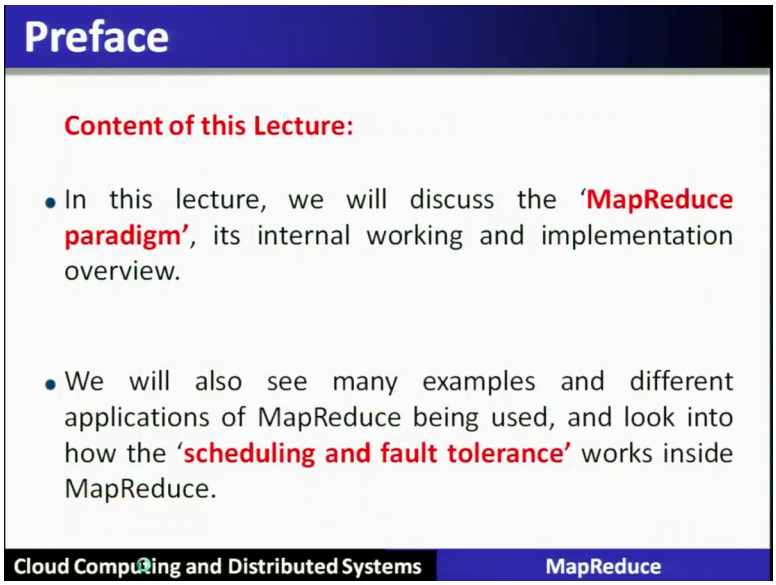
Cloud Computing and Distributed Systems
Dr. Rajiv Misra
Department of Computer Science and Engineering
Indian Institute of Technology, Patna

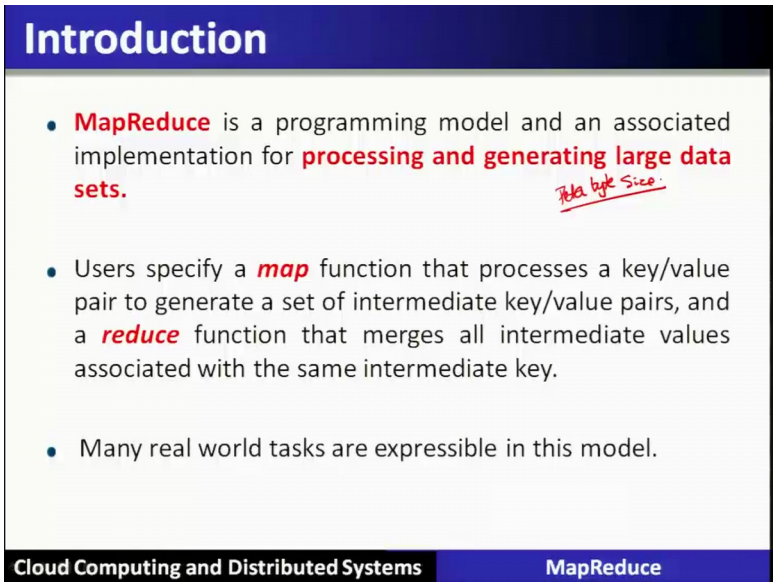
Lecture – 19
Map Reduce

(Refer Slide Time: 00:15)



MapReduce, preface content of this lecture we will discuss 'MapReduce paradigm' it is internal working and it is implementation overview. We will also see many examples, using MapReduce and different applications of MapReduce. And also we will look into the aspects how the scheduling and fault tolerance is done inside the MapReduce.

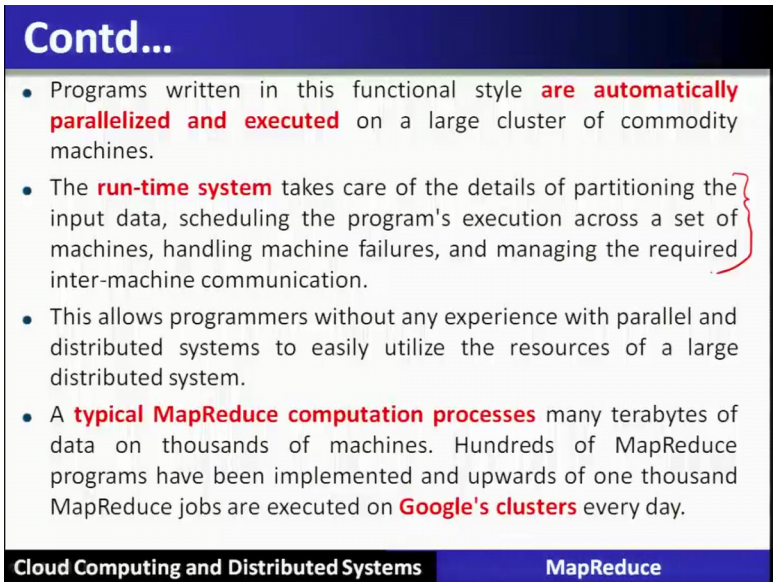
(Refer Slide Time: 00:42)



Introduction: MapReduce is a programming model and is associated and implementation for processing and generating large data sets. So, the large data sets we mean that if the data cannot be stored and processed with within one computer system that is in the range of petabytes size.

Therefore, we require a new paradigm that is called MapReduce, which we will enable to process such a large data sets, which are useful in today's different applications. So, here in this simple programming model users have to specify a function which is called a map that processes key value pair to generate a set of intermediate key value pairs, which in turn will be taken up by another function which is called a reduce that merges all the intermediate values associated with the same intermediate keys. And therefore, together map and reduce will work together to basically solve all the many applications in the real world. So, many real world tasks are expressible in this particular model.

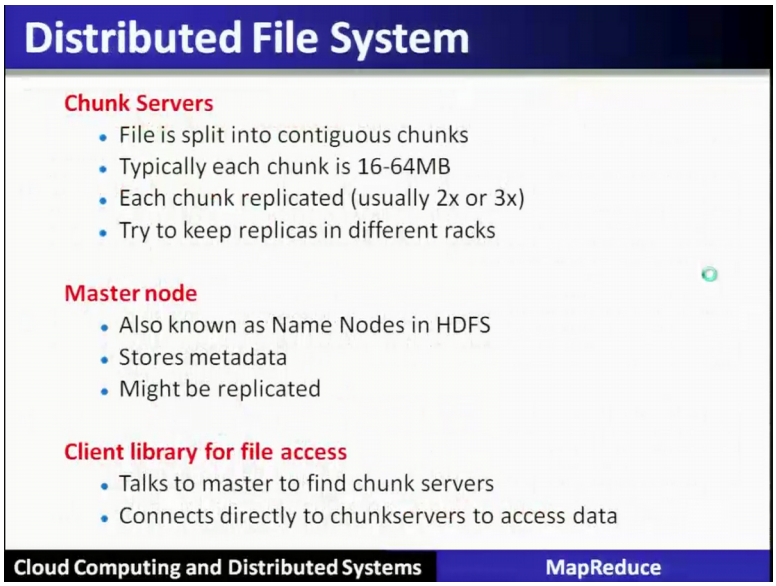
(Refer Slide Time: 02:18)



Now, programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run time system takes care of the details of partitioning the input data, scheduling the programs execution across the set of machines, handling machine failures, managing the required inter machine communication that is all will be made abstraction from the programmers. So, this allows the programmers without having any experience with the parallel and distributed system to easily utilize these resources for solving the problem which comprises of a large data set.

A typical MapReduce computation processes many terabytes of data on thousands of machines. So, that is the scale on which this MapReduce computation can work. Hundreds of MapReduce programs have been implemented and upwards of one thousand of MapReduce jobs are executed on Google's cluster every day.

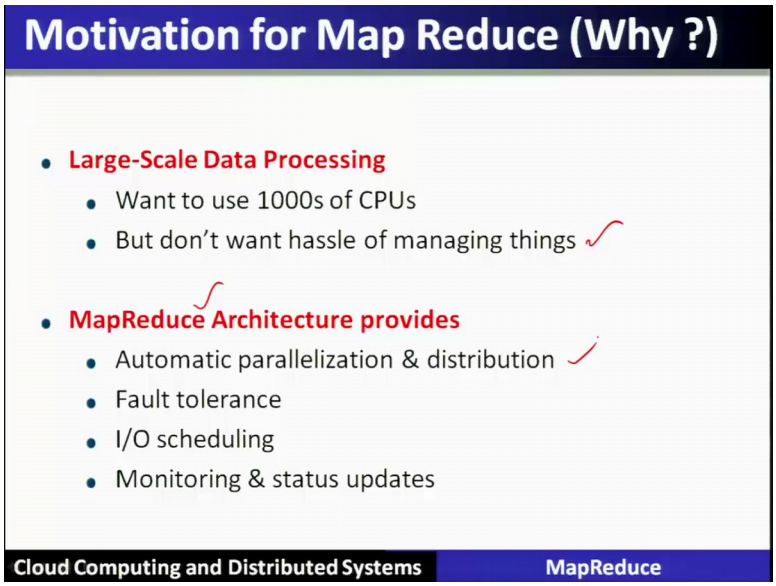
(Refer Slide Time: 03:29)



This MapReduce internally uses are distributed file system, which is known as HDFS and there are certain components which will be useful in the design. So, we will discuss this HDFS in a separate lecture, but for this point of discussion at this point of time we will understand that the distributed file system has the master node which also known as the name node in HDFS. And this stores the metadata and these may be replicated as per different file system and here we are using the HDFS file system.

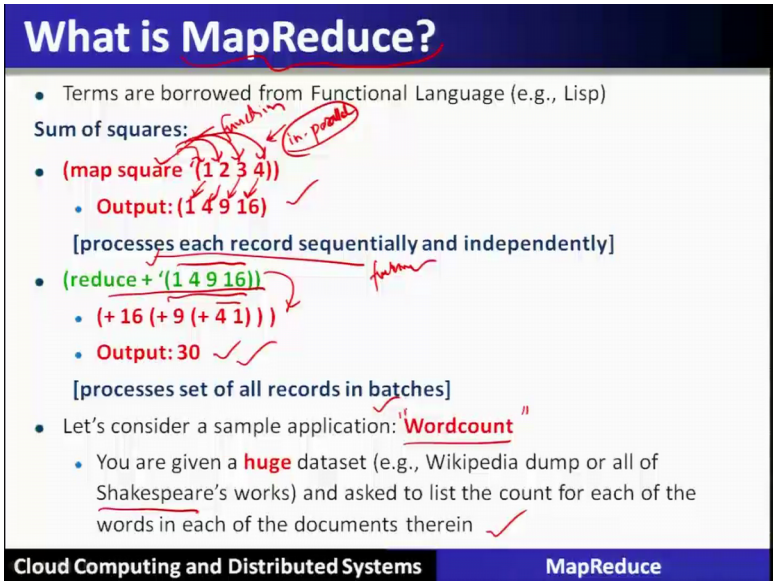
There are some client libraries for the file accesses. So, they will talk to the master to find out the available chunk servers, which connects directly to the chunk server to access the data. So, chunk servers are the servers who manages the splits into the contiguous chunks. And each chunk is of the size 16-64 MB and these chunks are replicated normally by default it is 3 times replicated in HDFS file system. This replication is also ensures that these replication also goes into different racks. So, that it will become a rack tolerant, rack fault, tolerant system it can build up in this process of replication.

(Refer Slide Time: 05:16)



So, let us see the motivation of using MapReduce for large scale data processing. So, to handle this scale of data processing we required 1000s of CPUs and also we do not want the hassle of managing such a large system. Therefore, the MapReduce is a programming model and it is architecture will provide automatic parallelization. And the distribution the fault tolerance and I O scheduling monitoring and status updates we will see their more details in further slides.

(Refer Slide Time: 06:04)



So, MapReduce paradigm let us understand from the scratch about this paradigm. These terms map and reduce their borrow from the functional language which is called a lisp. So, for example, the map function which is there in the list, if it is having the operation or a function called square and a Lisp is given. Then this particular function will be applied on each and every element of this particular list.

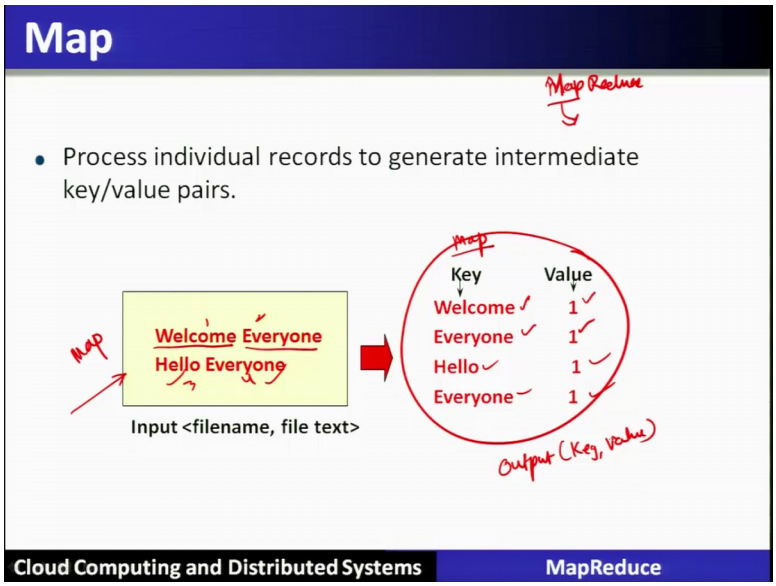
Now this particular function can be executed in parallel why because, this particular function if it is applied on these input values they can be able to generate the output that is the square of 1 is 1 square of 2 is 4 3 is 9 4 is 16. So, if four different machines are involved or four different threats are executing this is square at four different elements. So, in one go it will be able to generate this.

So, that is how the inherent parallelism is being exploited here in the functional languages and that is how the MapReduce will also do the same way without botheration of internally how this parallel programming is to be done by the programmer end. Similarly there will be a reduce function so, in the reduce function there is an operation that is the function which is defined.

So, function is a plus which involves all the elements, which are to be added and given the output. So, internally it will be represented in this particular equation form and two operators at a time because, plus is a binary operation two at a time this will form a complete expression and this will be evaluated to give the output 30. So, these map and reduce functions which are there in functional language like lisp, it is being inspired and it is not that map and reduce of lisp function, but it is a separate construct which is called a MapReduce which is now available to solve this kind of problem.

So, the problems of application for example, a large document large data set for example, it is a dump of a Wikipedia. And here you want to find out the word Shakespeare or some other keyword and you want to list the count for each of these words in a given document there in if that is the application. Then a MapReduce construct will be very easy and its program is called a word count. So, let us see how the word count program will work, where in we can submit a huge data set and using the word count program of MapReduce it can able to solve.

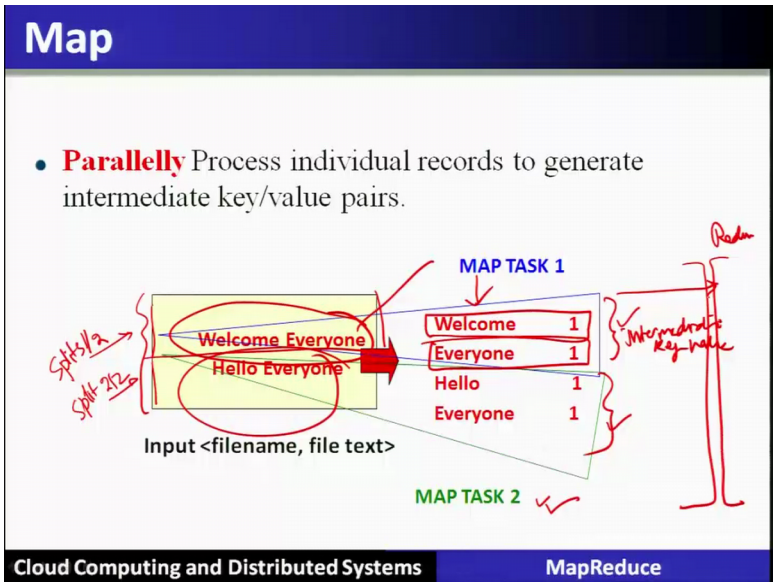
(Refer Slide Time: 09:35)



So, now in the MapReduce there are two functions map and reduce. So, out of them the map function let us discuss first. So, the process individual record to generate the intermediate key value pairs so, take this snapshot of the entire big document, let us say it has four different words in this particular sentence that is welcome everyone hello everyone. Now this particular document when it is given as the input. Then this map function should be able to generate the key value pairs. So, key means every keyword it will generate and it is count it will emit in the map that value of 1. So, for every word it will emit 1 count.

Similarly, hello also will emit 1 count and everyone is appearing again so, it will be generating that particular count so that is the function of map. So, map will produce this as an output. So, when map is given this particular filename, it will generate this particular output that is in the form of a key value pair.

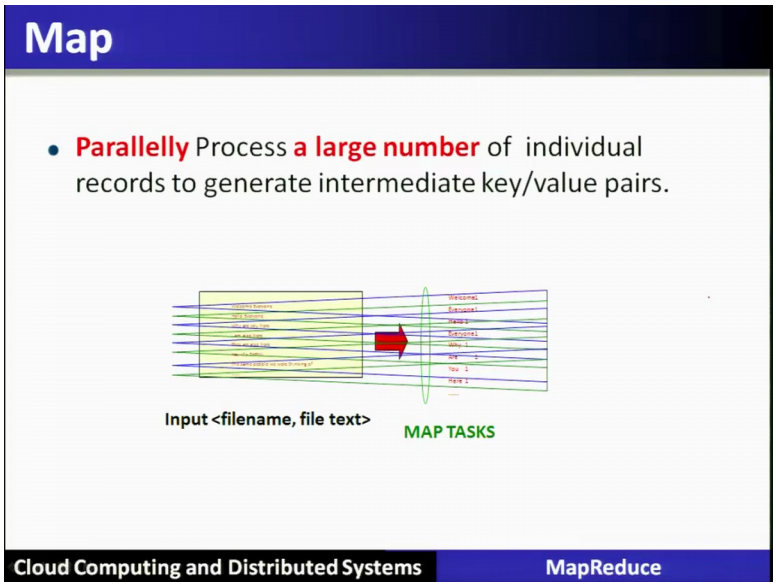
(Refer Slide Time: 11:23)



Now, this particular map function will processes these individual records to generate the intermediate key value pairs that is what we have shown. Now, this particular task can be executed by two different map task or we can divide this file into two different, two different chunks or we can call them as a splits. So, there are two splits and we intend to execute in parallel. So, that this particular document can be processed big size document can be processed.

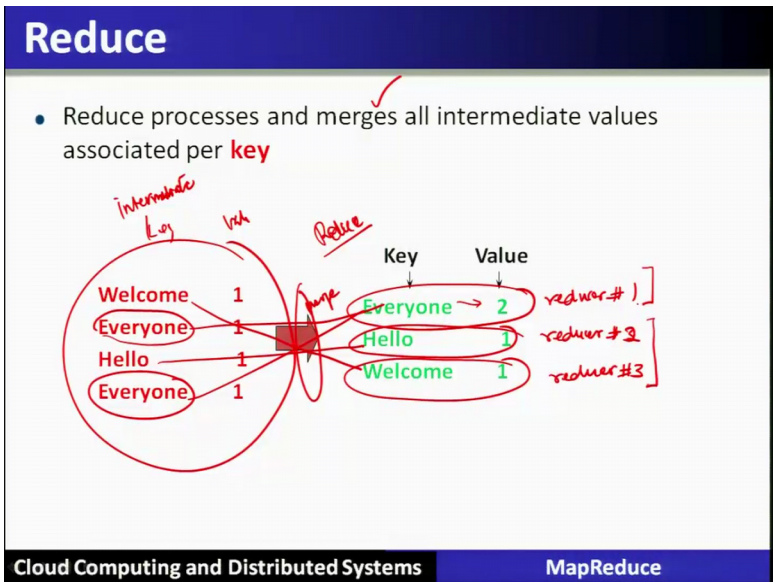
So, in that process these particular splits are given to different servers, they are called as a map task 1and the other one is called map task 2. So, this particular input is given to the map task 1 and this map will generate the intermediate key value pairs they are called intermediate key value pairs. So, also the map task 2 also will do the same thing. These intermediate key value pairs will now be given as the input to the next phase which is called reduced phase. So, for reduced phase the output of the map phase will become the input.

(Refer Slide Time: 13:08)



So, the parallelly processing a large number of individual records to generate the intermediate key value pair is basically, inherent in this paradigm that is called map using the map tasks.

(Refer Slide Time: 13:26)

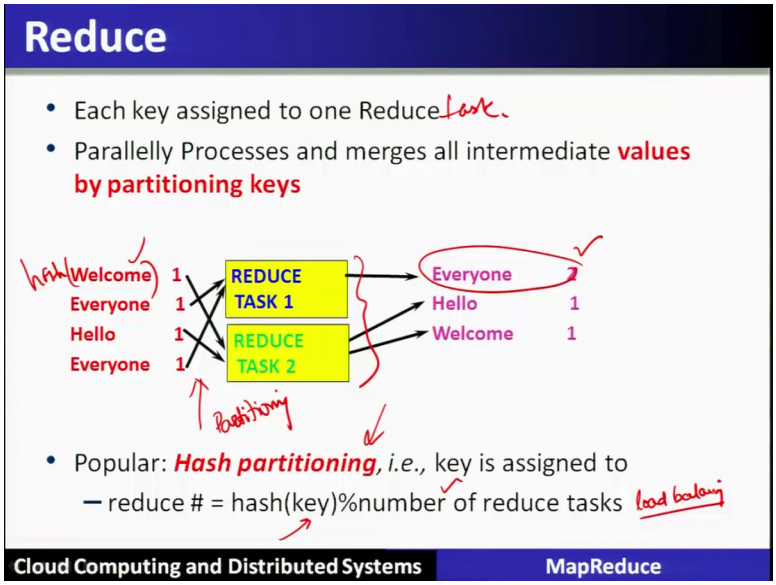


Then this particular output of map phase is given to the reducer. That is the key value pairs are given this is called intermediate key value pairs, key value pairs are given to the reduce function. So, reduce processes and merges all the intermediate values associated per key together in this phase. So, here you can see that everyone is appearing twice. So,

they should be merged in the reduce phase and therefore, it will give the final output associated with every key.

So, let us see when this happens using the reduce function what it does? It combines or it merges, these two keywords and it will generate the combined values. Similarly, hello is appearing only once and welcome is also appearing once. Now to merge these different keys which are having the same keys they have to be merged together in the reducer phase. So, they have to give be given to the same reducer. And this hello keyword may be given to the separate reducer. Now there are requirements of three different reducers, but if you have only two machines available, which can be assigned to these reducer. So, a particular machine can handle two different or more than one different reducers also.

(Refer Slide Time: 15:40)



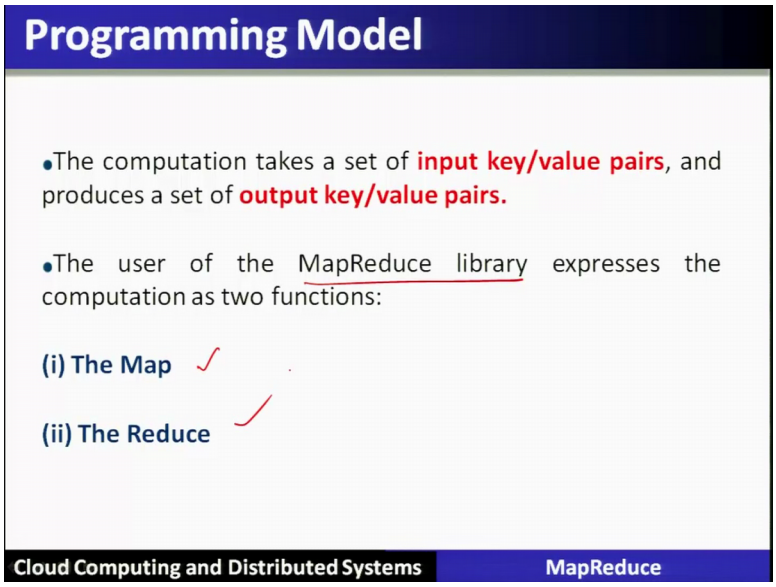
So, let us see the details of this reduce function. So, each key will be assigned to one reduce to one reducer or to one reduce task. So, parallelly they will be processed and it will merge all the intermediate values associated of the same key to one reducer by partitioning this particular keys. So, here in this case you can see the partitioning or these keys are basically able to map to reducers which you have here shown in this example.

So, for example, map 1 will go to reduce task 2, everyone will go to reduce task 1 hello will go to reduce task 2 and everyone will go to the reduce task 1. And reduced task 1 will then combine these 2 everyone into that combined function that is it will aggregate.

Now this kind of partitioning of the keys to the reduce task can be done using hash partitioning.

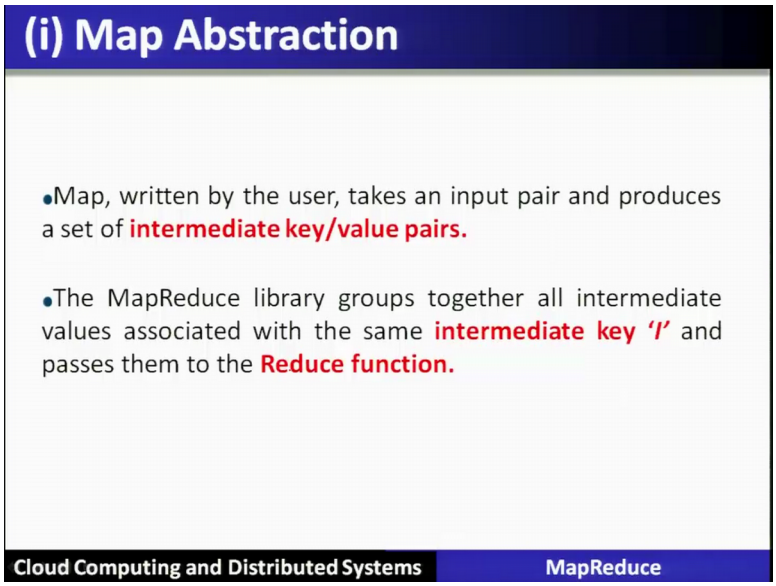
So, using hash partitioning we can hash a particular key. And this particular key when it is hashed then it will be taken the modulo of the number of reduce tasks. In this manner this particular keys will be partitioned equally across all the reduce task. So, there will be a proper load balancing of these keywords which are assigned to this reduce function.

(Refer Slide Time: 17:40)



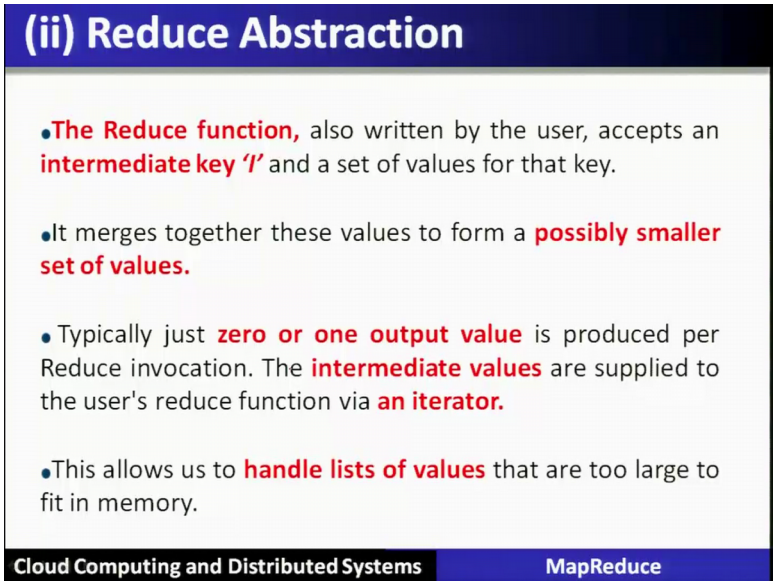
So, let us see the programming model in more detail of this MapReduce. So, the computation takes a set of input key and values pairs and produces a set of output key value pairs. So, the users they have to use the MapReduce library to express their computations using these two functions.

(Refer Slide Time: 18:06)



The map abstraction the map written by the user takes the input pairs. Pair and produces a set of intermediate key value pairs. MapReduce library a groups together all intermediate key values associated with that intermediate key and passes them to the reduce function.

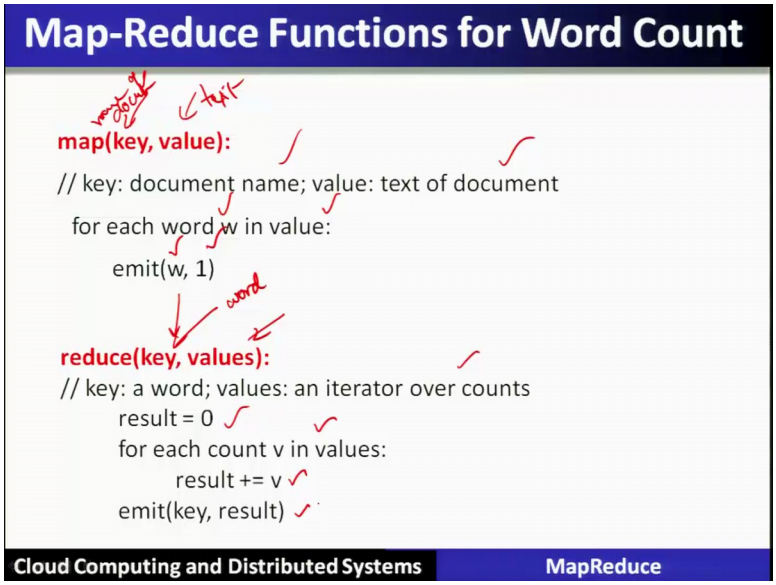
(Refer Slide Time: 18:23)



Reduce abstraction reduce function is also written by the user accepts the intermediate keys, which are generated from the previous phase that is from the map phase. And the set of values for that it merges together these values to form a possibly a smaller set of

values. Typically zero or one output value is produced per reduce invocation. The intermediate values are supported to the users reduce functions where it later. This allows us to handle the list of values that are too large to be fit in the memory.

(Refer Slide Time: 19:00)

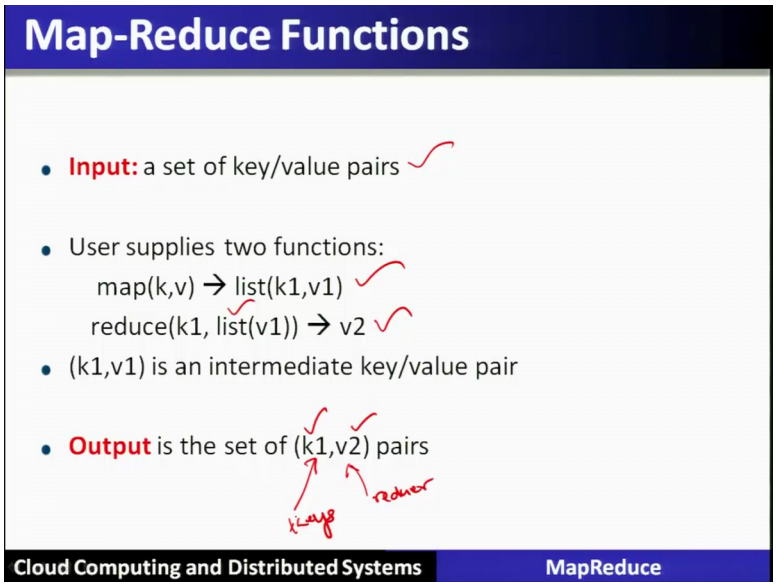


Let us see through an example of a word count. Here the map function will take the key values and key is nothing, but the name of the document and value is the text of that particular document. So, when given this particular document the name of a document and this is the text of a document or the entire document is given as a value then for each word which are separated by either the blanks are other separator. So, each word will be identified in the document or in the value of that is called the document. And it will emit that word w with a value 1 in the word count program.

Similarly, this output will be now taken up by the reducer, now here the key is a word and the values are the values which are being generated over here. So, values and iterator will count these values. So, let us begin this process with initialization of result is equal to 0. So, for each count v in these particular values, the result will be incremented by that value and it will emit for key this corresponding result and that will be output from this MapReduce function.

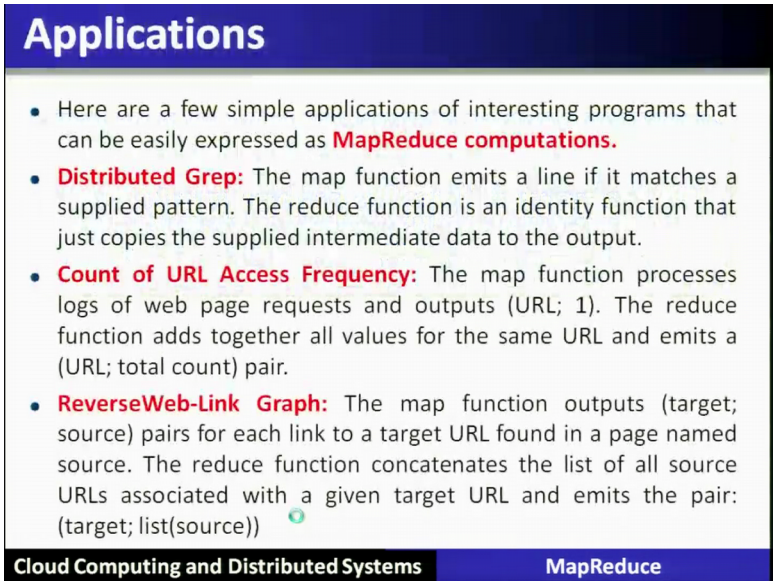
.

(Refer Slide Time: 20:35)



So, in general input will be a set of key value pairs, that we have already seen and the user has to specify it is own application through the map function; which will emit a list of intermediate key value pairs are now given as the input to the reduce function which will generate the output that is the values of all keys. So, output will be that keys and the final values which will be generated by the reduce function. And so, let us see some of the applications.

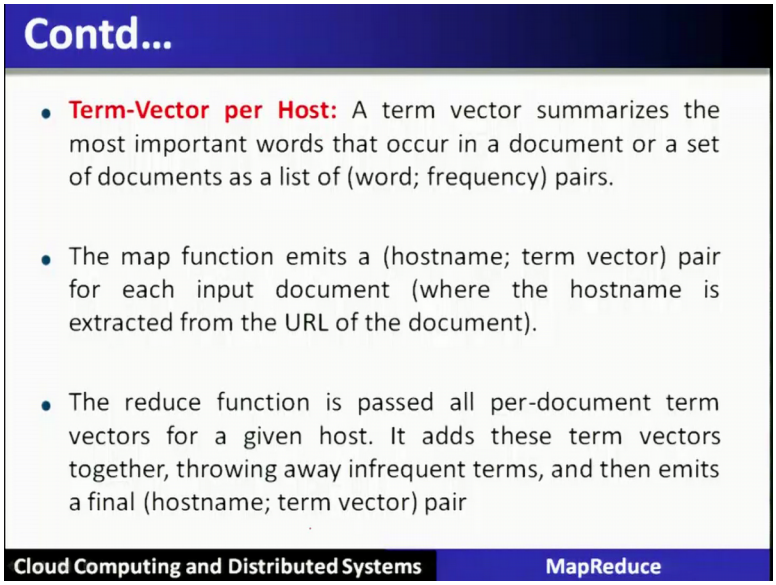
(Refer Slide Time: 21:24)



The applications of this MapReduce paradigms are many some of them we will here list them to understand the details of these applications. So, distributed graph application; that means, distributed graph can be applied on the log files or an error files wherein we can check or we can find out that pattern and find out the what kind of error it is incurring. So, since the log files are too big therefore, this distributed graph is an ideal application. Similarly, the count of URL access frequency is also one of the important application. And the list of URL's are being generated by any proxy server or in the cloud who is running the websites.

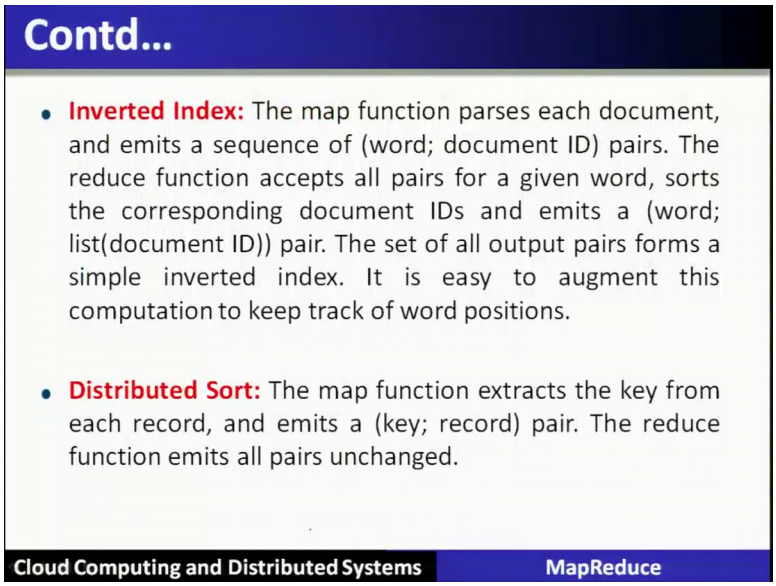
So, how many such accesses are being made across different websites accesses by the user. So, they are total count also becomes a MapReduce application. Similarly a reverse web link graph that means, a map that means, we want to find out the graph of those websites and who are accessing those websites. So, that is called link graph. So, we want to find out the reverse web link graph means to that website how many people they are accessing it. So, if we want to know that that is that application is called reverse web link graph.

(Refer Slide Time: 23:06)



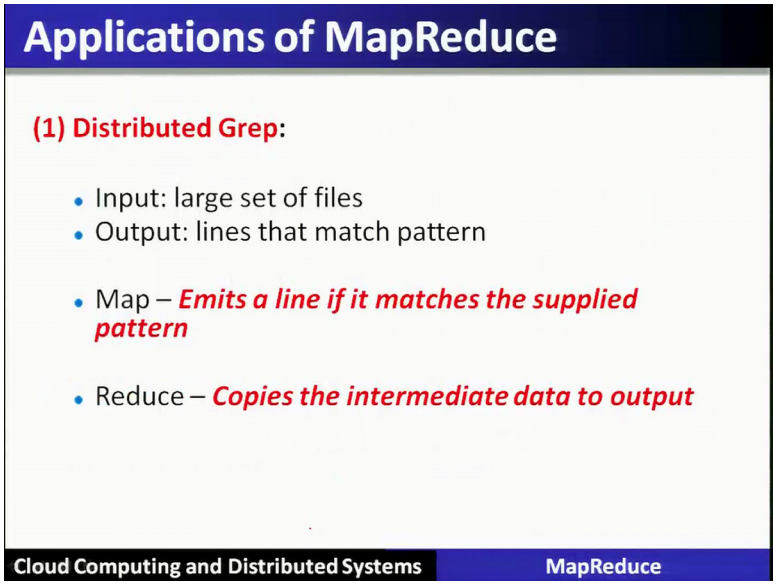
Similarly, term vector per host, a term vector summarizes the important words that occur in a document or a set of documents as the list of words and their frequency which is called a term vector per host.

(Refer Slide Time: 23:27)



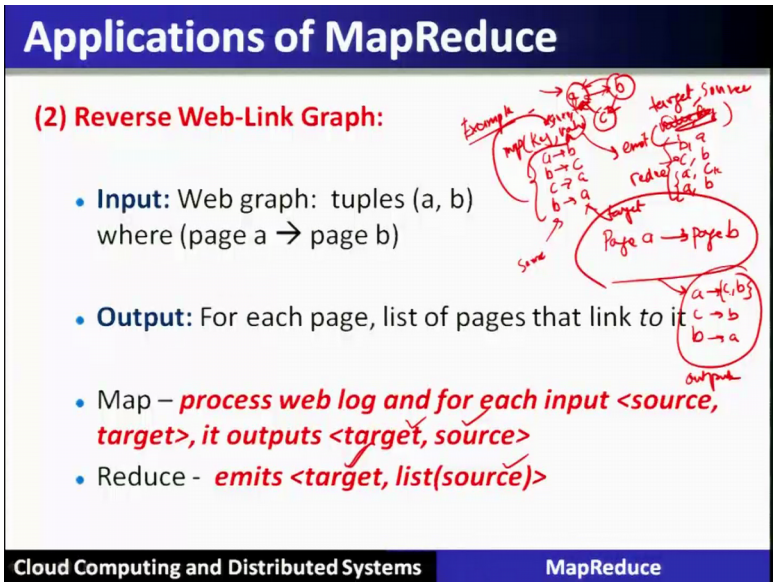
So, this particular application can also be programmed using MapReduce inverted index that is the map function parses each document and emit a sequence of words. So, the reduce function accepts all pair of the given words and sorts the corresponding document ID and emits the world. So, the set of all output pair forms a inverted index and it is easy to augment this computation to keep track of what position. Similarly, distributed sort that is the sorting algorithm in a large scale is called distributed sort.

(Refer Slide Time: 23:59)



So, let us see some of the use of MapReduce in these applications. So, distributed graph here the input is a large set of files and the output is the lines into the file that matches that particular given pattern. So, map will emit a line if that pattern is matches the supplied pattern and reduce will copy the intermediate data to the output.

(Refer Slide Time: 24:28)



So, this can be written using a simple program like word count. Another application is called as reverse web link graph here the input is the web graph in the form of a tuples a, b where the page a will basically accessing or having a link to the page b. Now, given this particular input the output for each page is the list of pages that link to it. To do this, the map function what it will do? It will process this particular web log for each input and it outputs the reverse of that that is the target and the source.

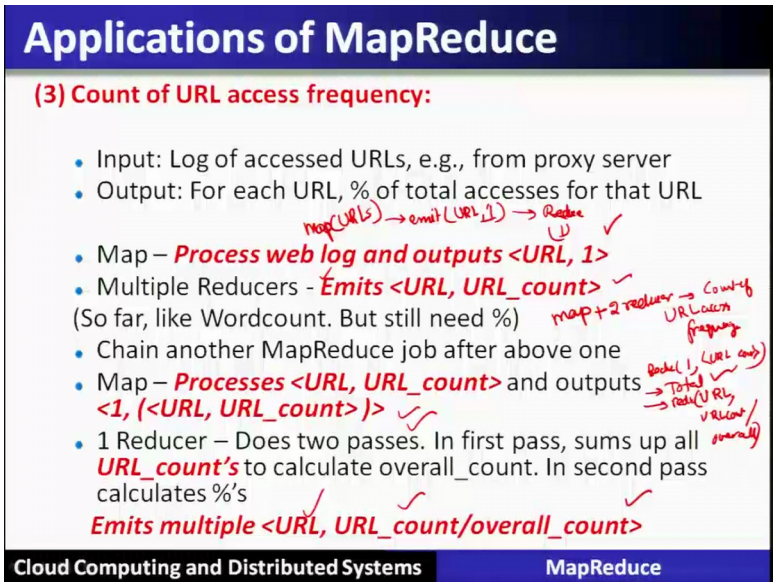
Then reducer will collect the target and form the list of source. Let us take an example; let us see a graph, a b and c. So, here in this particular graph, there are three nodes a b c or let us say there are three pages a b c, a is now referring to b or accessing to b and b is accessing to a and a is accessing to c. So, the list of pages here it will be in the form of a b then, then b c, then c a and then b a. Now the task is to find out for example, in a how many links are being accessed? So, here it is showing there are two links. Let us see how using MapReduce we can do this.

So, this will be the input this value will be the input, this is the key and this is the value this will be input to the map function. So, the map function will be given this input. Now

the map will emit value, comma key so, that means it will emit this particular output. Let us say here it is given as source and target, this is called source and this is called target. So, this will be given as the values to this particular thing. Now this will emit the target, comma source. So, MapReduce output will emit b, comma a it will output c, comma b it will output a, comma c it will output a, comma b. Then this particular output will be now fed to the reducer, reducer will now generate based on the key as the target it will find out the list of source.

So, this particular key for example, a will be combined and the list will be generated for a c comma b this will be list for c there will be list called b and for b the list will be a this will be generated out of the reducer. So, here we can see the node a is having two links pointed one by b and the other by c. So, using MapReduce program we can easily write down the map and reduce function which is very small program. But here we have to know the tricks to solve these particular problems. And when a big programmer or a big file is given it will be able to generate these values.

(Refer Slide Time: 29:01)



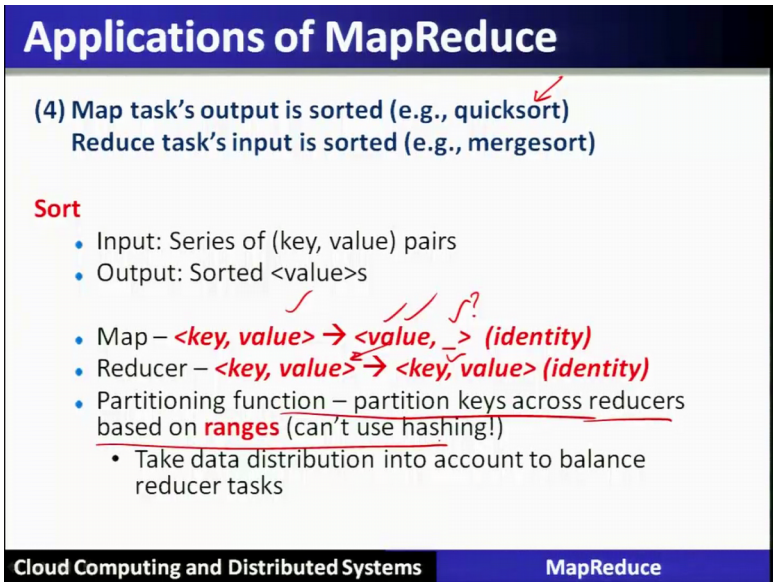
Now, let us see another MapReduce program to count URL access frequency. Now here the input is a log of accessed URL's from the proxy server and output will be the each URL and the percentage of total accesses for that URL so, this will be a bit difficult. Let us see the details how we can write down a program for this particular application of URL access frequency. So, here first what we will do is that map function will process

the web log and output URL with a value 1. So that means, all the URL's will be given to the map function, URL's will be give raise the input and what it will emit here is URL and 1, that means, for every URL it will emit 1.

Now, the next phase comprises of multiple reducers. So, the first phase what it will do? It will take it will take this particular output. Now reducer number 1, it will take this output URL 1 and then it will it will count the total URL's, that is it will generate a reducer 1, comma URL and the count. So, with 1 and all the output whatever is generated, so that means, it will generate the total number of URL's present. So, it will generate the total values of URL present.

Now, then another reducer will be basically, done in the second pass which will count the percentage of the URL in the form that this particular reducer will take the URL's, which are omitted by the map function. And for every URL count whatever is output by the reducer in this particular phase, it will divide by the overall count whatever is calculated in the previous pass of the reducer. Therefore, with one map function and 2 times application of a reducer function, we will be able to calculate the count of URL access frequency. So, this particular application has shown that it is possible to cascade multiple MapReduce job to solve an application.

(Refer Slide Time: 32:40)



Now let us see another application, here in this application it will do the sorting. Now the sorting using MapReduce; that means, if a very big file is given: how are you going to

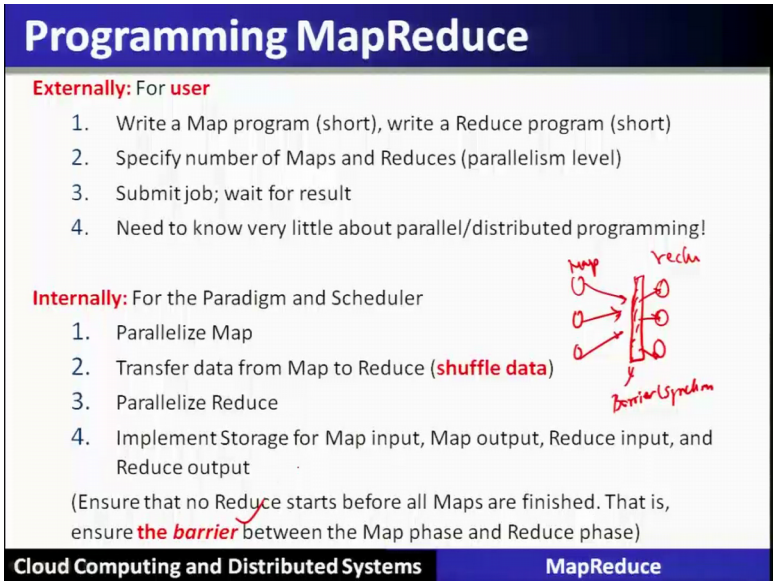
sort using MapReduce is the task. Now here the map tasks outputs they result that is intermediate results in a sorted form, that is using the quick sort already is implemented inside the MapReduce, inside the map function.

Similarly, the reduce task also input in the form of sorted manner that is also done using some sorting algorithm for example, using merge sort. So, using this information let us see how the sorting can be done using MapReduce in a very simple manner. So, let us see that if the input is given to the map. So, it will generate that particular value which is output from the map function. And the second one that is the value of will be any value that is not that important.

So, you know that these particular values are already sorted so, the map will generate these particular keys in a sorted order. Then as far as the reducer is concerned this as a input of this map function and it will generate the same value again. The only thing we have to do is that in the partitioning, we have to provide the range instead of hash function because, hashing will randomize them. Since the output of the map function is already in the form of sorted and we do not want to disturb it so, that we have to provide the range values the range of keys. So, range of keys means the reduce task will be assigned according to the ranges.

So, that this particular range will not be disturbed and the output of this map function will be preserved which is in the sorted list and the output will be sorted. So, this particular example or application of MapReduce could make this sorting algorithm easy why because, it has exploited the way map function is giving the output it is in the sorted order. So, we have to just preserve that sorted order without being disturbed by the partitioning function so, that the entire file will be in a sorted order output.

(Refer Slide Time: 36:06)



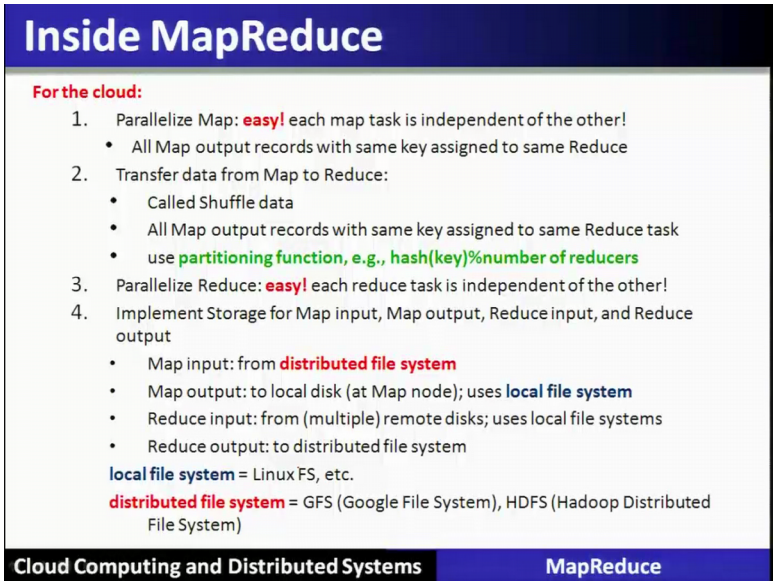
Now, let us see the MapReduce scheduling how it is done? Now, as far as programming of a MapReduce is concerned externally for the user we has to only write down the map program, this is very short programs. And they have to specify the map and reduce function and submit the job and wait for the results. The users need not have to know the details of parallel and distributed concepts and the programming.

As far as everything is done internally, so, for internally for the paradigm and the scheduler has to build everything and give the output. So, it has to parallelize the map function execution; that means, it will transfer the data from map to the reduce, that is done by the shuffling that we have seen earlier that a partitioning and shuffling using hash function was being carried out.

Then parallelize the reduce function and implement the storage for map input map output reduce input and reduce output. One thing is there that we have to ensure that, no reducer start before all the maps are finished, that is being done inside MapReduce. So, this ensures that a barrier between the map phase and the reduce phase is there. So, barrier in the sense this is called a barrier is nothing, but a synchronization. So, when all the map function will finish, some are finishing early some are finishing quite late. So, earlier who ever finish they have to wait till all the map function completes their execution then only the next phase called reducer phase will begin afterwards so, this is called a barrier.

In some of the implementations this barriers can also be in some cases being a synchronized; that means, they are not synchronizing may be in the next phase without completing all the reduce phase the next phase can begun.

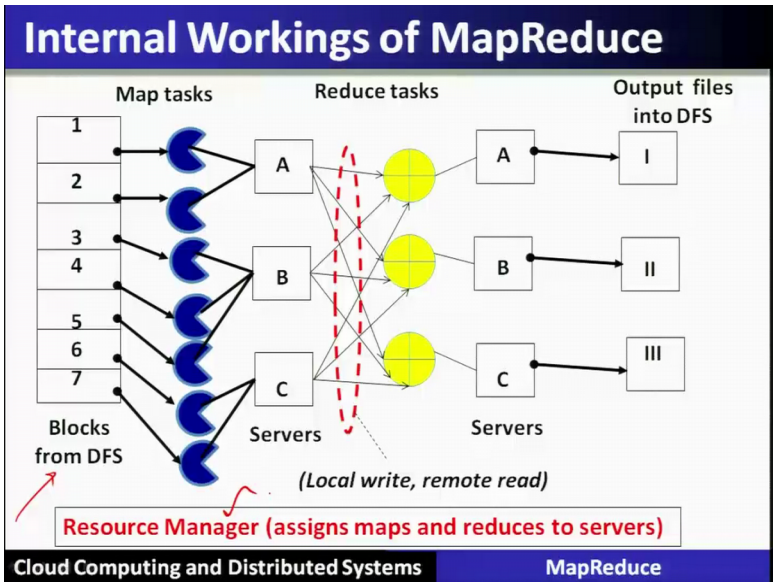
(Refer Slide Time: 38:34)



So, inside the MapReduce we have seen that parallelize map is quite easy now why because, each task in the map they are all independent. So, they can parallelly execute inside a cloud. To transfer the data from map to reduce a shuffle function is to be called and all map output with the same key is assigned to the same reduce job and this requires to use the partitioning function which is hash based portioning. So, that it will be uniformly distribute across all the servers which are running the reduce task.

Third step is to parallelize reduce is quiet easy why because, reduce is independent of each other, implement the storage for the map input, map output, reduce input,
reduce output. So, here we can see that the map input is done from the distributed file system. Map output also uses the local file system and reduce input also from multiple remote disks using the local file system and reduce output node is done through the distributed file system. Distributed file systems which are used here in the MapReduce are either the Google File System or Hadoop Distribute File System.

(Refer Slide Time: 40:11)

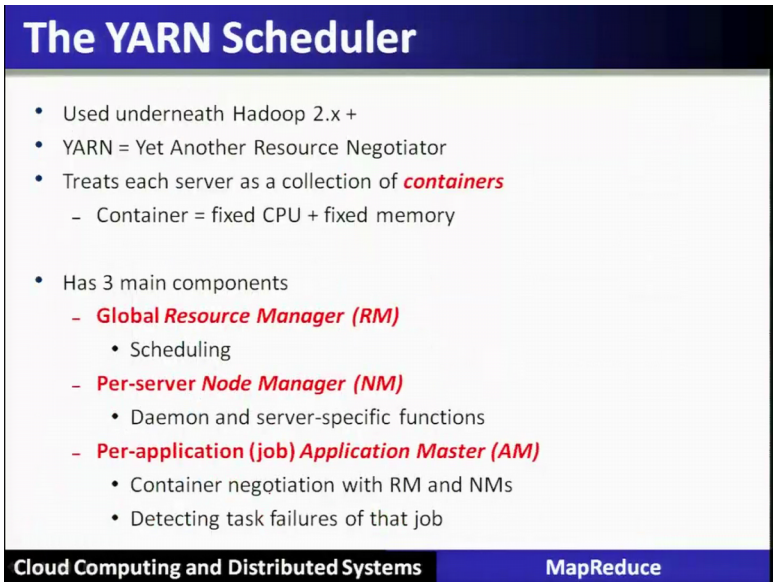


Let us see the internal working of the MapReduce. So, first thing is whenever the input file is given. So, this particular input file is splits it is divided into different splits seven different splits are shown over here. Each and ever is split is given to different map tasks now these map tasks are to be assigned a servers or the machines. So, for example, here it is shown that two different map tasks are assigned to on server. A three different map tasks are assigned to the server B and two map tasks are assigned to the server C. So, it depends upon how many servers are there these map task will be assigned.

Similarly, the output of these servers will be shuffled; that means, they will be grouped by the keys and will be given to the reducer functions. So, here this example shows that there are three different reduce task is being are defined over here. And these reduce task will be assigned to different servers. So, there are three different servers are assigned to the reduce task which will be executing these reducers. At the output of these particular servers are stored in a file in a distributed file system, which are given different names or there maybe there could be only file will also it depends upon.

Now, who allocates who assigns these map tasks to the servers? And the reduce task also to the different servers may be the same servers. So, it is the resource manager, it is the resource manager of a distributed file system or a yarn server that we will see scheduler which will basically internally does this.

(Refer Slide Time: 48:24)



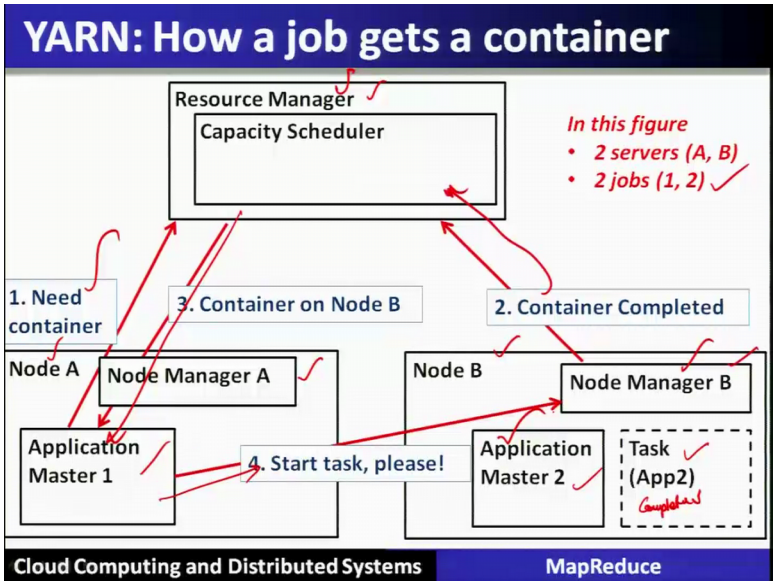
Let us see the yarn scheduler which is combined with the Hadoop 2.0 versions onwards. So, yarn full form is yet another resource negotiator. So, this particular yarn will treat each server as a collection of container. So, it means that, if the server is having some number of course, let us say 3 and it is having 3 GB of internal memory. So that means, one core and 1 GB will form one container so, it has now three containers. So, three different task can be assigned in this manner. So, this yarn scheduler has 3 different components, the use of one component we have seen in the previous slide that is called resource manager. Which assigns these containers to the different tasks that is to the map task and to the reduce task so that is called scheduling.

So, this there is a global resource manager which performs the scheduling of this particular task map or reduce task to this particular machines or the servers or to the container here in this terminology. Then another component of yarn is called the node manager. So, for every server there is a daemon process called node manager which basically, will manage all the server related functions so, that means, a server if it is giving three different containers so the node manager will manage it.

Then per application there is a application manager, application master. So, for every application there is a application master and this particular application master will negotiate with the resource manager and the node manager for allocating their tasks to

the machines. Also the application master is responsible to deal with the task failures at the time of execution of the job.

(Refer Slide Time: 44:45)



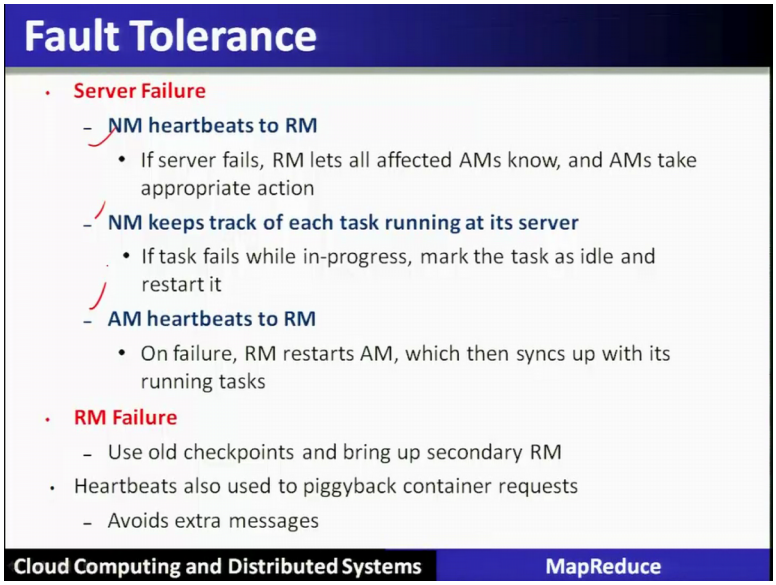
Let us see how the yarn will get a job of a container? So, here there is a resource manager and this is the example which is shown that there are two nodes node A and node B. So, the node A is having it is manager node A manager and node manager B will manage this server node B. So, there is resource manager which will manage all the nodes which are available in this scenario that is through the yarn.

Now, for there are two applications 1 jobs job one and job 2. So, there are two different application master they will manage these two tasks. Now let us see the example of these interactions with the resource manager. Now let us assume that the application master to it is task that is the application 2 task is completed. Therefore, this will return back through the node manager that it is free.

Similarly the application one master negotiates with the resource manager asking for a container, now since this container is available why because node B has completed it is task. So, that particular container is now free so it will be assigned back to the master 1 and it will start the task execution.

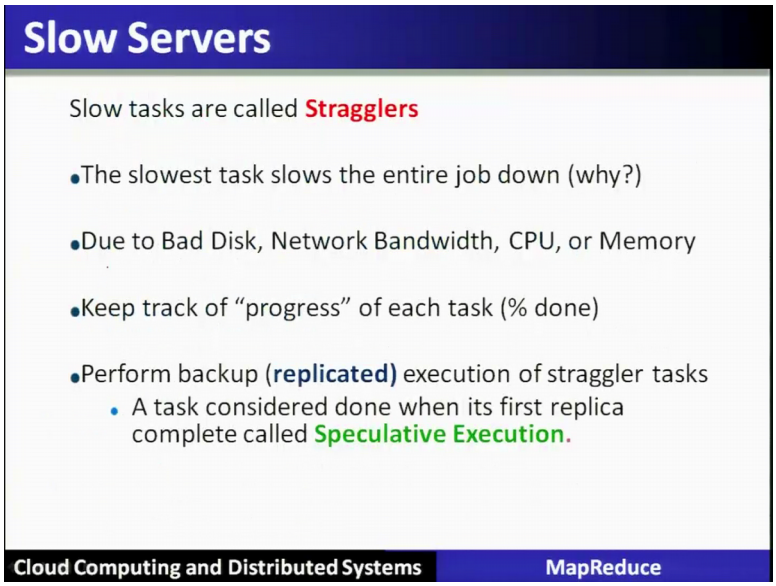
So, this particular interaction of different resource manager, the node manager and the application master, they will ensure that the resources which are required to execute the applications are done. Let us see the fault tolerance, which is handled in map reduce.

(Refer Slide Time: 46:52)



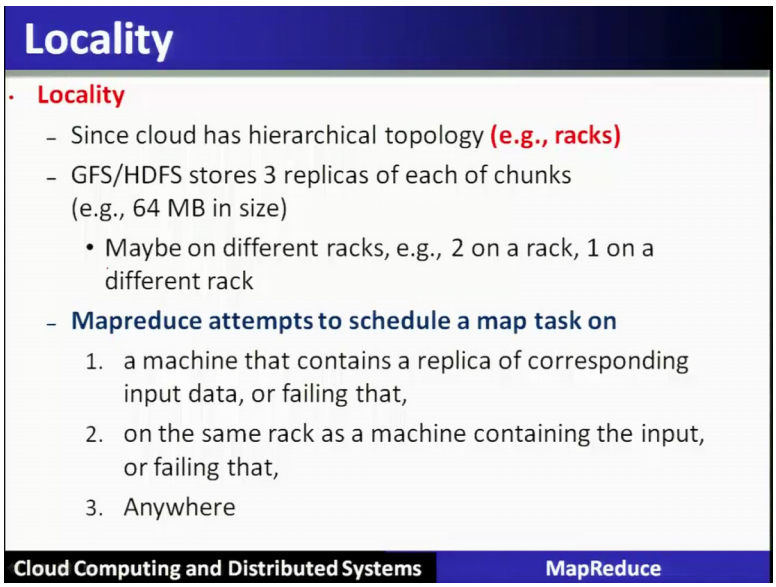
So, MapReduce handles different kind of faults at different level. For example, whenever there is a server failure so, this will affect the name node and resource manager, name manager, resource manager and application manager, will be affected in this particular scenario whenever there is a server failures. Similarly, whenever there is a resource manager failure then basically, it uses the secondary resource manager to failover and start working at.

(Refer Slide Time: 47:38)



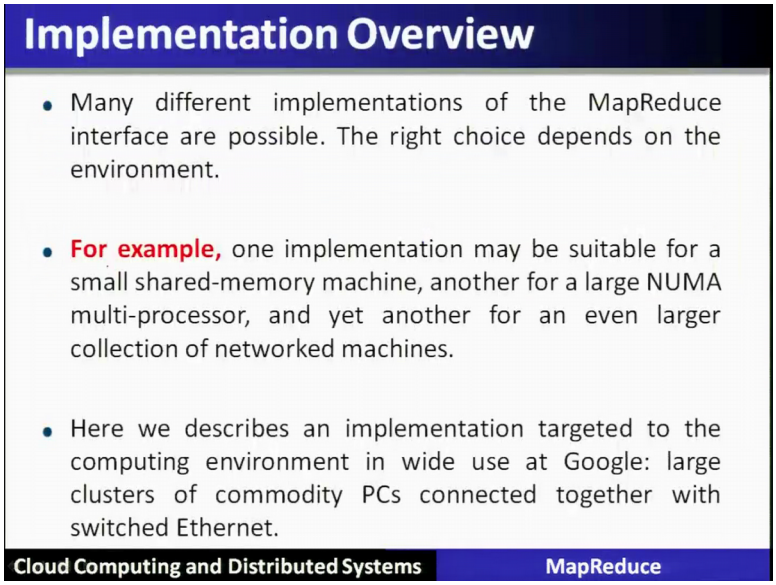
These are detected using the heart beats which are contained in the request and also avoid. There is a possibility of slow servers called stragglers. So, the slowest task will slow down the entire job that we have already seen because, it is using the barriers concept. So, slow servers are due to the bad disk, network bandwidth, CPU, memory, there can be many different issues. So, it has to keep track of the progress of the task and performs the backup that is replicated execution of staggered task which is called as a speculative execution.

(Refer Slide Time: 48:18)



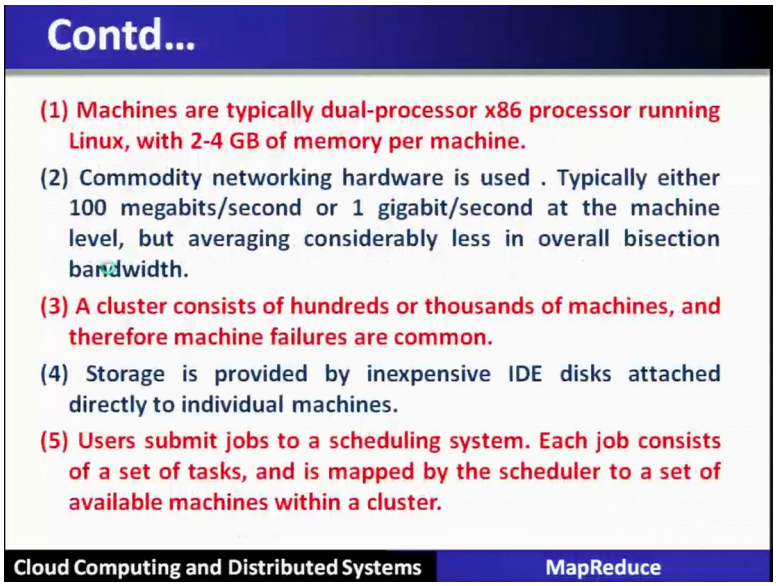
Locality; that means, that the cloud has the hierarchical topology of a data centre that is the servers racks and different racks are connected through the top of racks switch. So, the locality has to be exploited, that means, MapReduce will try to attempt to assign the server for a particular task which is very close to it so, that the network delay can be avoided here in this case.

(Refer Slide Time: 48:58)



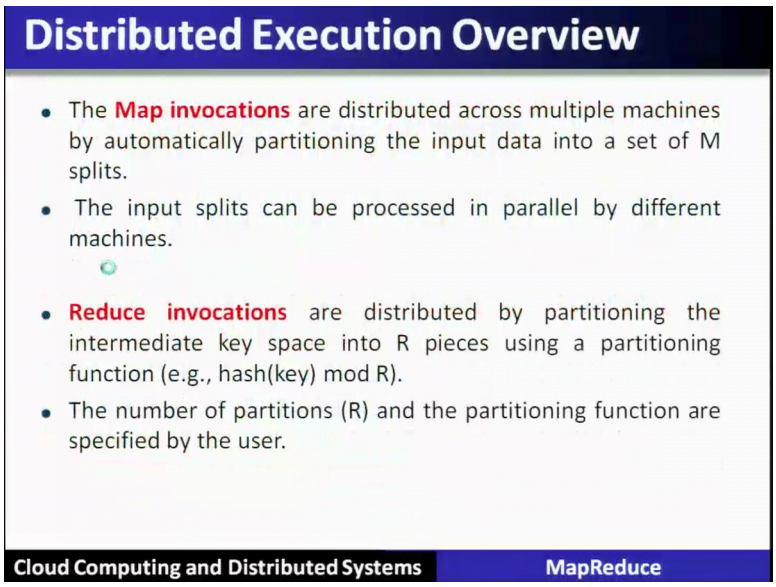
So, let us see some of the implementation of MapReduce, there are many implementation of MapReduce an interface an interfaces are available. So, let us see one such example.

(Refer Slide Time: 49:10)



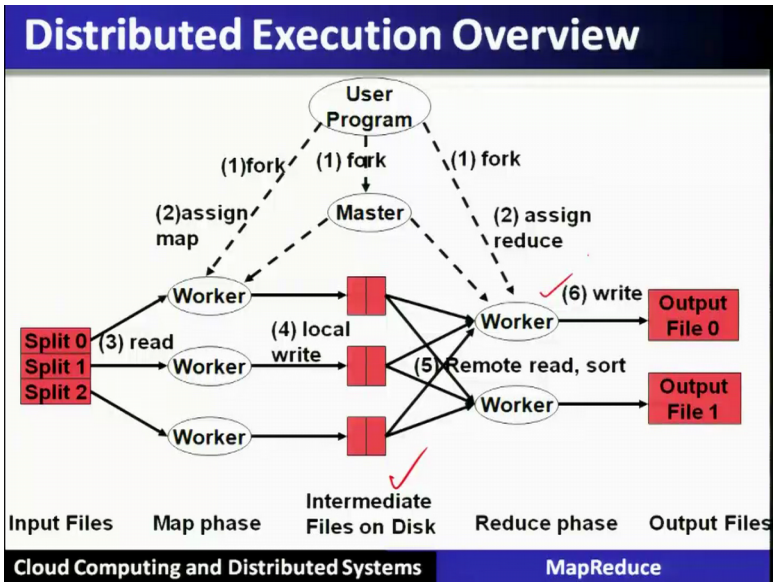
Here where we assume that the machines are dual processor commodity networks hardware is available and the clusters consist of hundred and thousands of machine is there.

(Refer Slide Time: 49:25)



And storage is also provided and user can submit the job to this kind of system. Let us see the distributed execution of this MapReduce invocation.

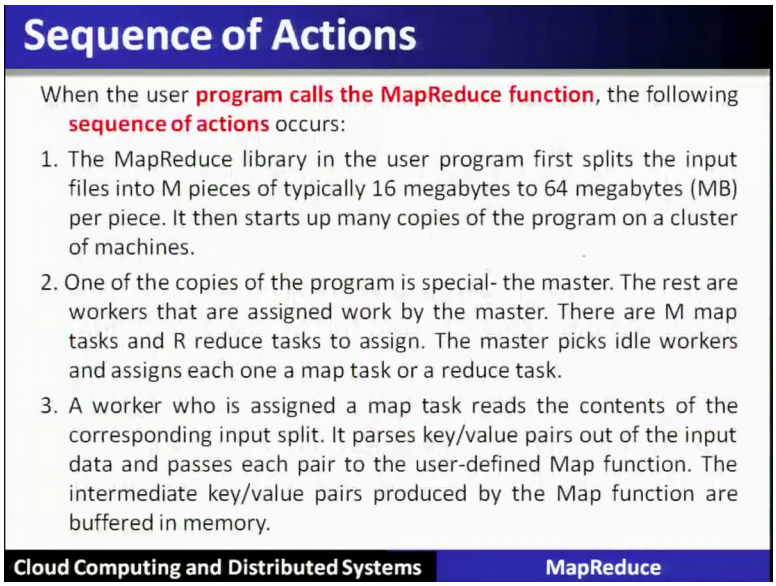
(Refer Slide Time: 49:32)



So, here we can see that the user program, user can submit their job in this particular manner and this particular user also gives the splits that is the input file is divided into different splits. And these particular user program, that is the map phase will generate different workers and these workers are assigned to the different machines. And when these splits argument to the worker they will output and they will write down on the intermediate files on the disks using some distributed file system.

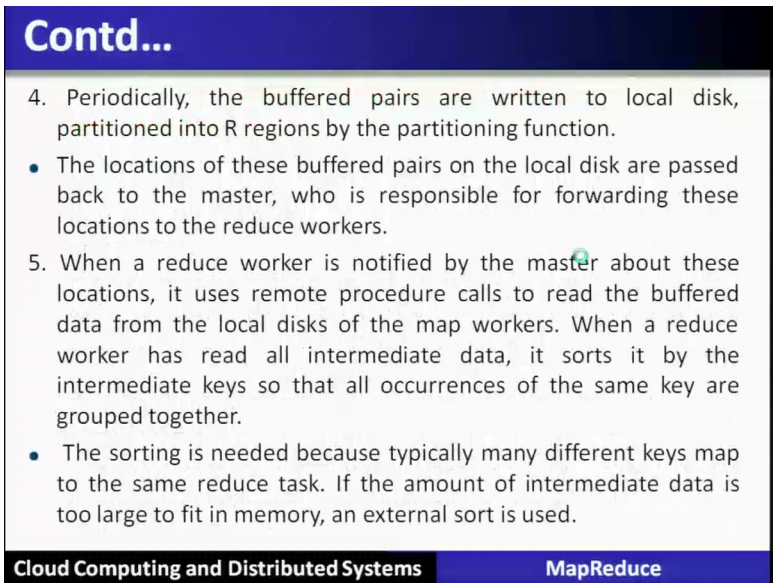
Now, the next task is that these particular intermediate output is given to the input to the next phase that is to the reducer phase. And this reducer phase that is called worker will be assigned by the user program that is to the master. Now, then these particular workers are assigned by the user program that is through the master. Now, then these particular workers are assigned the servers in the reduce phase and they will output through the to the output file in the distributed file system. So, this is the typical scenario of a distributed execution now here we can see the sequence of actions in this distributed execution.

(Refer Slide Time: 50:51)



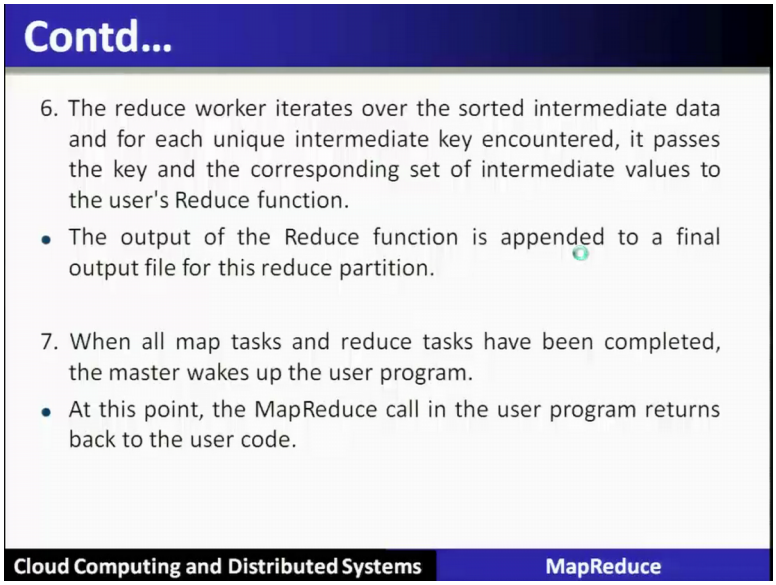
So, the program calls the MapReduce function and the sequence of actions, which we have told is being listed over here. For example, MapReduce library will split the user file into different pieces of 16 MB to 64 MB and then these one of these copies is called one of the program copies is called a master. And the rest are it will generate the different worker process and these workers are assigned the map task.

(Refer Slide Time: 51:31)



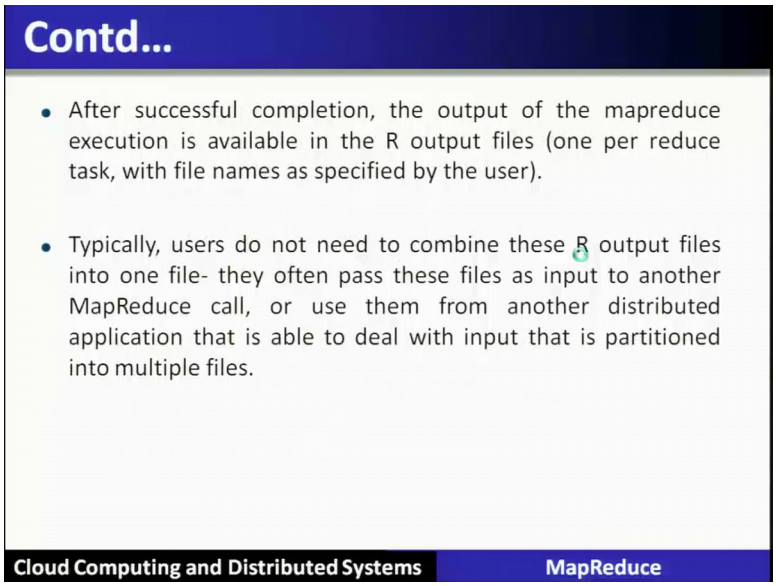
And into the end each split will be assigned to these workers and periodically they are being buffered and partitioned. These reduce and these particular intermediate results will be stored in the file system.

(Refer Slide Time: 51:51)

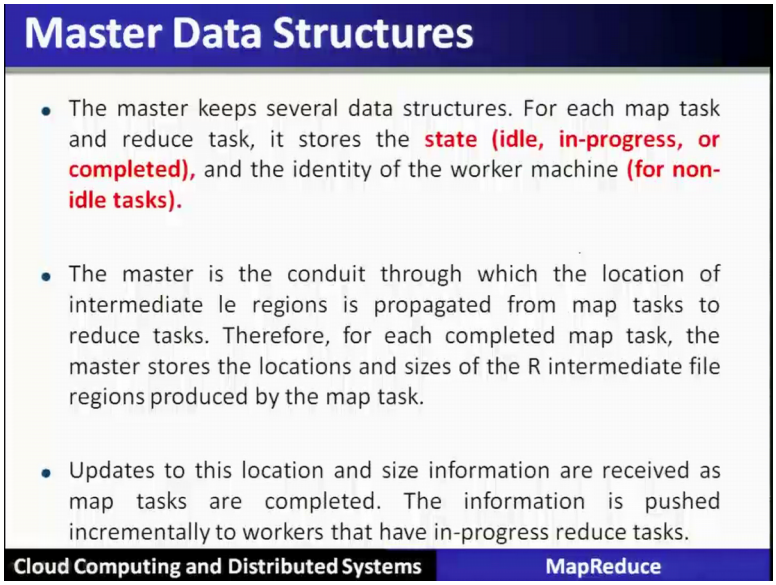


Now, the reducer worker is notified by the master and this particular output, intermediate output of the previous stage will be given as the input and it will be reducer phase will give the output.

(Refer Slide Time: 52:03)

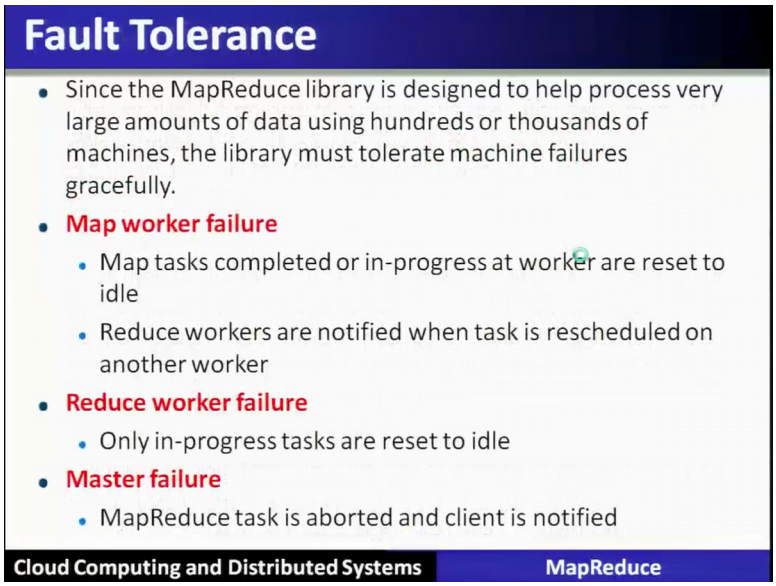


(Refer Slide Time: 52:04)



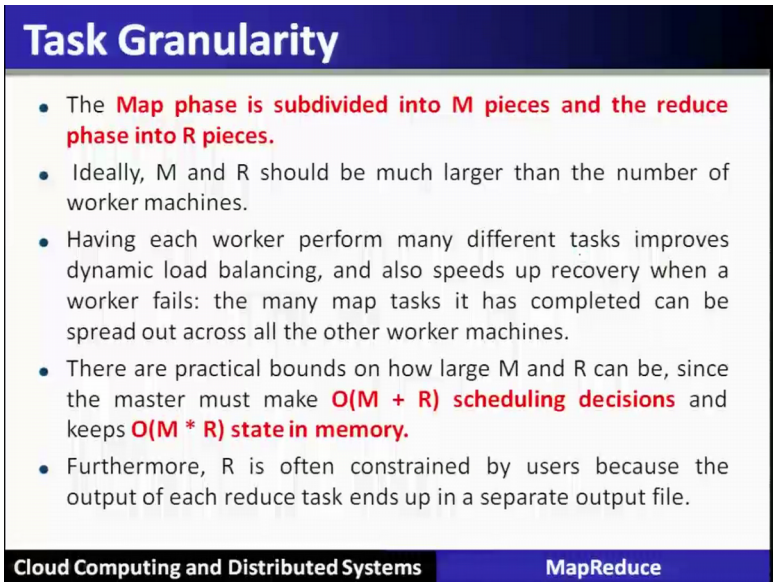
So here the master data structure is like this, that it basically has different states.

(Refer Slide Time: 52:18)



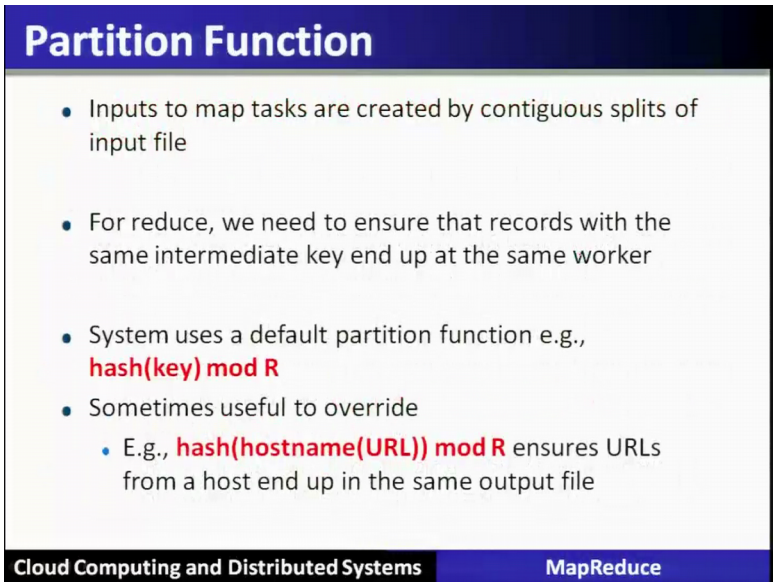
And it will trace through different states internally in the execution.

(Refer Slide Time: 52:27)



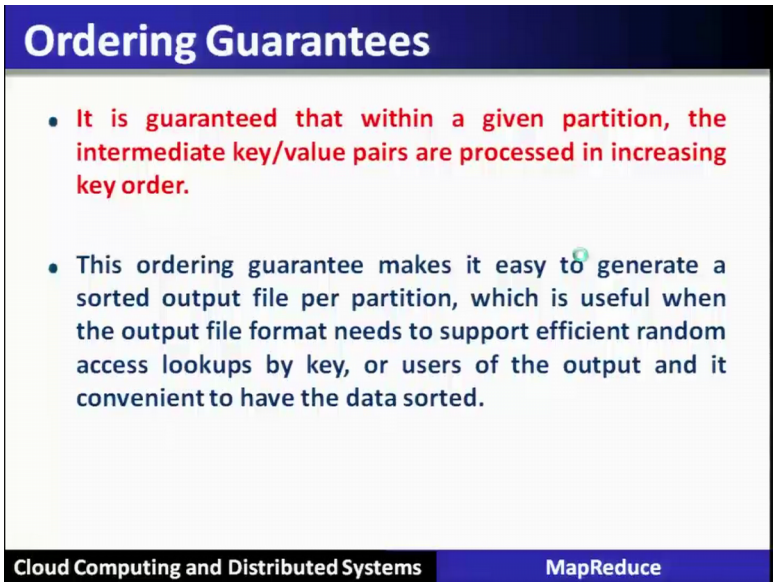
.

(Refer Slide Time: 52:29)

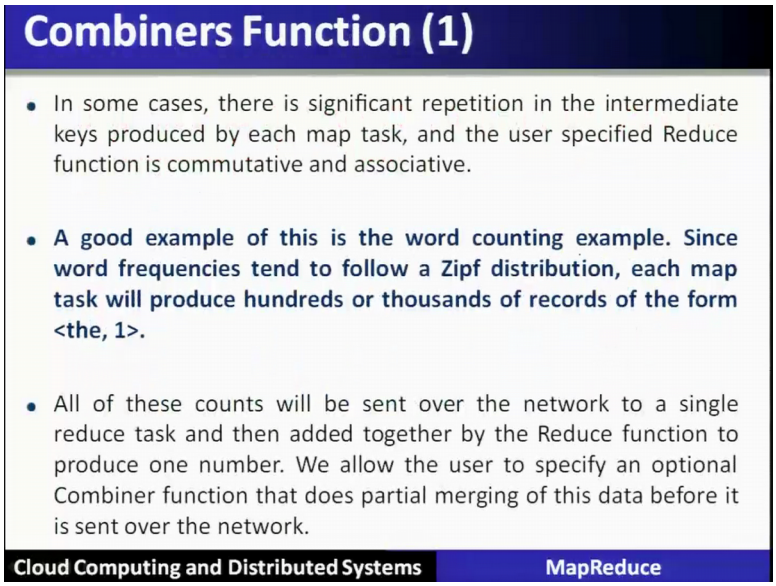


Fault tolerance we have already covered, locality is being exploited task granularity we have assumed partition function we have covered.

(Refer Slide Time: 52:31)

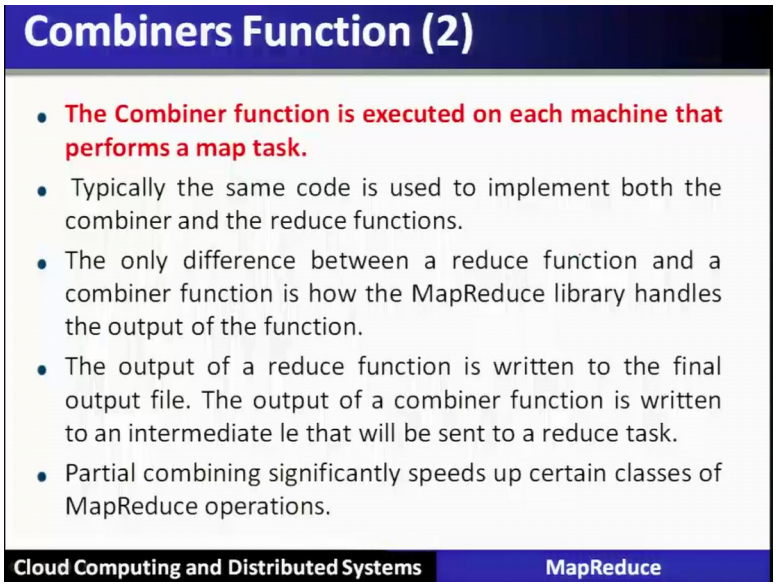


(Refer Slide Time: 52:38)

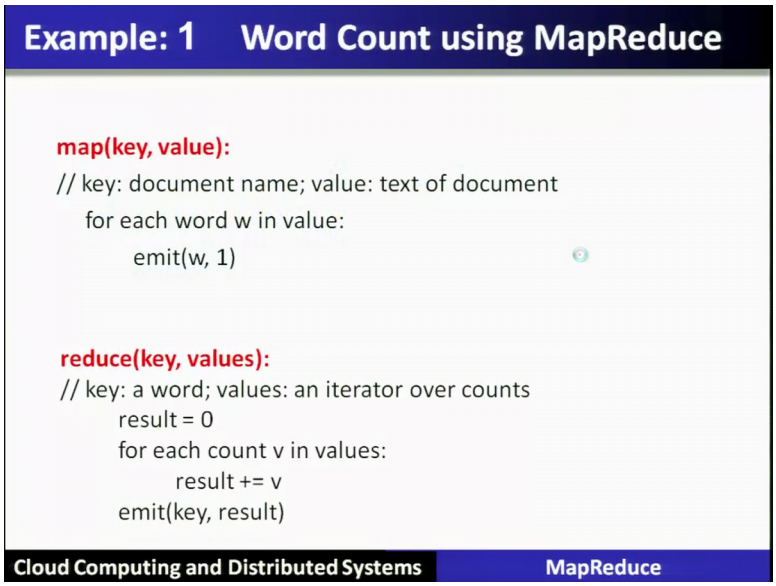


And ordering guarantees is also there that we have used we have shown in the sorting application and this is the combined function.

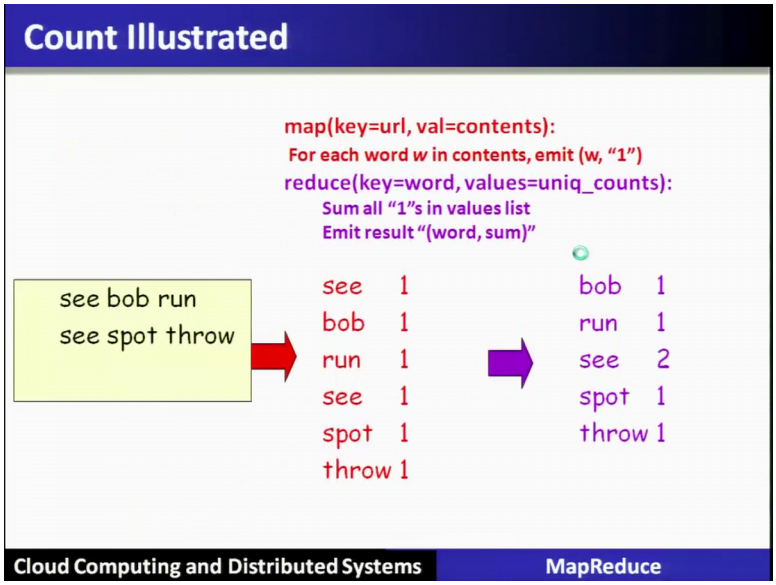
(Refer Slide Time: 52:44)



(Refer Slide Time: 52:47)

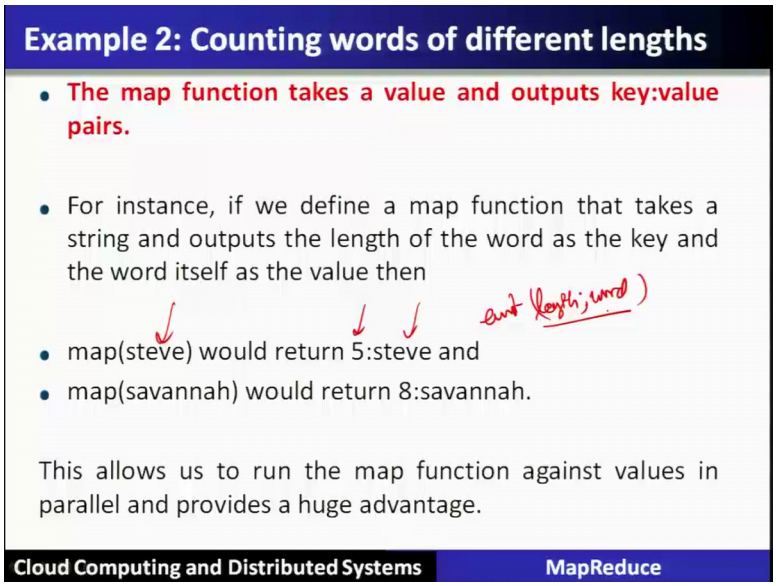


(Refer Slide Time: 52:51)



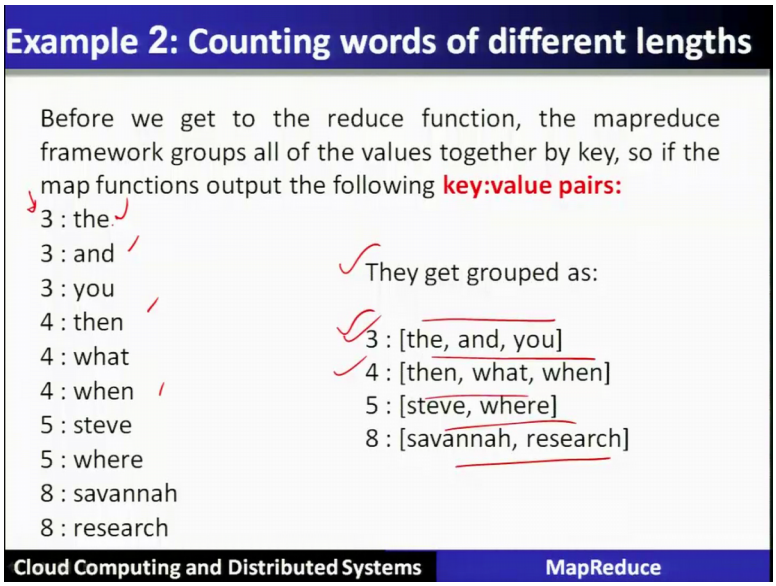
So, MapReduce more examples are available.

(Refer Slide Time: 52:52)



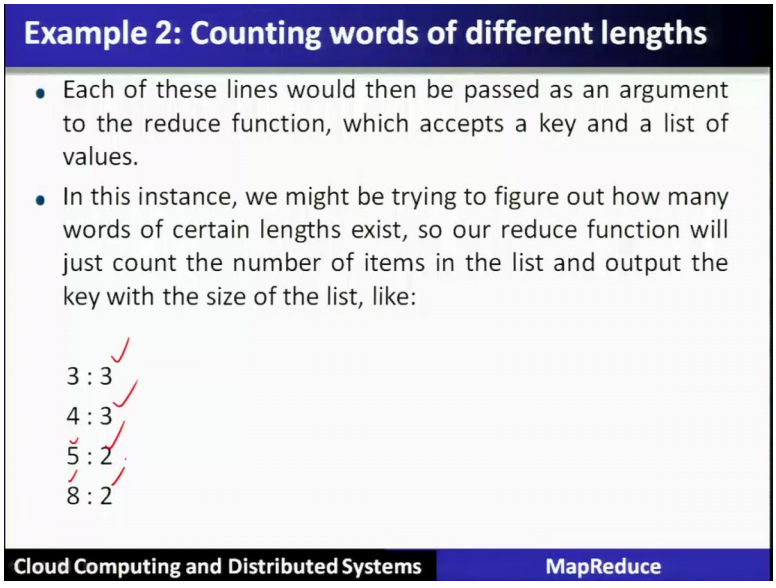
This word count we have already done. Now the counting words of different length, if we want to find out. Then what we will do is? Given a particular input that is then we will written its length and that particular word also. So, it we will written the length and the word both will be emit in the map function.

(Refer Slide Time: 53:27)



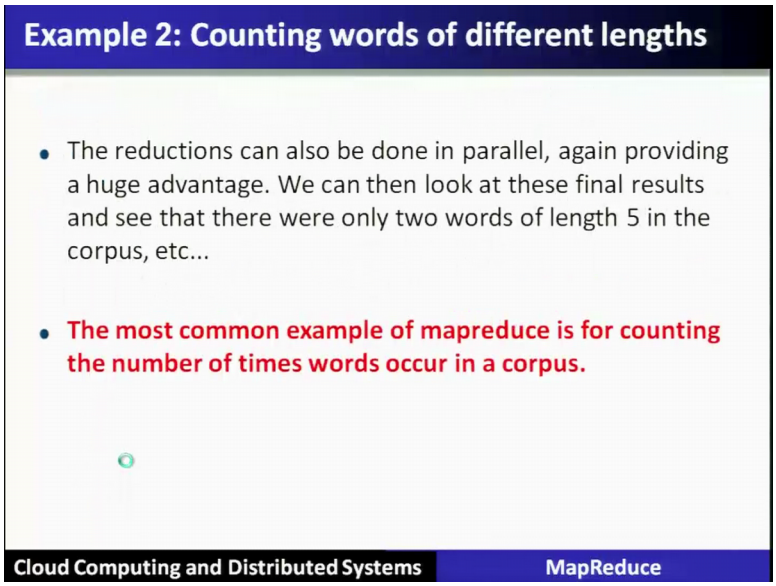
So, that is the case then for example, all these particular words, first their length and their words will be written. And then reduce function will combine them using that particular key that is the length of the words. And in this case for every length of word every length different keywords will be combined in the form of a list.

(Refer Slide Time: 53:58)

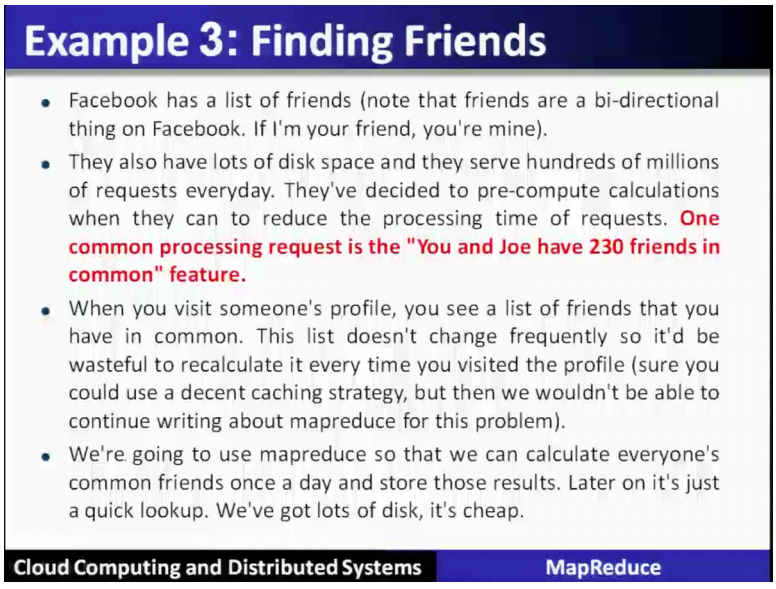


And this shows that the word that is having the length 3 there are 3 different words and length 4 there are 3 different words, length 5 there are 2 different and length 8 there are 2 different words. So, here we have counted the words of different length.

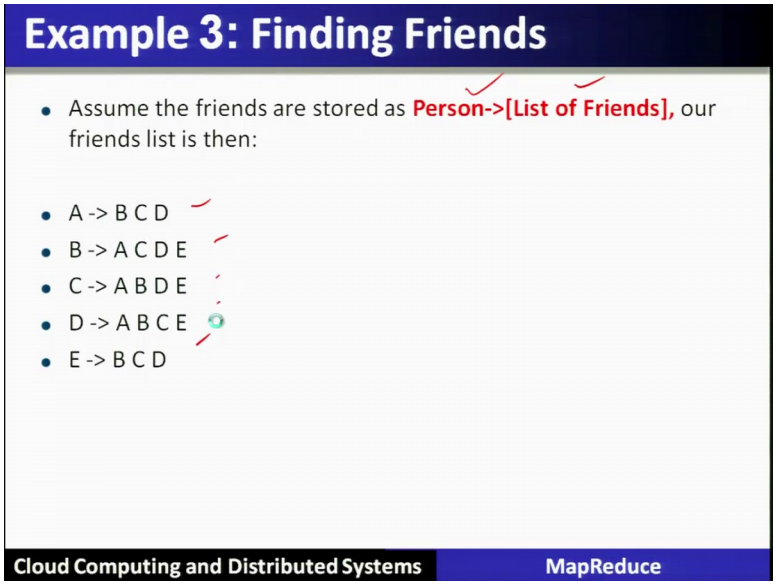
(Refer Slide Time: 54:10)



(Refer Slide Time: 54:14)

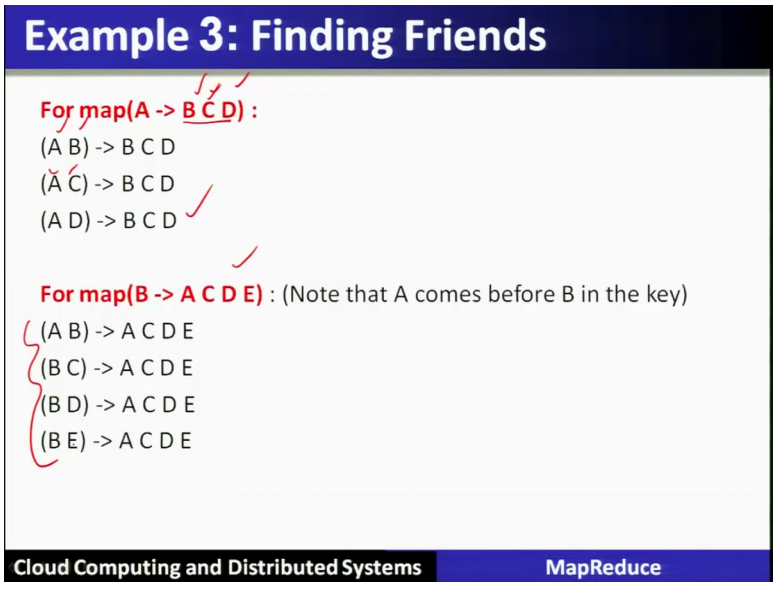


(Refer Slide Time: 54:16)



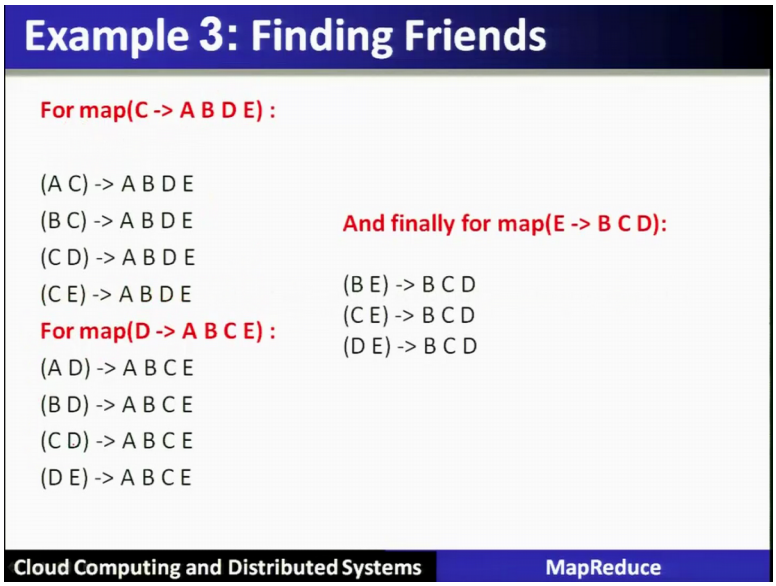
Another example is to find a common friends, list of common friends. So, here every input every person will provide a list of their friends. So, A B C D E they have provided their own list of friends, if this is the input given to the map function.

(Refer Slide Time: 54:29)



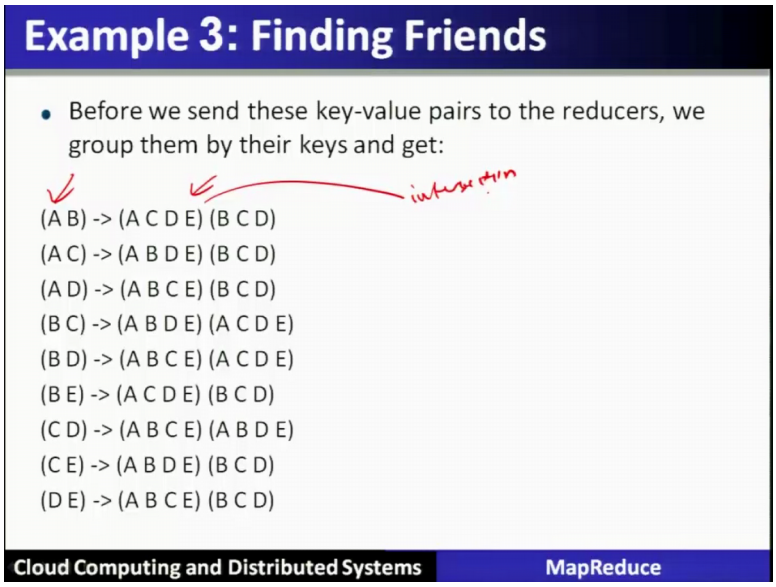
Map function will output, for every such list it will give a pair that is A B will list out this particular list of friends. And A C will list out friends and similarly A D. Similarly the person number B will give it is list and this also will be emitted in this particular format.

(Refer Slide Time: 54:56)



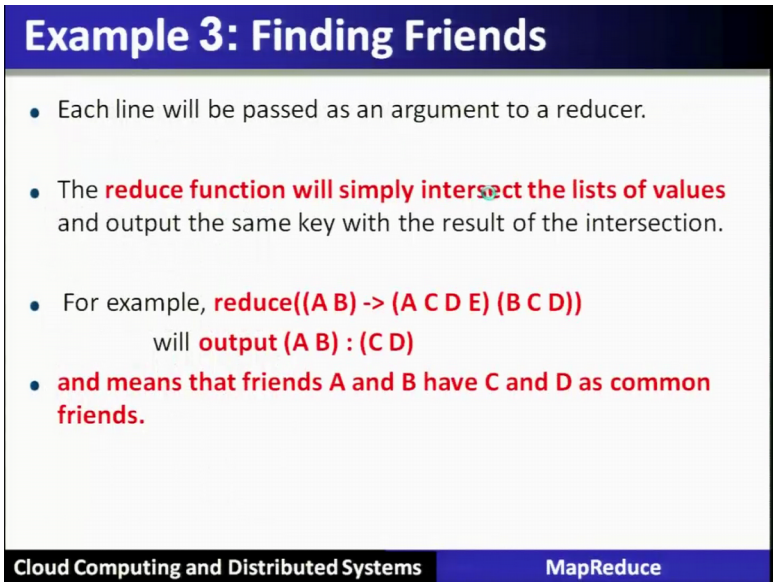
So, when all these list are output.

(Refer Slide Time: 55:00)



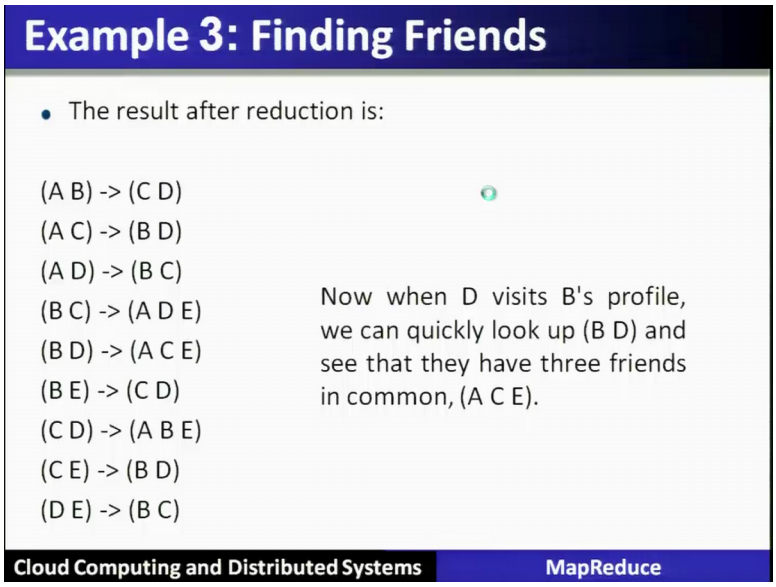
Then as far as the reducer is concerned, what reducer will do? Reducer will now for every keyword which is there. And the list which is basically available in the intermediate they will take the intersection. Intersection of these particular list of common friends.

(Refer Slide Time: 55:25)



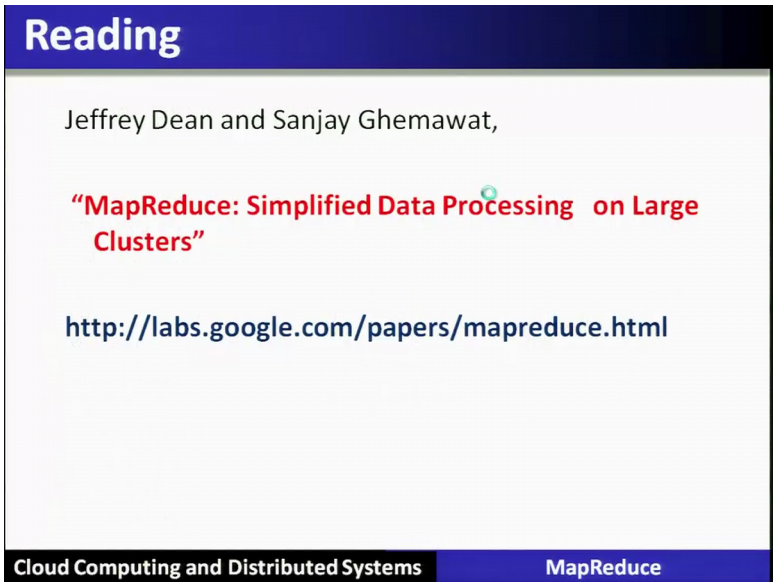
So, intersection will find out the list of common friends and that will be output.

(Refer Slide Time: 55:27)



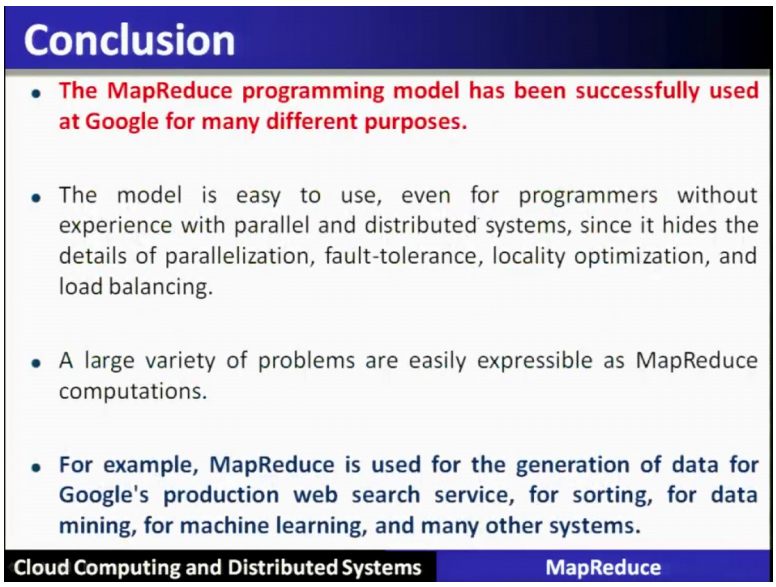
So, this is the result which is shown over here. So, using MapReduce all these particular program become simple, but writing or thinking about this program is not so, trivial.

(Refer Slide Time: 55:39)



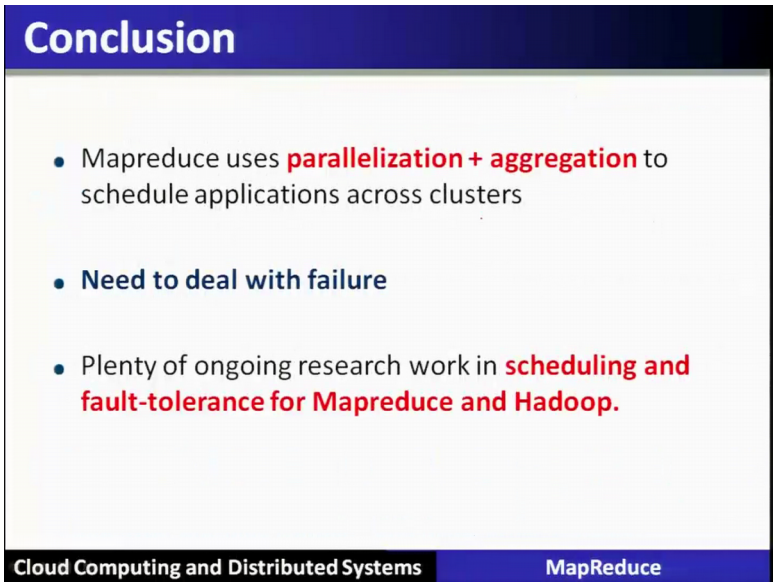
More details of this MapReduce you can refer to this particular paper by Jeffrey Dean and Sanjay Ghemawat that is the MapReduce simplified data processing on large clusters.

(Refer Slide Time: 55:57)



So, conclusion MapReduce programming model has been successfully used in the Google and many different purposes.

(Refer Slide Time: 56:08)



So, we have seen that for a large size data set, this particular simple programming paradigm we have also seen the internal details. We have also shown how it is automatically dealing with the parallelization and dealing with the failures. There are many ongoing research work happening in the in the scheduling fault tolerance in the scenarios of MapReduce and Hadoop.

Thank you.