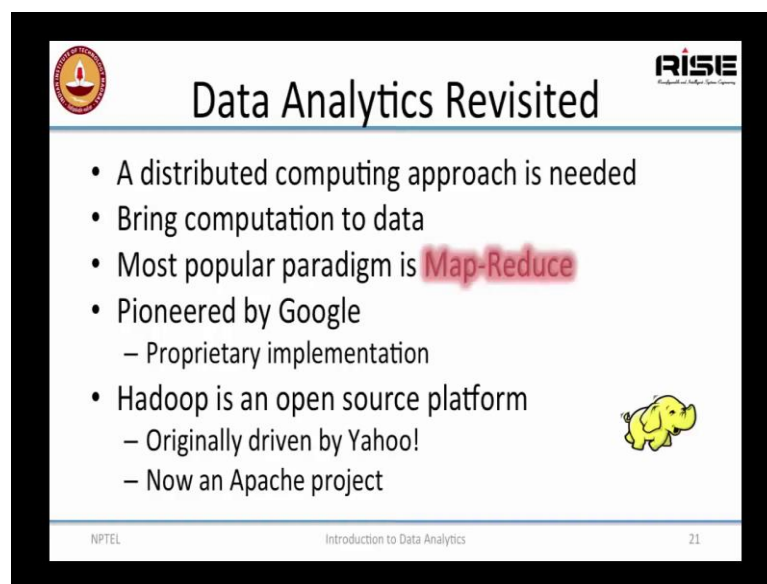


Introduction to Data Analytics
Prof. Nandan Sudarsanam and
Prof. B. Ravindran
Department of Management Studies and
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Module - 08
Lecture - 40
What is Big Data?
A small introduction

(Refer Slide Time: 00:14)

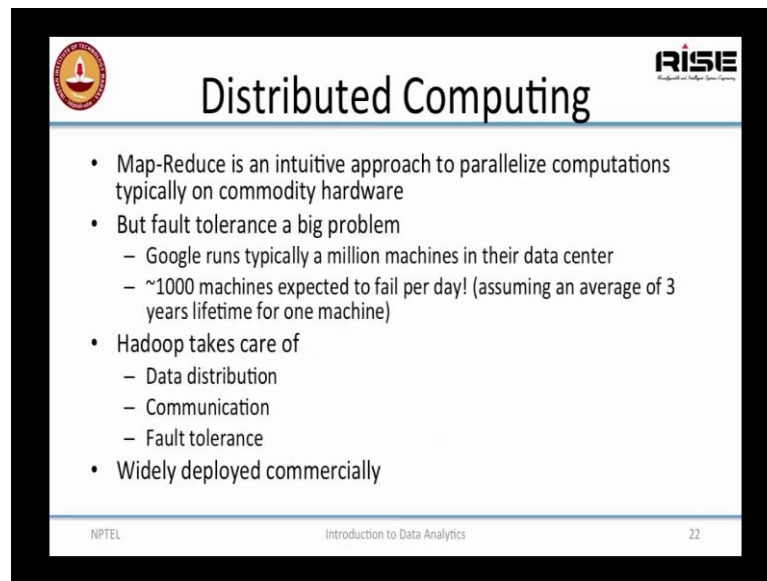


The slide is titled "Data Analytics Revisited" and features a list of bullet points. In the top left corner is the IIT Madras logo, and in the top right corner is the RISE logo. A small cartoon elephant is positioned to the right of the bullet points. The footer contains the NPTEL logo, the text "Introduction to Data Analytics", and the slide number "21".

- A distributed computing approach is needed
- Bring computation to data
- Most popular paradigm is **Map-Reduce**
- Pioneered by Google
 - Proprietary implementation
- Hadoop is an open source platform
 - Originally driven by Yahoo!
 - Now an Apache project

Hello and welcome to the 2nd module on Big Data. That is we saw earlier that the variety and the volume of data; requires has to have a new paradigm for handling all the computing. And one of the most popular paradigm that is used for handling big data computation is Map-Reduce. So, this was a programming model that was pioneered by Google which were reusing initially proprietary implementation. But later on a Hadoop an open source platform, that implements Map-Reduce became very popular. And so, in the next few slides I will give you a very brief on introduction to Map-Reduce.

(Refer Slide Time: 01:04)

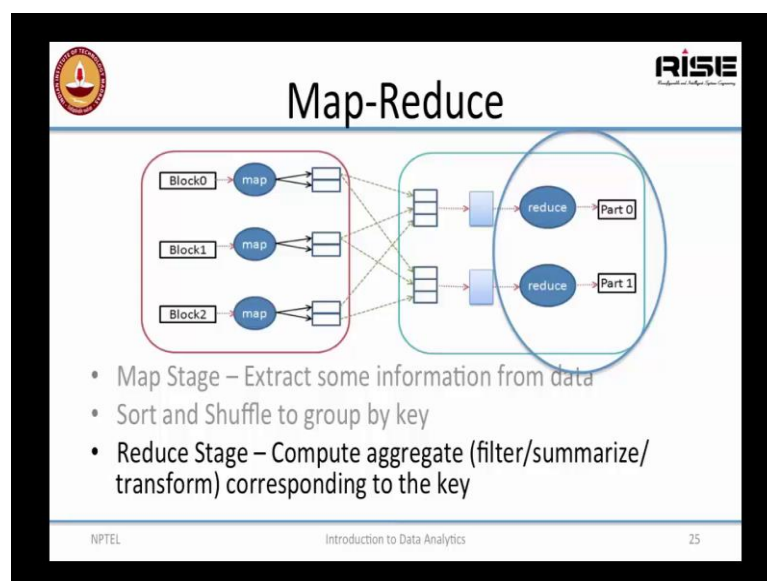


The slide is titled "Distributed Computing" and features a list of bullet points. In the top left corner is a circular logo with a lamp, and in the top right corner is the "RISE" logo. The bottom of the slide contains the text "NPTEL", "Introduction to Data Analytics", and the number "22".

- Map-Reduce is an intuitive approach to parallelize computations typically on commodity hardware
- But fault tolerance a big problem
 - Google runs typically a million machines in their data center
 - ~1000 machines expected to fail per day! (assuming an average of 3 years lifetime for one machine)
- Hadoop takes care of
 - Data distribution
 - Communication
 - Fault tolerance
- Widely deployed commercially

So, Map-Reduce as we will see, is a very intuitive approach to parallelize computation typically on commodity hardware. But the big problem when you are using, especially using commodity hardware is that machines tend to fail a lot. So, fault tolerance becomes a big problem. So, for example Google runs typically a million machines in one of the data centers and assuming on average 3 years lifetime for 1 machine. You would expect about 1000 machines to fail per day. So, Hadoop is a distribution frame work, that fundamentally uses Map-Reduce as that computing paradigm, but on top of it takes care of data distribution, communication, and fault tolerance; and makes it very convenient to use this kind of a distributed; set up and solving everyday problems. And hence Hadoop is being wildly deployed commercially, and variants of that are continuing to be very popular even today.

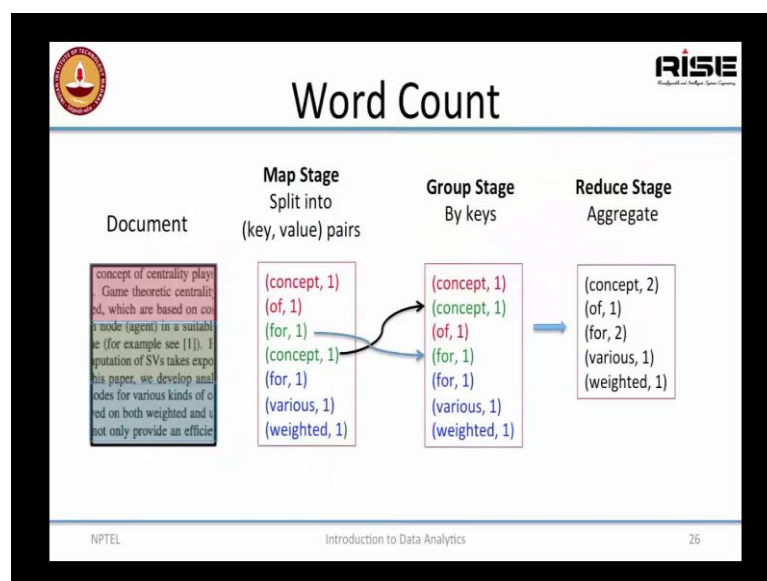
(Refer Slide Time: 02:17)



Let us look at the Map-Reduce computing model. The Map-Reduce as you can imagine consist of 2 stages: the map stage and the reduce stage. And in between there is a group and redistribute phase. So, the map stage, so the each mapper who runs on an individual machine takes a block of the data; that is you have to originally process, extracts some information from the data and output these as key value pairs. So, the key as you know, as identifier the value is any value that you would like to send along with the key. So, we will look at examples later.

And then before sending to the reduce stage, we sort and shuffle all of these key value pairs and you group them by the keys. So, that all the pairs which have the same key values, will go to the same reduce stage. In the reduce stage you essentially compute aggregate statistics corresponding to the key. We could add things up; we could summarize them, we could transform them, we could do any kind of filtering that you want based on the key, whatever it is we can compute aggregate values based on the keys.


(Refer Slide Time: 03:45)




Look at this an example of a very simple word count, it is like the hello world of the Map-Reduce programming. So, let us say that you have a document. So, what you would do is you would divide the document into different blocks as shown here. So, the 1st block I have colored it red, the 2nd block green and the 3rd block blue. And we send it to the different mappers. So, what the mapper does is? It reads each word in the document that outputs that as a key and the value as the count of the word in the document. So, every time it encounters a word, it just going to output it as that word with the count of 1. So, the red mapper is going to output; concept, one, of, one; and then the green mapper is going to output variety of things on this case is going to say for one, and concept one. And the blue marker again outputs for one, various one, weighted one so and so forth. So, in the group stage what happens, you group things by the keys.

So, all the key value pairs which have concept of the key get grouped and they are send to one reducer, likewise all the key value pairs which have for as a key get grouped and are send to another reducer and so on so forth. And finally, the reduce stage aggregates all these counts regardless of which mapper they come from and then outputs the total count. So, in this case you would get (concept 2) (of 1 for 2 various 1 and weighted 1. This is very simple way of doing word count and your document could be very large as long as we have sufficient number of machines we can compute this very efficiently. That is not only in working with documents and the other things. Even simple items, even simple computations is you would have thought they were straight forward can become complicated when you are dealing with it scale.

(Refer Slide Time: 06:01)



Computing Degree

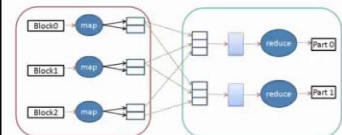


- On large graphs computing even degree of a node is hard!
- Data arrives one edge at a time
 - A called B at time t and spoke for m minutes
 - Customer C bought u units of item I
 - User X posted on the wall of user Y
 - Protein P was observed to interact with protein Q
- How to efficiently aggregate the edge events into a graph?

NPTEL Introduction to Data Analytics 27

So, here is an example I would like to compute degree of a node in a graph. On very large graphs computing even degree of a node becomes little harder. Why is that, because the data itself is not available to you at one time? Suppose I am trying to build the graph as of who called who. So, I am going to get things like A called B at time t and spoke for m minutes. So, that is an event that arrives to me after the call has finished. So, at no point of time do I have all the calls of A, stored in some place and we just going to find the degrees. It is not like a single graph that already being constructed firm. Or if you could think of trying to create some kind of an interaction graph on Facebook; so user x posted on the wall of users y . Now these events even if they are available to you post facto that are so numerous that aggregating them and to finding the degree of the graph is could take a while. So, you could actually use Map-Reduce to efficiently aggregate the each event into a graph.

(Refer Slide Time: 07:12)



Map-Reduce

Algorithm 1 Degree

Input: Edgelist of $G(V, E)$
Output: Degree


```
class MAPPER
  method MAP(n, edge)
    (nodeA, nodeB) = edge.split()
    EMIT(nodeA, 1)
    EMIT(nodeB, 1)

class REDUCER
  method REDUCER(node, values)
    sum = 0
    for all value in values do
      sum += value
    end for
    EMIT(node, sum)
```


NPTEL Introduction to Data Analytics 28

Here is a simple program that would do now. So, the mapper essentially takes each edge event, each event, each interaction could be the posting of a message on Facebook; could be the making of a call, takes each of those events and then it creates 2 key value pairs for every event. That has of A call B it will say, A 1 and B 1. and on the reducer essentially now takes in every event or every key value pair that have the same key; that essentially means is going to gather all the node A's. Add up the events corresponding to the node A it will output the degree of node A. Fairly simple, fairly straight forward and so you do not really have to create a adjacency matrix or adjacency list representation of your graph. We will be able to answer to queries like beginning. We can stick with the edge list representation and still answer these kinds of queries.

(Refer Slide Time: 08:19)



Data Analytics– revisited



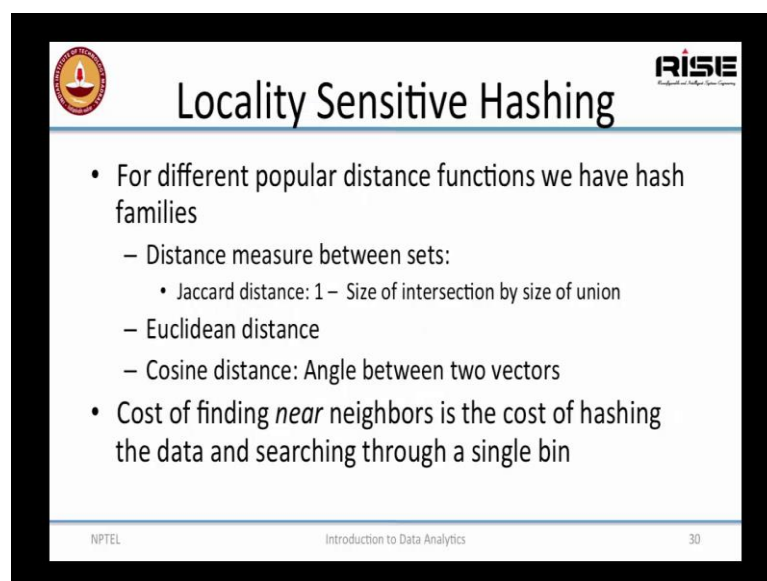
- k NN – Find nearest neighbors, but how?
 - Linear search in space impossible
 - Any index structure should be small to fit into memory
- New approach: Locality Sensitive Hashing (Broder 97)
- Find hash functions that:
 - If $d(x,y) < threshold$ then x and y hash to the same bin
 - If $d(x,y) > threshold$ then x and y hash to different bins
 - with high probability

NPTEL Introduction to Data Analytics 29

So, let us look at some other questions that not necessarily need Map-Reduce; but become harder when you are looking at large data. So, you looked at k nearest neighbors in one of the earlier modules. So, how would you find k nearest neighbors, if you have huge volumes of data? So, linear search is impossible; because any index structure should be small enough to fit into memory for you to do linear search. That is not going to happen if data is very large. So, there is a new approach for finding neighbors in data call locality sensitive hashing introduced by Andrei Z. Broder and others.

So basic idea here is to, find hash functions. So, you remember hash functions, they are functions that take a key as an input and then the hash it into one of n buckets. So, here what we do is? We want to find hash functions, such that 2 elements x and y , if their distance is less than a certain threshold the hash to the same buckets or same bin. Two elements, x and y ; if the distance is greater than a certain threshold, then x and y hash to different bins; and we would like this to hold with very high probability. We would like this to hold for sure, but then it is hard to get something that will work always. So, you would like to hold for this to hold with very high probability. So, now, if I want to find if y is nearest neighbor of x , then all I need to do is look through the bin into which x hashes.

(Refer Slide Time: 10:02)



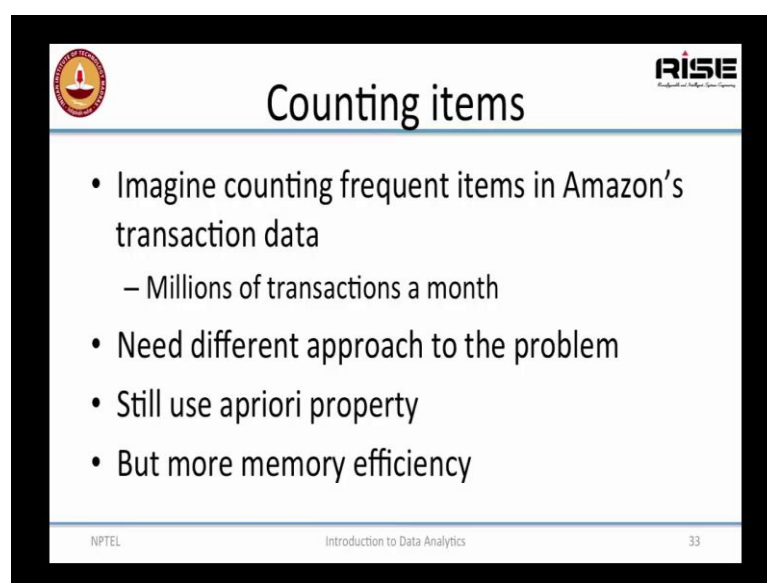
The slide is titled "Locality Sensitive Hashing" and features a list of bullet points. In the top left corner is a circular logo with a lamp, and in the top right corner is the "RISE" logo. The footer contains the text "NPTEL", "Introduction to Data Analytics", and the number "30".

- For different popular distance functions we have hash families
 - Distance measure between sets:
 - Jaccard distance: $1 - \frac{\text{Size of intersection}}{\text{Size of union}}$
 - Euclidean distance
 - Cosine distance: Angle between two vectors
- Cost of finding *near* neighbors is the cost of hashing the data and searching through a single bin

And since this is only with very high probability, so you might actually miss your nearest neighbors. But you will certainly get some neighbors that are close enough because since the data set large, even close neighbors are usually sufficient. So, depending on the kind of distance function that, you want to use on your data. If you remember we talked about different distance function in the various neighbor depending on the distance function you want to use on your data; you are going to have to define different hash functions. And so for example, if you want to look at distance measure between sets; this has groups of words and so on so forth. You probably like to usage the Jaccard distance, which is 1 minus the size of intersection by the size of union. Or if you could say Euclidean distance between points or between vectors you could want to see cosine distance; and for all of these people have worked out what are appropriate locality sensitive hash functions.

So, we are not going to get in the details of the locality sensitive hashing, just wanted to give you a feel for how hard it can be, when you are looking at large volumes of data. Even what you thought is simple operation become harder, when you are looking at large volumes of data.

(Refer Slide Time: 11:19)

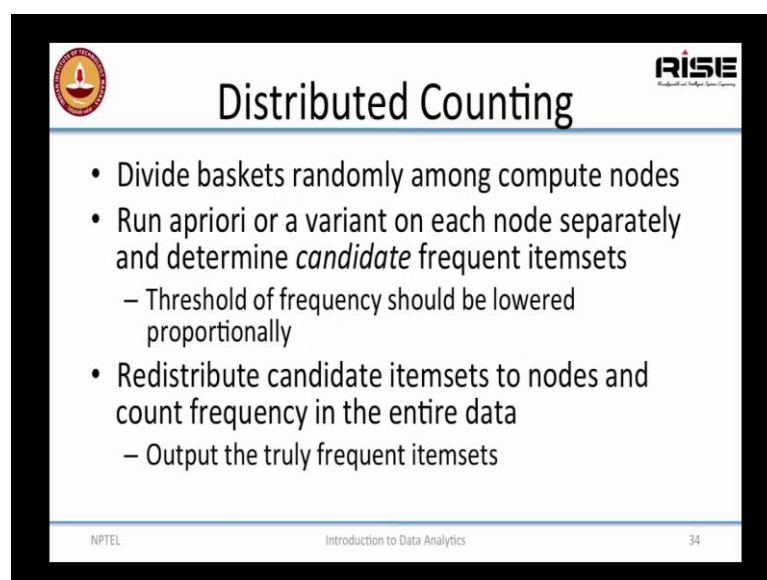


The slide is titled "Counting items" and features a list of four bullet points. The first bullet point is "Imagine counting frequent items in Amazon's transaction data", followed by a sub-point "– Millions of transactions a month". The second bullet point is "Need different approach to the problem". The third is "Still use apriori property". The fourth is "But more memory efficiency". The slide includes logos for NPTEL and RISE in the top left and right corners, respectively. The footer contains "NPTEL", "Introduction to Data Analytics", and the slide number "33".

- Imagine counting frequent items in Amazon's transaction data
 - Millions of transactions a month
- Need different approach to the problem
- Still use apriori property
- But more memory efficiency

Here is another example; we talked about frequent pattern mining and association rule mining. Imagine counting frequent items in amazon's transaction data; we have millions of transactions a month. So, just blindly running apriori is just not going to work. So, we need a different approach to the problem. We can still use apriori property, but we have to think of how you would handle the memory more efficiently.

(Refer Slide Time: 11:46)



The slide is titled "Distributed Counting" and features a list of three bullet points. The first is "Divide baskets randomly among compute nodes". The second is "Run apriori or a variant on each node separately and determine *candidate* frequent itemsets", followed by a sub-point "– Threshold of frequency should be lowered proportionally". The third is "Redistribute candidate itemsets to nodes and count frequency in the entire data", followed by a sub-point "– Output the truly frequent itemsets". The slide includes logos for NPTEL and RISE in the top left and right corners, respectively. The footer contains "NPTEL", "Introduction to Data Analytics", and the slide number "34".

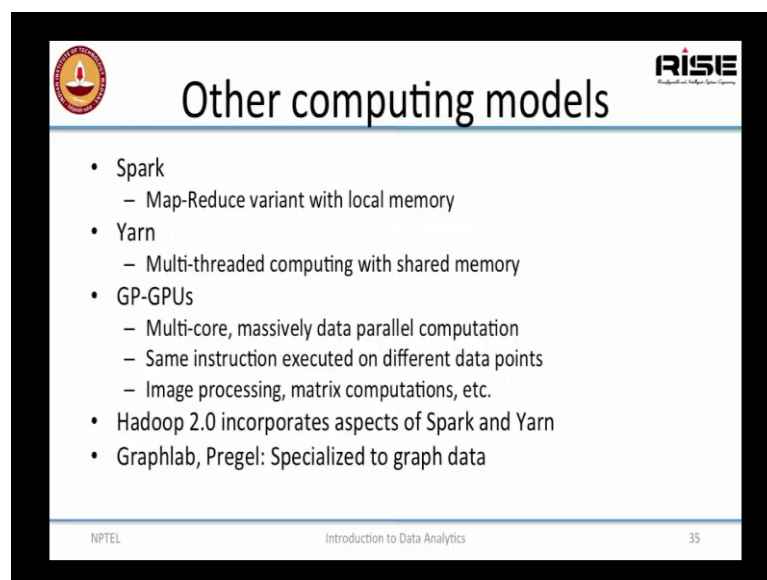
- Divide baskets randomly among compute nodes
- Run apriori or a variant on each node separately and determine *candidate* frequent itemsets
 - Threshold of frequency should be lowered proportionally
- Redistribute candidate itemsets to nodes and count frequency in the entire data
 - Output the truly frequent itemsets

So, here is a very simple approach, to do distribute counting; you divide baskets randomly among compute nodes. So, here we are talking about market basket data. So, essentially what I mean here is the transactions are randomly divided among all the nodes you have. You run apriori in each computing nodes separately. And whatever turns

up as frequent item sets in each of those nodes are now candidate frequent item sets. So, what we have to now do is, go back and count the actual frequency of these candidates item set; because the original candidates were determined on a small subset of the data. You will have to go back and count the frequency on the entire data to determine if these candidates set are frequent.

So, remember that when you are doing this in the distributed fashion the threshold that you are using for defining frequent item set should be lower. So, if you are splitting your data at 10 ways then the threshold that you had for frequency should be lowered by 10 times. So, once you have this candidate frequent item sets you do not have to count it in a single machine, you could still do this in a distributed fashion. So, again each node will count the frequency of just this one candidate item set and then at the reducer we could basically combine the frequency reported by each node, and then report frequency of the item set on the entire data.

(Refer Slide Time: 13:31)



The slide is titled "Other computing models" and features a list of distributed computing frameworks. It includes logos for NPTEL and RISE. The footer contains the text "NPTEL Introduction to Data Analytics 35".

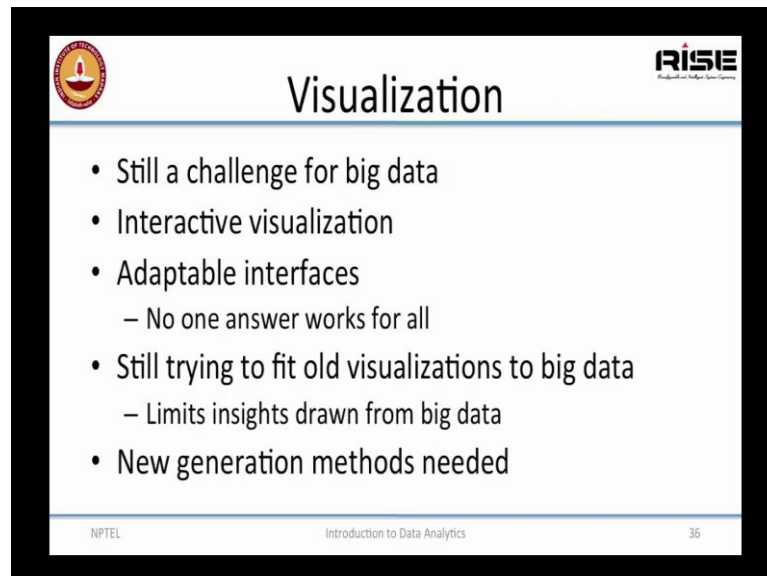
- Spark
 - Map-Reduce variant with local memory
- Yarn
 - Multi-threaded computing with shared memory
- GP-GPUs
 - Multi-core, massively data parallel computation
 - Same instruction executed on different data points
 - Image processing, matrix computations, etc.
- Hadoop 2.0 incorporates aspects of Spark and Yarn
- Graphlab, Pregel: Specialized to graph data

So, there are other computing models; I just spoke about Map-Reduce and this top. And the other thing has Spark which is Map-Reduce variant with local memory and then there are GP-GPUs, which are multi core massively data parallel computations. And then there are other models with shared memory multi core repetition. So, a depending on what is the use case that you have, we will have to use different computing models.

It is not that Map-Reduces one solution for all method. So, depending on the solution, depending on the problem that you have, you will have to pick the computing model that

the shows you.

(Refer Slide Time: 14:15)

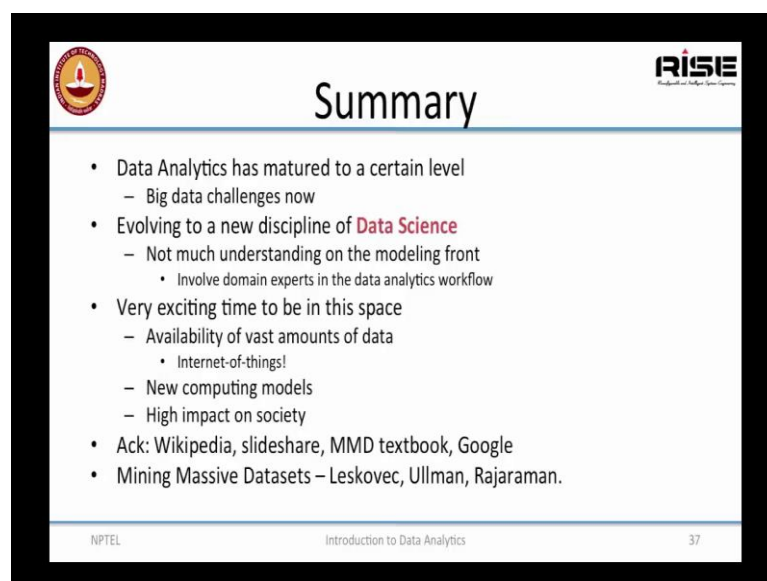


The slide is titled "Visualization" and features a list of challenges. It includes logos for NPTEL and RISE in the top corners. The footer contains the text "NPTEL Introduction to Data Analytics 36".

- Still a challenge for big data
- Interactive visualization
- Adaptable interfaces
 - No one answer works for all
- Still trying to fit old visualizations to big data
 - Limits insights drawn from big data
- New generation methods needed

One thing which I would like to point out is that data visualization is still a challenge for big data. So, visualization is a challenge for normal data analytics, but it is a challenge for big data. So, people are working on adaptable interfaces or interactive visualization, but more often than not people are still trying to fit old visualization ideas to big data and that is not working. So, it is very active area of the research and several new generation methods are needed here; I just wanted to draw your attention to the fact that it is something that you could work on.

(Refer Slide Time: 14:54)



The slide is titled "Summary" and features a list of points. It includes logos for NPTEL and RISE in the top corners. The footer contains the text "NPTEL Introduction to Data Analytics 37".

- Data Analytics has matured to a certain level
 - Big data challenges now
- Evolving to a new discipline of **Data Science**
 - Not much understanding on the modeling front
 - Involve domain experts in the data analytics workflow
- Very exciting time to be in this space
 - Availability of vast amounts of data
 - Internet-of-things!
 - New computing models
 - High impact on society
- Ack: Wikipedia, slideshare, MMD textbook, Google
- Mining Massive Datasets – Leskovec, Ullman, Rajaraman.

So, in summary; data analytics has matured to a certain level and so now, people are looking at big data challenges. So, it is some sense evolving to a new discipline of data science, where just the analytics techniques are not themselves sufficient, but we need more understanding from the modeling front. It is very exciting time to be in this space, availability of vast amounts of data and the internet of things like picking up as going to be a lot of work, more data available. The new computing models and that could very well be a high impact on society if you are able to come up with solutions, handle make sense of this big data.

That brings us to the end of this module.