 25:52).



### Reduce phase

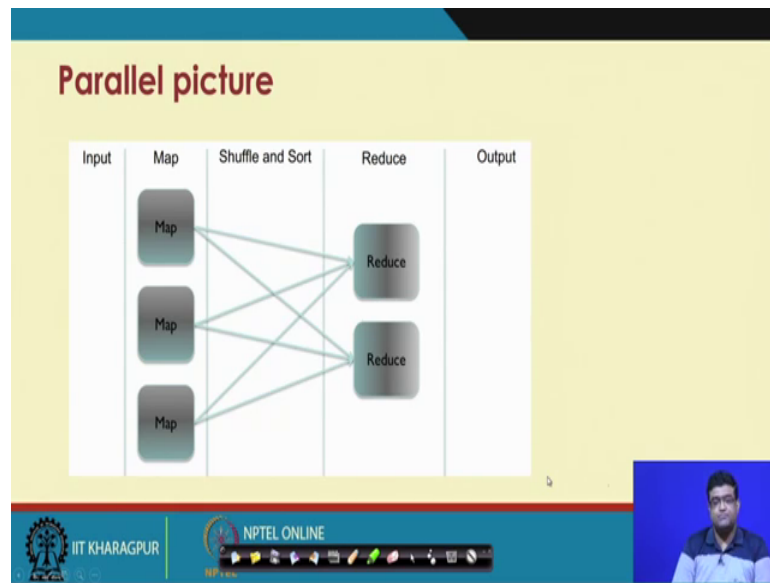
- ❑ Reducer is a user defined function which processes mapper output records with some of the keys output by mapper.
- ❑ Input is of the form <key, value>
  - ❑ All records having same key arrive together.
- ❑ Output is a set of records of the form <key, value>
  - ❑ Key is not important

IIT KHARAGPUR | NPTEL ONLINE

In the reducer again, the reducer is a user defined function which, which processes the mapper output records which are output by the mapper. So, the input is of the form key comma list of values and all records having the same key arrive together and then output contains the key and the aggregated value of each record.

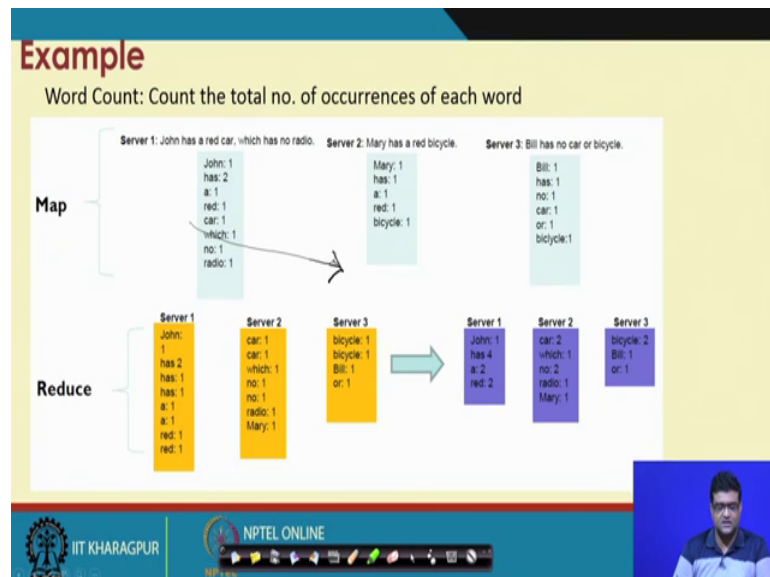


(Refer Slide Time: 26:32).



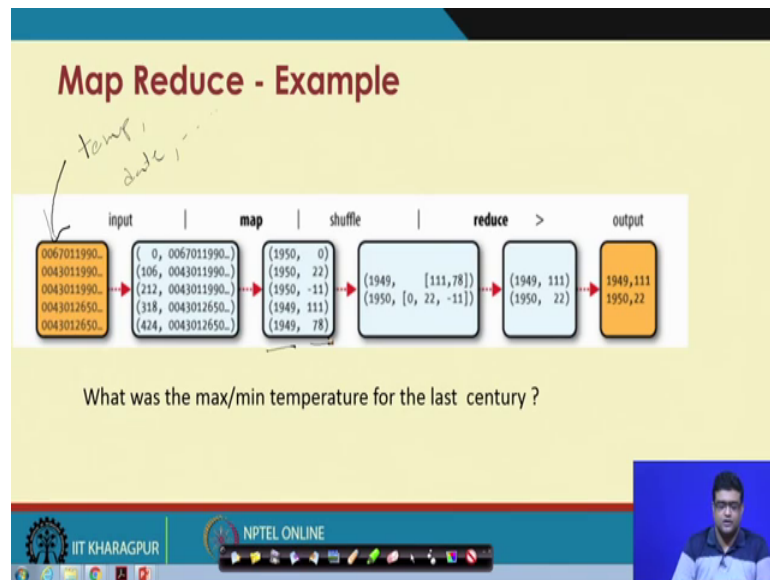
So, this is the parallel picture of map reduce. As you can see, since, there are many many servers. So, many many mapper programs will be executing at the same time, and each will generate many many records with different keys. So, Shuffle and Sort makes sure that a particular record with a key goes to the corresponding reducer.

(Refer Slide Time: 27:13).



And this is the word count example for another case.

(Refer Slide Time: 27:16).



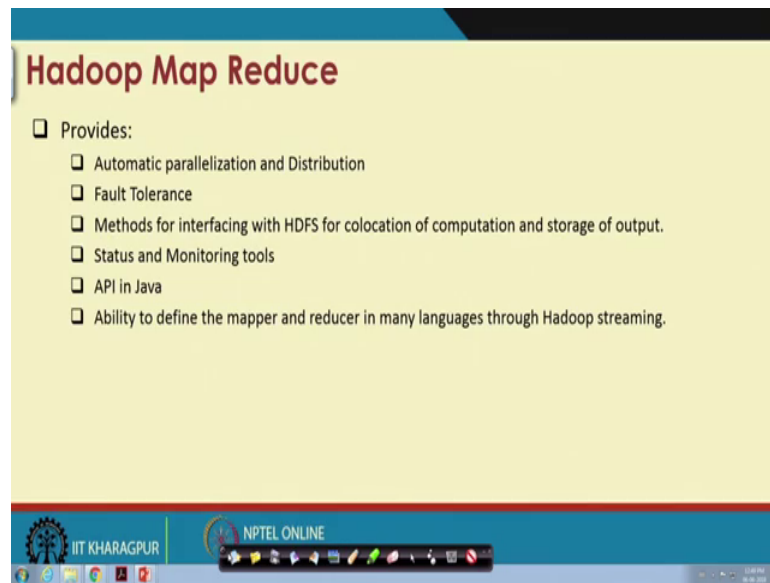
Another example application is where let us say, you have a huge file with maximum and minimum temperature for every weather station and every date as record. So, your input records are of the form. So, your input records here, are of the form temperature and date and maybe other attributes such as weather station etcetera. And your task is to find the maximum and minimum temperature, for the last century.

So, what should the mapper task be? So, the mapper task should take this record, as the input and output the year and the temperature for the particular year as the mapper outputs. The mapper input record is the, is the weather station data where you have temperature and the whole lot of other attributes.

The mapper output record only outputs the year which is the year. So, which is the key or the mapper output key and the temperature for that particular year. So, for every year you will have multiple mapper output records.

Now, Sort and Shuffle sort sit into records of this form where for every year you have a list of temperature values and each of these sorted records go to a reducer. The reducer takes in this record and for each year computes the maximum and the minimum recorded in that particular year which is the output. So, this is another example where you can find in a large data set, the maximum and minimum temperatures using the map reduce framework.

(Refer Slide Time: 30:21).



## Hadoop Map Reduce

- ❑ Provides:
  - ❑ Automatic parallelization and Distribution
  - ❑ Fault Tolerance
  - ❑ Methods for interfacing with HDFS for colocation of computation and storage of output.
  - ❑ Status and Monitoring tools
  - ❑ API in Java
  - ❑ Ability to define the mapper and reducer in many languages through Hadoop streaming.

IIT KHARAGPUR NPTEL ONLINE

So, in a nutshell, what does Hadoop map reduces provide? So, Hadoop map reduces provides automatic parallelization and distribution of the program that you have to write. So, you do not have to write either code for parallelization or distribution of the program. So, you do not have to write how each distributed component is calculated, how many mapper jobs should be spawned, how many reducer jobs should be spawned, all these things are taken care of by the Hadoop map reduce framework.

It provides fault tolerance. So, for example, if you write a program for computing let us say word count program and it is run on a thousand computers. And now if one of those computers fails, it will automatically make sure that the other 999 computers finish executing the whole task; even though or rather the whole job even though one of the subtasks has failed; because that particular computer has failed.

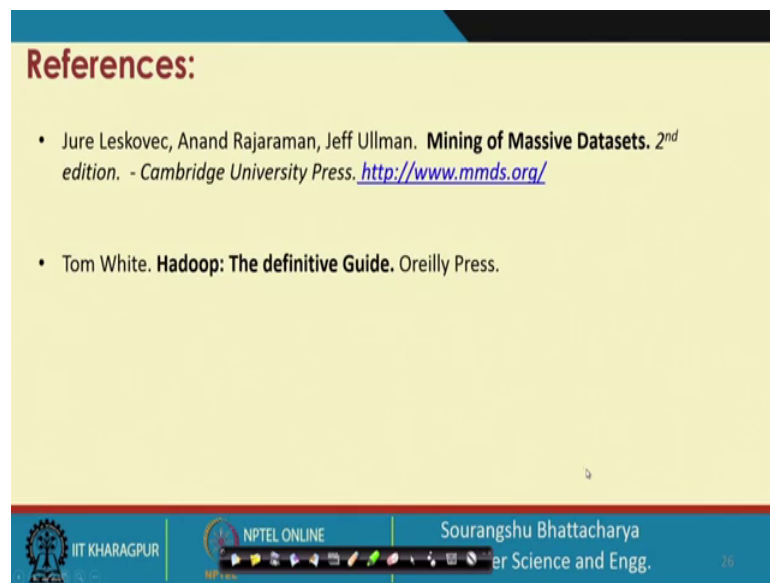
It provides methods for interfacing with the HDFS. So, reading the large amount of data from Hadoop distributed file system and collocation of computation. So, the fact, that the computation should be scheduled on a node, where the HDFS is storing a particular piece of data. That facility is also taken care of by Hadoop map reduce framework and it also stores the output in an appropriate location.

Additionally, it provides status monitoring status and monitoring tool. So, if a number of jobs are submitted to a Hadoop map reduce Cluster it provides tools to see what is the

status of a particular job, how many mapper tasks are running, how many reducer tasks are running and what is the status of each mapper task and reducer task .

It provides an API in java and also in many other languages now, and also it provides AP, API for through Hadoop streaming for using many many other languages. So, you can provide your mapper or reducer function in, for example, Python or, Pearl or any other language.

(Refer Slide Time: 33:25).



**References:**

- Jure Leskovec, Anand Rajaraman, Jeff Ullman. **Mining of Massive Datasets**. 2<sup>nd</sup> edition. - Cambridge University Press. <http://www.mmds.org/>
- Tom White. **Hadoop: The definitive Guide**. Oreilly Press.

IIT KHARAGPUR NPTEL ONLINE Sourangshu Bhattacharya  
er Science and Engg. 26

So, part of this lecture was from this book which is also available online on mining of massive datasets and another very nice reference is Hadoop: The Definitive Guide .

Thank you.