Noc19-cs33

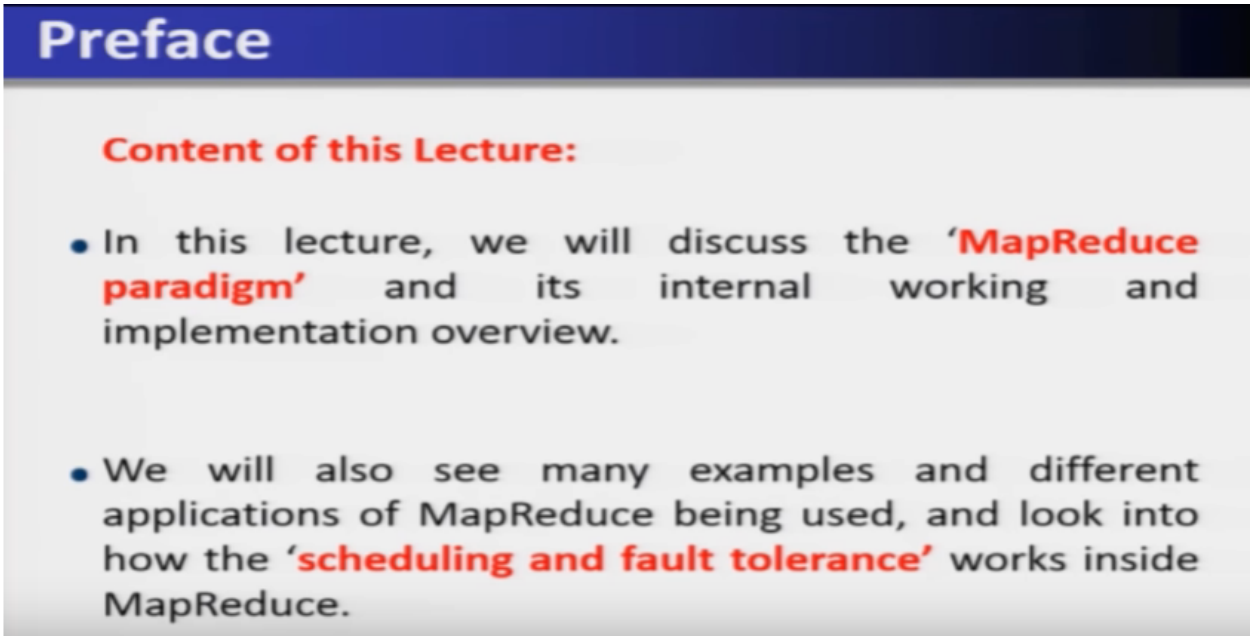
Lec 06-Hadoop

MapReduce 2.0

(Part-I)

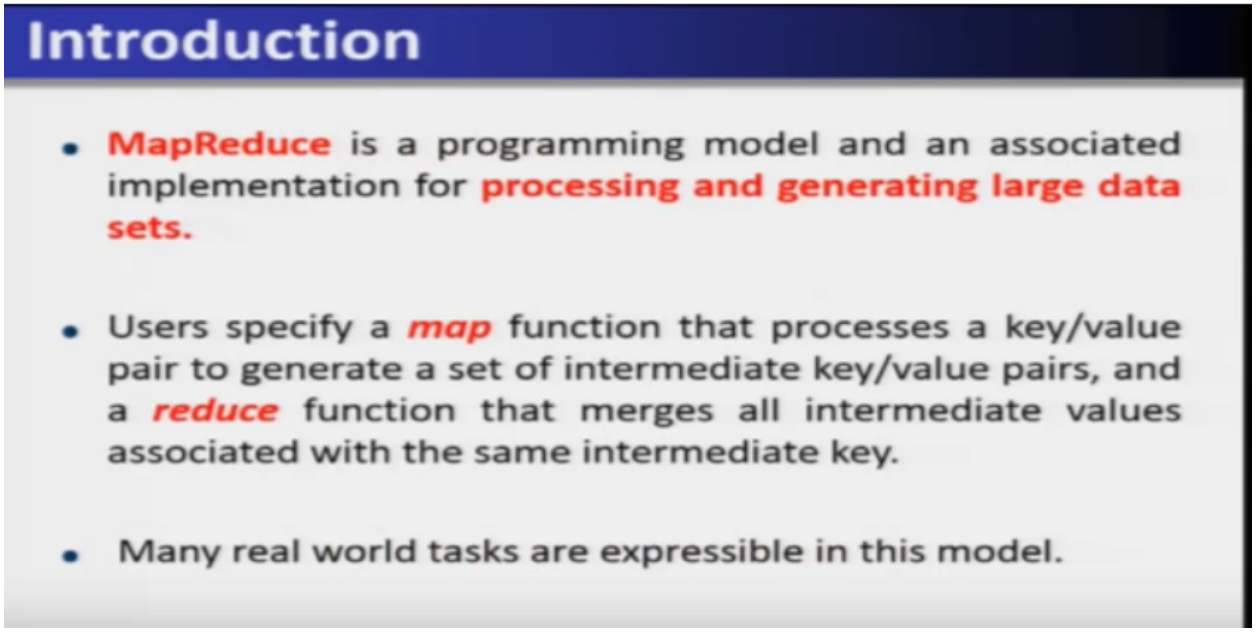
Hadoop, maps reduce 2.0 versions.

Refer Slide Time :(0:17)



Content of this lecture in this lecture; we will discuss MapReduce paradigm and also what's new in MapReduce version 2? We will also look into the internal working and the implementation or we will also, see many examples how, using MapReduce we can design different applications and also look into how, the scheduling and the fault tolerance is now, being supported inside the MapReduce implementation of version 2.

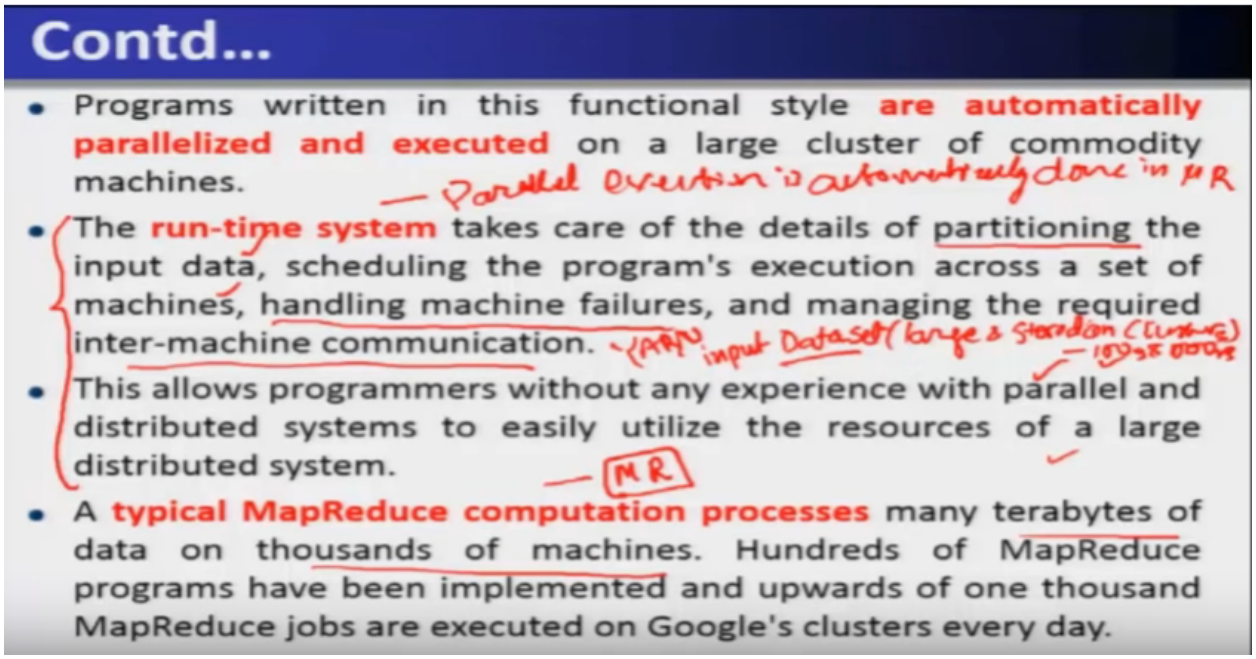
Refer Slide Time :(0: 49)



Introduction, of MapReduce we have to say that, MapReduce is a programming model and associated implementation for processing and generating the large data set. Now, in a version 2.0, MapReduce per version 2.0 now we have, separated out the programming model and as far as the resource management and scheduling, is done using YARN, which is another component of Hadoop 2.0. So, in Hadoop 2.0 if we see this kind of a stack. So, it becomes three different layers. So, the first layer is HDFS 2.0, then, this is Hadoop 2.0 and then comes the YARN and here the MapReduce version 2.0. So, now YARN will, do the functionalities of resource management and scheduling which was a part of MapReduce 1.0 version. So, this will make the simplification, into the MapReduce which is only now, the programming model it will focus on the programming aspect and rest of this particular part that is the resource management and the she ruling will be performed by YARN. So, with this, we can design many new applications on this particular HDFS and YARN, bypassing the MapReduce also. So, we will see, the Hadoop stack or different distributions, which either uses HDFS YARN or HDFS, YARN, MapReduce or top of it lot of applications, are available for big data computations. Now, in this particular framework that is MapReduce version 2 which is, now purely a programming model the users, can specify the map function, that processes a key value pair, to generate a set of intermediate key value pairs, meaning to say that the map function it is accepting the input, in the form of a key value pair. Which is quite trivial why because, any data set we can represent in the form of a key value pair? So, the record is nothing but, a key value pair. So, this will become the input to the map function and output is also, the specified in the key value pair. So, what is the key and value pair? That we will discuss, in more detail in the for the slides but, let us general let us understand let us, us at this point of time that any, set of records or a big data set we can represent easily in the form of a key value pair. It's not a big thing that we are going to discuss. Then the second part is called the, ‘Reduced Function’. So, reduced function merges, all the intermediate values associated with the same intermediate key. So, that means the internet values or the output which is generated by the map, will be the input to the, to the reduced function that is the output of map function that is nothing but, in the form of a key value pair, that is in the intermediate form of the result and then it will, now do the combination it will combine, it and give the final output. It will combine and combine by

the key and give the results out. So, the map functions and reduce together, will perform in parallel all the computations which are desired to process the large data set. And so, many real-world applications or the task we can express using this particular model that is using MapReduce jobs. So, this particular model allows, to execute the application in, in parallel at all the machines which are basically running that the, the task that is, the slave machines.

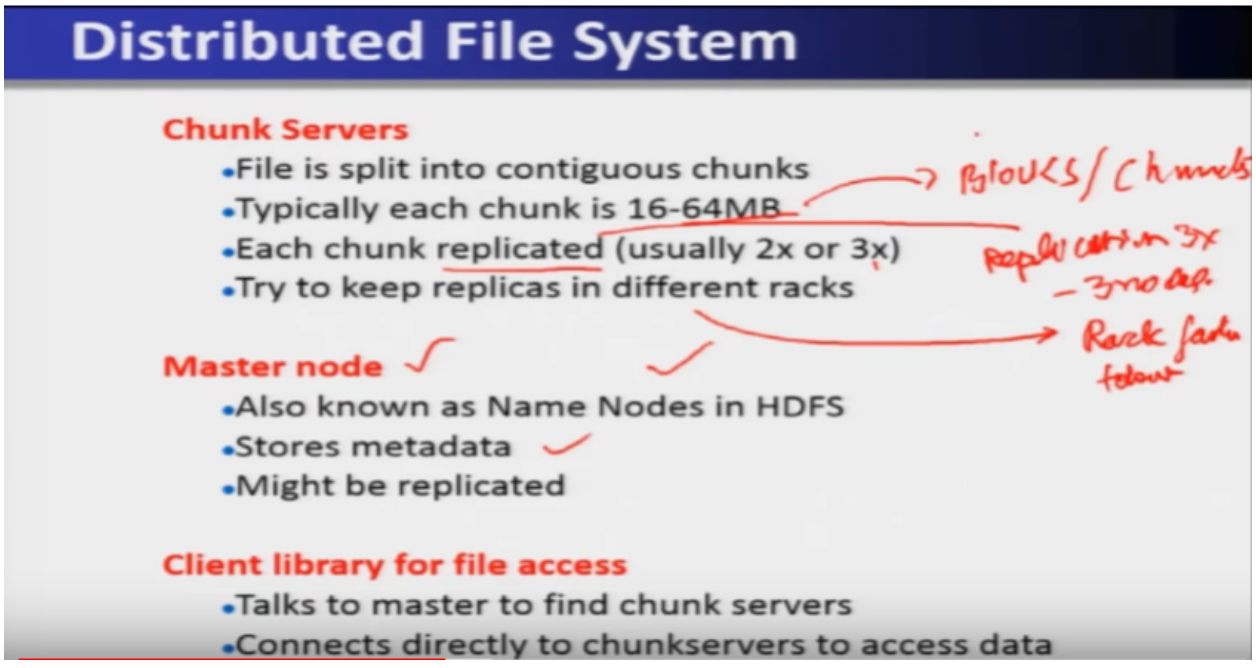
Refer Slide Time :(6: 05)



So, the programs, are written in this style are automatically, are executed in parallel on a large cluster of commodity machines. So, the parallel programming paradigm is automatically being taken care and the users or the programmers, are not going to be bothered about, how the parallelization and how the synchronization, is to be managed. So, the so, parallel execution is automatically done in, the MapReduce framework. Now as far, as the run time is concerned, run time take care of the details of partitioning the data scheduling the programs like equation across the set of machines, handling machine failures, managing the required inter machine communication and this, allows the programmer, without any experience with the parallel and distributed computing to easily utilize the resources of the large distributed system. So, let us explain, this entire notion of a runtime operations which this particular MapReduce allows or provides to the execution environment, to the for the execution of these programs on a large-scale commodity machines. So, as far as MapReduce is concerned now, starting with the data set, the input data set, we assume to be a very large and stored on the clusters, that means may be stored on hundreds and thousands of nodes one data set let us assume that it is stored. So, this will be stored with the help of the partitioning. So, the data set will be partitioned and stored on these nodes. So, let us say that if 100 nodes, are available the entire data set is partitioned into 100 different chunks and stored, in this way, for computation in this scenario. Now, the next task is, to schedule the program's execution, across these, let us say 100 different machines where the entire data set is stored. So, then, it launches the, the programs in the form of a map and reduce, we'll execute, in parallel, at all places ,that is let us say if hundred different nodes are involved, all hundred different chunks, will experience the execution in

parallel. So, this is also, to be done automatically by, the MapReduce using, the job tracker and a task tracker, that we have already seen. And in the newer version that is 2.0, this particular scheduling and resource allocation, this is to be done by the YARN, once this particular request comes from, this MapReduce execution, for that particular job, of that data set. Now, there is a issues like, these nodes, are comprises of commodity, Hardware that we have seen in the previous lecture of HDFS. so the nodes may fail, in case of node failures, automatically ,it will, be taken care by ,doing the scheduling ,of this alternative replicas and the execution still carries on without any effect on the execution. so it requires, also to manage the inter machine communications, for example ,the intermediate results, when it is available ,then, they are sometimes required to communicate with each other ,some values, some results, this is called,' shuffle' and 'combined '.so, shuffle and combine requires, intermission communication, that we will see, in more detail in the further slides. Now, this will allow the programmers without any experience, with the parallel and distributed systems, to use it. And this will simplify, the entire programming, paradigm and the programmers has to only write down a simple MapReduce programs. so we will see ,how the programmers will be able to, write the MapReduce program ,for different application and the execution will be automatically, taken care by, this framework. so a typical, MapReduce computation ,may vary from, terabytes on thousands of the machines .so that we have, already seen ,hundreds of MapReduce programs have been implemented and upwards of 1,000 MapReduce jobs are executed ,on Google's, cluster, every day. So, the companies like Google and other big companies, which works, in the big data computation they uses this technology, which is the MapReduce programs. That means applications are written in the MapReduce that we are going to see them.

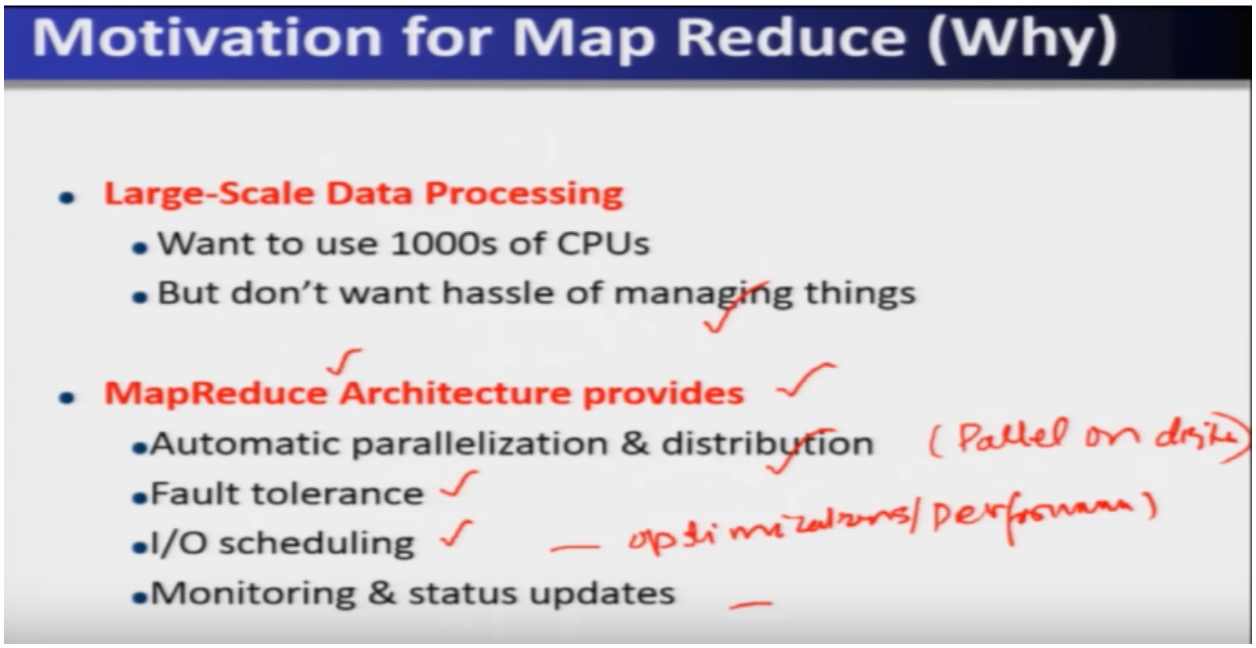
Refer Slide Time :(12:02)



now, we will see, these components, which are used in, the MapReduce program execution and they are the chunk servers, where the file is split into the chunks, of 64 megabytes and these chunks are also

replicated ,this is also sometimes called as a,' blocks'. We call it as chunks, both are synonymous, to be used here in this part of the discussion. Now, another thing is, the these particular chunks are, replicated sometimes, two times the replication all three times, this is called,' replication factor'. So, it's replicated, that means ,every chunk is not is stored on, more than one, that is more than one nodes ,for example ,when it is the replication factor is three times ,so every chunk, is stored on three different nodes. Sometimes, they are try to be stored on a different rack, to have the rack, failure tolerant. So, that means if, the node on one rack or if one rack fields ,so it continues to get the other replicas on a different rack, so that is how, different replications ,are done, to support the, the fault tolerance or the failure tolerance. Now the next component of a distributed file system, which is used here in the MapReduce is called, 'Master Node' also known as the name node in HDFS, which stores the metadata, metadata means the information where the exact, data is stored on which data nodes. So, this information is maintained at the master node by and it is also called as a, 'Name Node' in the terminology of HDFS. Now, another component is called, 'Client Library', for the file axis, which talks to the master, to find out the chunk servers, connects directly to the chunk server, to access the data.

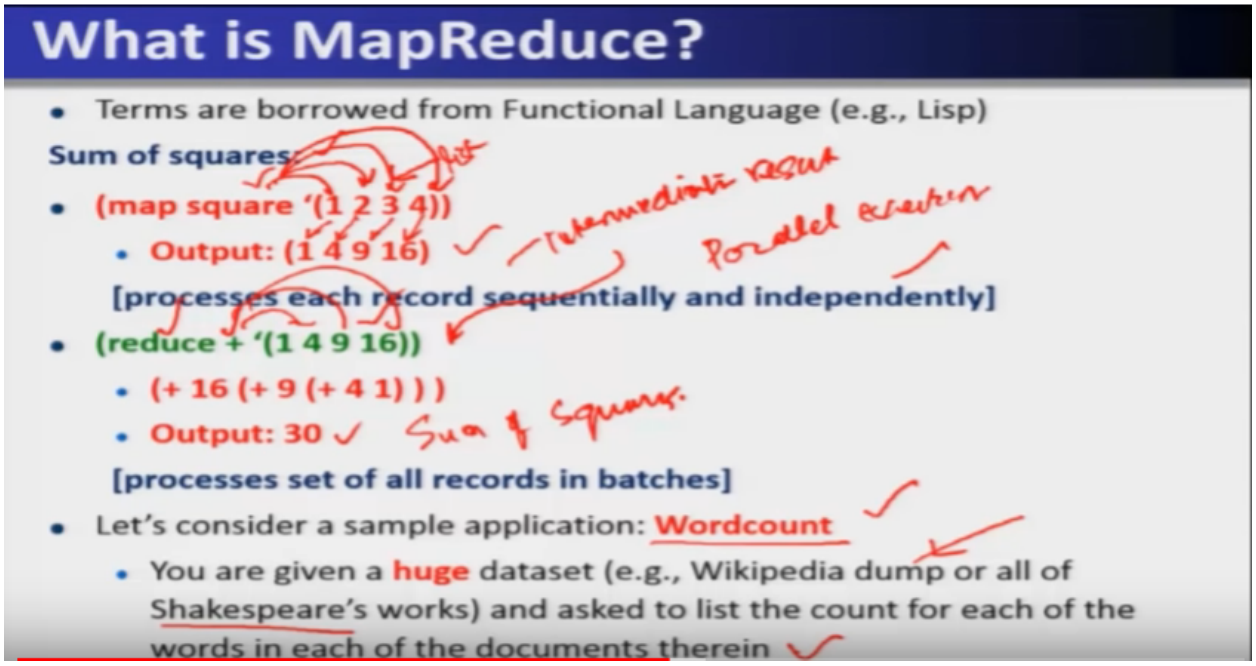
Refer Slide Time :(14: 07)



Now, we will see about, this MapReduce and its motivation why we are? So, much enthusiastic, about this more produced programming paradigm and which is very much used in the Big Data computation. So, the motivation for MapReduce is, to support the large-scale data processing. So, if you want, to run the program on thousands of CPUs and then this MapReduce is the only paradigm available and all the companies, which are looking for the scale? A large-scale data processing the MapReduce is the only framework, which is basically doing all this task? And another important motivation is that, this particular paradigm makes the programming very simple and doesn't require the programmer to go and manage the intricacies of the menu detail, underneath. Now MapReduce architecture also provides automatic parallelization and the distribution, of data and the configuration of it. So, the parallel computation, on distributed system is all abstracted and the programmer is provided a simple MapReduce paradigm and he

doesn't have, to bother on these details automatically, the parallelization and distributed computation, is being performed another important part is the failures. So, the fault tolerance is also supported in the MapReduce architecture and the programmer doesn't have to bother about, that automatically it has taken care, provided the programmer gives sufficient, configuration information so that this fault tolerance is taken care of in different applications, another thing is input-output scheduling, is automatically done. So, those lots of optimizations are used to reduce the number of I/O, operations and also improve upon the performance of the execution engines. So, that is all automatically done by the MapReduce architecture. Similarly the monitoring of all the data nodes, that is the task, that is the task records, execution and their status updates is all, done using the client-server interaction in the MapReduce architecture, that is also taken care of.

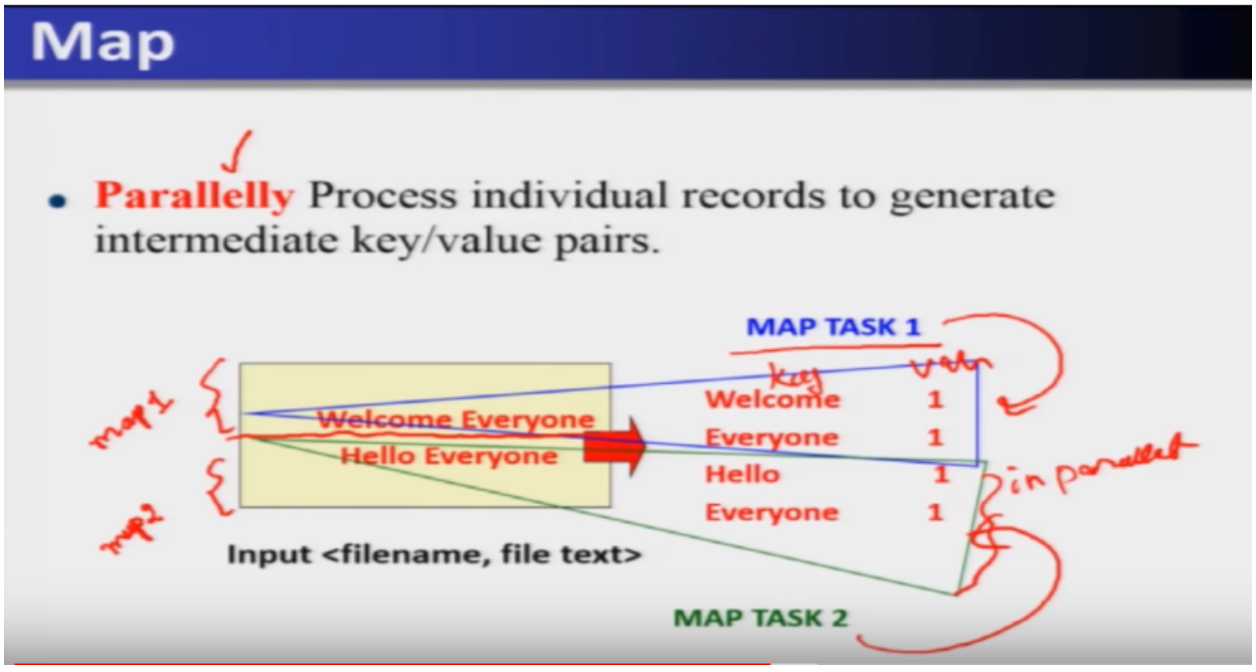
Refer Slide Time :(17: 05)



Now, let us go in more detail of the MapReduce paradigm. So, let us see, what this MapReduce? Is from programmers perspective, this particular map and reduce terms is borrowed from the functional programming language, with let us say that Lisp, is one such programming language, which has this kind of features, that is MapReduce. So, let us see the functionality, which is supported in the functional programming language of MapReduce? And let us say that, we want to; write a program for calculating the sum of squares of a given list. So, let us assume that the list which we are given, is one, two, three, four is a list of numbers and we want to find out, the squares of these numbers. So, there is a function which is called a, 'Square'. And a map says that this particular square function is to be applied on each and every element of the list. So, let us see the output will be, that for that means ,the number the element number 1, 1 when the square is taken 1 will be there, 2 square is 4, 3 square is 9, 4 square is 16. So, all the squares are computed and output is given in again in the form of a list. Now, this particular operation is square, may be executed in parallel, at all the elements and this result will be given very efficiently ,it's not that sequentially one by one, all the squares are being computed. So, therefore it will, process them

independently instead of, these records are not to be carried out in sequentially hence, it's a parallel execution, environment of map function ,of a map function that is, performing the square operation on all the elements, which are being provided in the map function. Similarly to find out the sum this was to calculate the, the squares, now another routine which is required to make the summation, of this intermediate result. So, this result is called, 'Intermediate Result'. So, with this input, we have to perform another operation which is called a, 'Reduce'. Reduce will require an addition operation, the addition on this particular list, which is nothing but, an intermediate result which is calculated by the map function and you can see that, the sum is the binary operator. So, it will start doing the summation, 2 elements at a time and once they are calculated. So, the entire output will be given in the form of whatever is required sum of the squares. So, this is the sum of squares. So, given this kind of functionality or a function of a functional programming language, we will see that, how this particular MapReduce? Can be applied here, in the Hadoop scenario. Now, let us see a small application, simple application, here it, is written a sample application of a word count. So, that means, let us say we have a large collection of documents, let us assume it is a Wikipedia dump, of particular Shakespeare's, works it can be any, different important persons related works, we are going to process this huge data set and we are asked to find out, to list the count for each of, the words in this one document and then query will be to find out a particular reference, in the Shakespeare's works about, the characters how many? Which character is more important based on how much referencing? Shakespeare has done for that character.

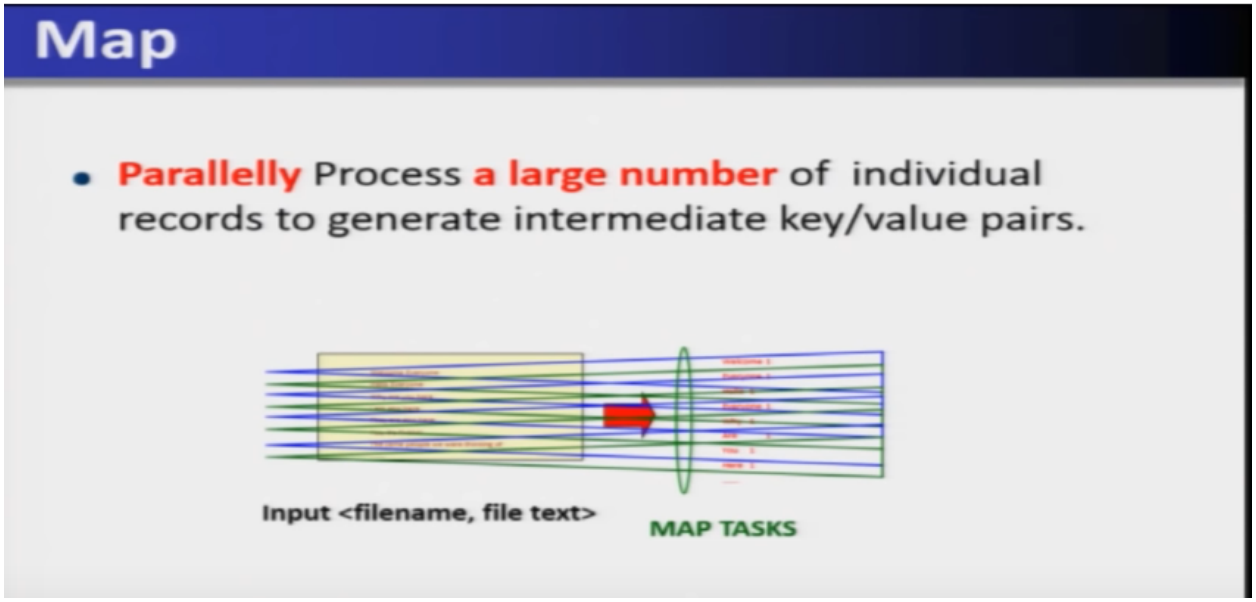
Refer Slide Time :(21: 31)



So, let us see how this is all done using MapReduce? So, MapReduce will huge collection, of data set first it will, be considered in the form of the key value pairs, for example if it is, the document is let us say welcome everyone, hello everyone. So, here this is the key is about, every word it becomes a key and how many times it is appearing in a particular sentence is becomes the value? So, the file is also in a given the input in the form of the filename and the text at the file text so, so this becomes the key, value. So, file will name will be the key and these are text, the value and further on when we process using map

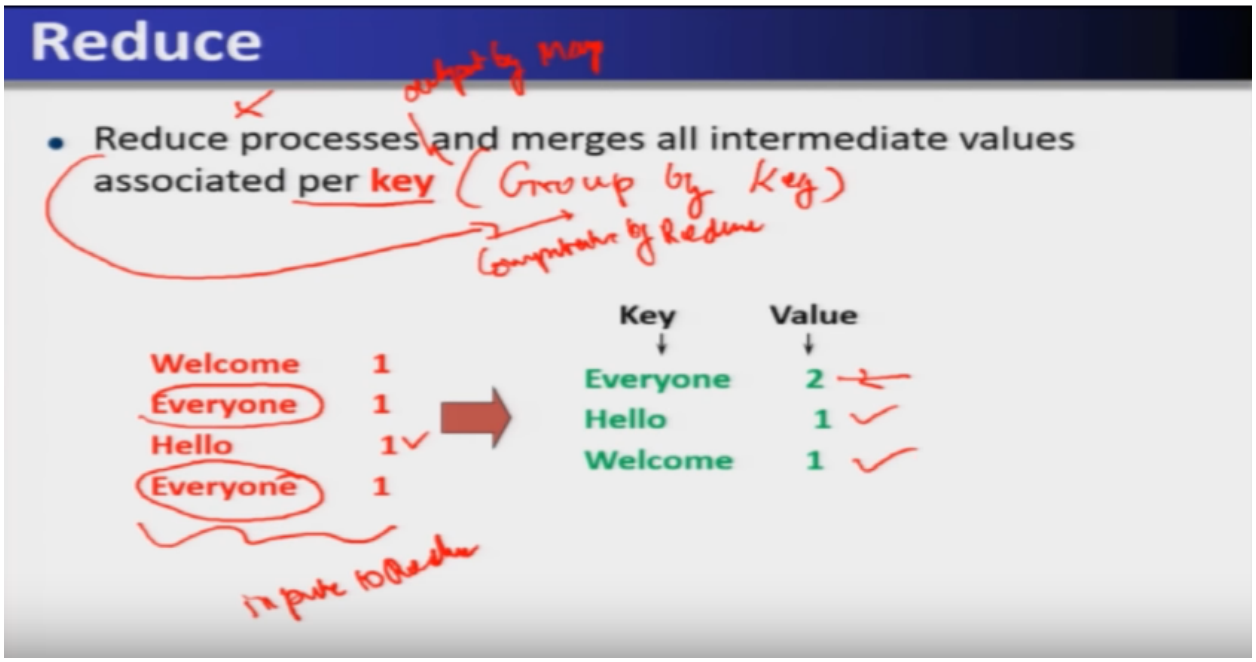
function, when we apply a map function, on this particular input it, will generate this another intermediate key value. This is called, 'Intermediate Result'. In a form of key values, that means for every word, will become a key, every word will become a key, what will become a key? And the instances, when it is occurring, into a particular line of a, of a file then it becomes the value, for example on in a line number one, welcome, is appearing once. So, it gives one similarly in the line number one everyone is also is available only once. Similarly in the second line when it is being processed. So, hello is the appearing one. So, it will be treated as once and on the second line everyone is appearing again, so this particular key value, which is emitted out of map? Is further used by the reduced function. Now, let us say that, let us see how, in parallel this map function can work ,for example these two lines, instead of processed them sequentially let us divide in two, into two different chunks and this chunk, will be given map function one and this another chunk will be given second map function. So, if there are two chunks we are going to process them in parallel. So, this particular task, map task one will execute and map task 2 will also execute, in parallel. So, in parallel if they are executed ,so key this key value pair welcome and everyone, will be collected up by the task number 1, similarly the other task that is hello and everyone, will be collected in the map task 2. So, we need to say that, this particular map function allows, the process individual record to generate the intermediate key value pairs, in parallel and that is done automatically.

Refer Slide Time :(24: 46)



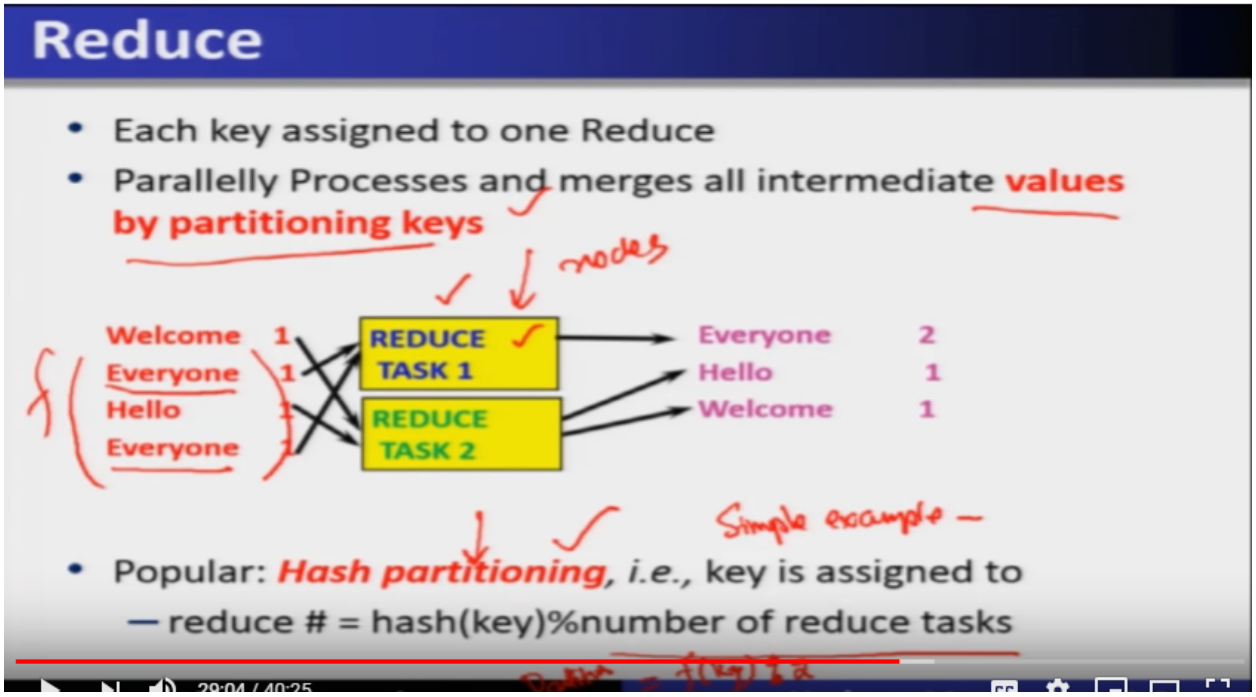
So, parallel process a large number of individual record, generate the intermediate key value pair, that we have seen if these numbers or the chunks, are increased more than, two then, this is, an example where multiple map functions are simultaneously, at the same time that is in parallel they will, execute and process these records.

Refer Slide Time :(25: 09)



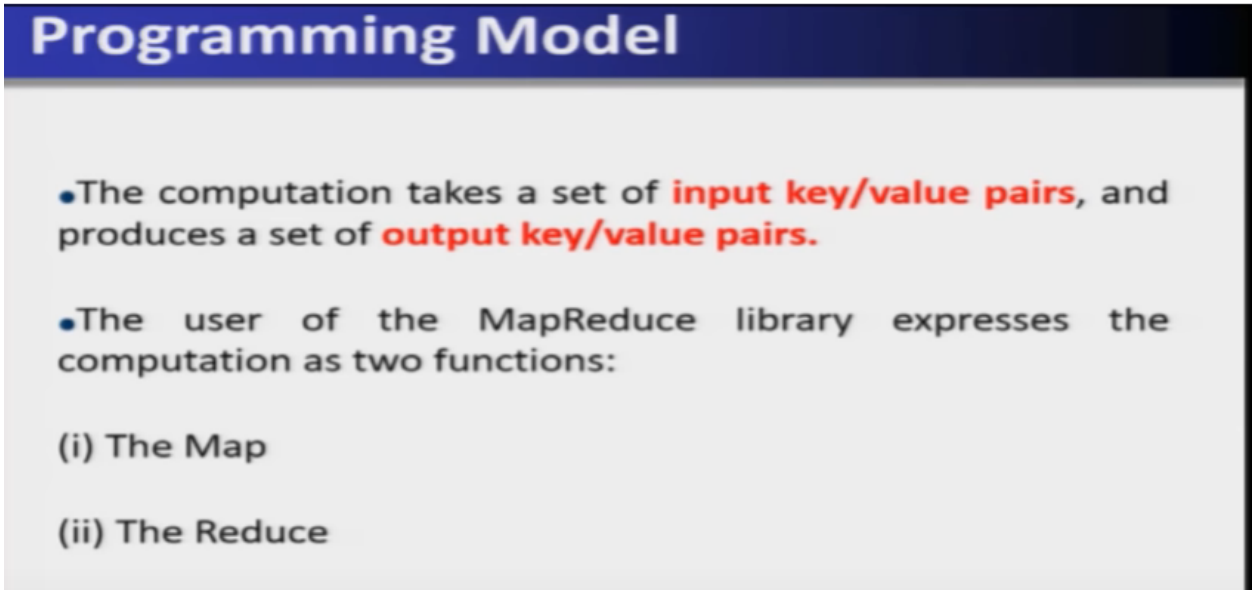
Now, we will see the next phase that is called, 'Reduced Operation'. So, after the map the intermediate values, which is being generated, in the form of the key value pair will be used as the input for the reduced function. So, reduce processes, these intermediate values, which is output by the map function and then merges all the intermediate values, associated per key, that means based on the similarity of the key values, they are all combined together that is group by key operation, is being performed and using, in this particular, group by key operation whatever is the reduced function applies, this will be the computation will come, from the reduced function. So, computation on this group by key, will be applied by, the reduce function to get the entire output, for example if this is, the input which is, input to the reduced function and we want to do, a word count. So, now what it will do is, it will group by the key when you say group by the key then, everyone is appearing twice. So, with everyone there are two different values or two times, this everyone one will come and this will be added up by the reduced function, similarly hello, one is appearing once. So, group by key is, having only this tupple or entity similarly for welcome. So, this particular reduced function, will execute in this particular manner.

Refer Slide Time :(27: 05)



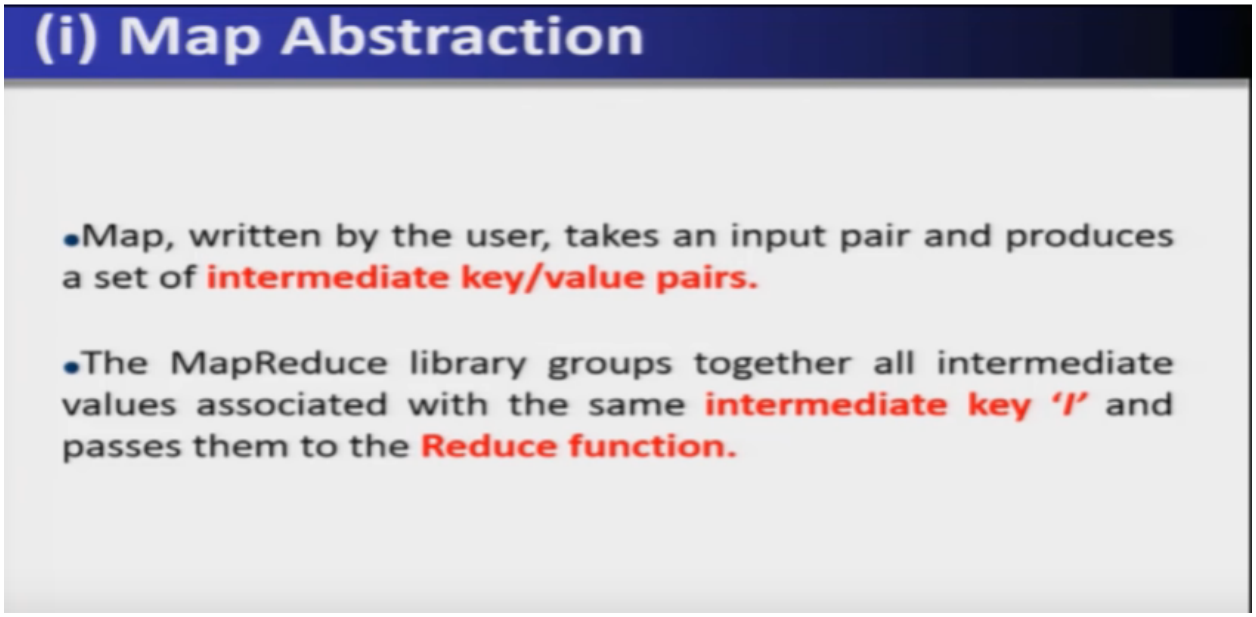
So, each key assigned to one reduce function and here we can say that if there are two different nodes, which runs this reduce task, let us say task 1 and reduce task 2. So, each key is, assigned to one reduce task for example this, particular key will be assigned to the task 1. So, when this second key also second time when everyone comes it should also, assign to that same reduce task, whereas welcome and hello can be assigned to the reduce task 2. Now, parallel process and merges all, the intermediate value by partitioning keys that, we have already told you, but how, these keys are to be partitioned and the simple example is, let us, say that use a, hash partitioning, that means if you have a hashing function f, and when you apply this has have a hashing function, on a particular keyword, it will give a particular value and this value will be mod, how many number of reduced tasks here? Let us, assume that it is, the number of tasks is 2 and modulo, some hash function and on that particular hash function, we have to define a key and this will generate the, the partitioning. So, partitioning we have shown you through, a simple example, the way partitioning is carried out, that is through the highest partitioning, it can be very complicated also depending upon different application how ,the partitioning is done and how many different worker nodes are reduced nodes are there or reduce task is allocated by the yarn? And based on that this particular partitioning is done and after partitioning then it will merge, the intermediate values and gives, the result back.

Refer Slide Time :(29: 09)



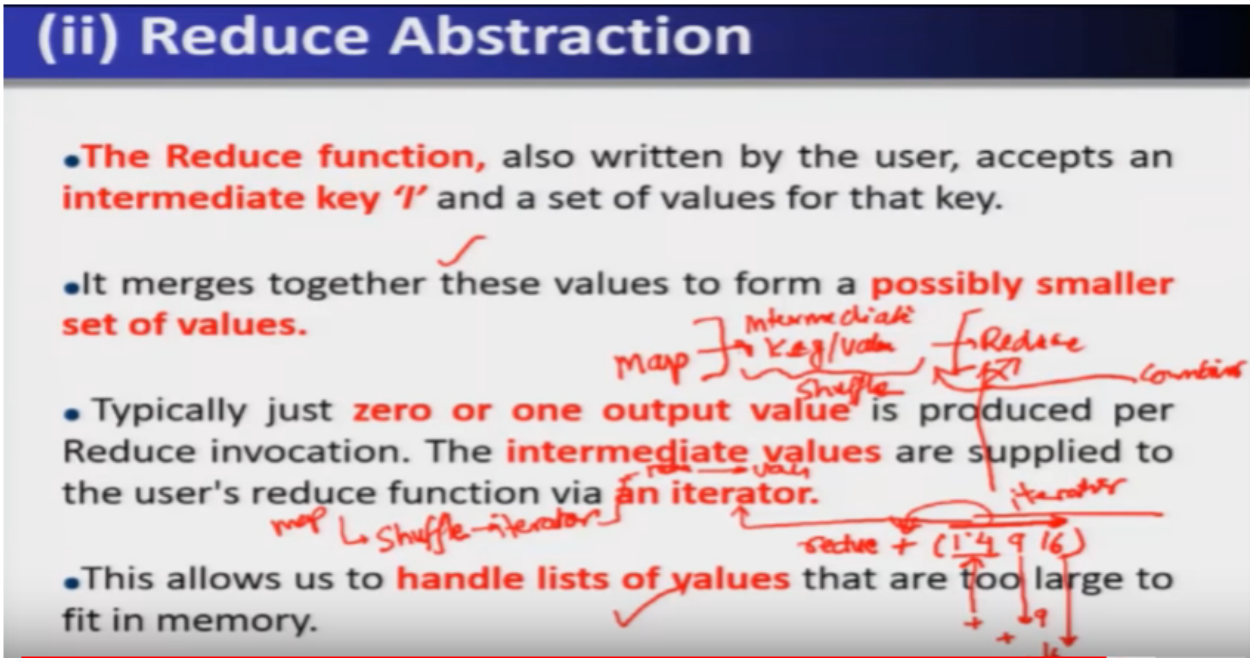
Programming model, the computation takes a set of input key value pairs and produces a set of output key value pairs, in this programming model. To do this, the user has to specify using MapReduce library the two functions, the map and reduce function, which we have already explained.

Refer Slide Time :(29: 38)



Map abstraction is written by the user which will take the input pairs and produces the set of intermediate key value pairs. This function is to be written, based on the applications and the programmer has to, write down as per the application what is to be done in the map function? And the remaining part will take care in the reduced function. So, the reduced library, MapReduce library, groups together all intermediate values associated with the same intermediate key and passes them to the reduced function that was the map abstraction.

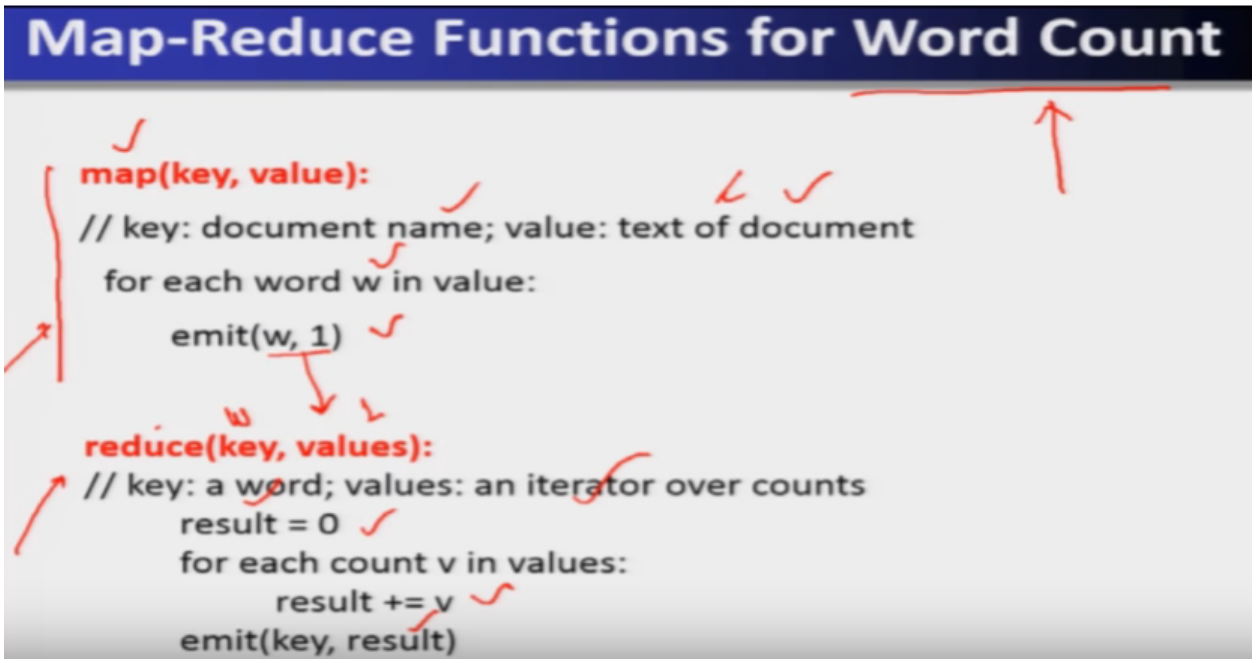
Refer Slide Time :(30: 21)



In the reduced function, also it is written by the user, which accepts an intermediate key and a set of values for that particular key, it merges together these values, to form a possibly a smaller set of values, typically just 0 or 1 output values produced per reduce invocation, the intermediate values are supplied to the users, reduced function y and an iterator. This allows us to; handle the list of values that are too large to fit in the memory. So, that means, if we see the outcome of the map function, it will generate the intermediate key value pair, which will be taken in the reduce phase as the input. So, here, what it does is? This stage which is called a, 'Shuffle'. It merges together, shuffle and the combiner, here there is a combiner, sometimes a combiner is a part of both the activities map and reduces. So, the combiner. So, first it will shuffle, that is it will arrange, according to the group by keys and this is called a, 'Shuffle'. And after that it will combine, that is all the same keys, with the values to be combined and is given to the reduce to operate. So, therefore, 0 r1 output values is produced, per reduce invocation the n iterator intermediate values, supplied to the users reduced function, wire and n iterators meaning to say that this particular function, which is called, 'Iterator'. Can be understood, in more detail like this, for example if you want to do, a summation of the intermediate values which is, basically the sum of squares. Let us say, we want to, do this, particular summation of all these things. So, iterator has, two for example if there are, maybe another example, iterator has to evolve for example if there are yeah, iterator has to evolve or to see the, the elements which are similar, it has to scan and this is called, 'Iterator'. So, it will scan through all, the elements and perform the operation together and for example one and four, can be treated together. So, iterator here will identify, a binary or two set of elements, first two elements and perform the plus operation and then it will, take the next operation, next element and perform the plus operation and then perform then take the next element 16 and performed on the plus operation to become the overall sum. So, the iterator job is, to scan, through this particular list and provide the elements, for the operation which is specified in the reduced function. So, we have seen, that out of the map function the, there is a activity, which is called a, 'Shuffle'. Which will sort, the elements? So, that all the key value, key values,

with the same key group by same key they are to be sorted and given back to the next phase. And in the reduced function, it will use the iterator, to scan, through this particular list of elements, which are already sorted and provided by the shuffle. So, after the map function, lot of internal communication and activities are going on, in the form of shuffle and iterator and then only the reduced function will be, applied and give the values out. So, this allows, these operations allows us, to handle the list of values, that are too large to be fed, in the memory and why because? They are not, to be brought at one place but, they are to be, operated wherever these use our reciting, hence it is, possible that the, the operations are performed, even on a very large data set which cannot fit, into a particular memory.

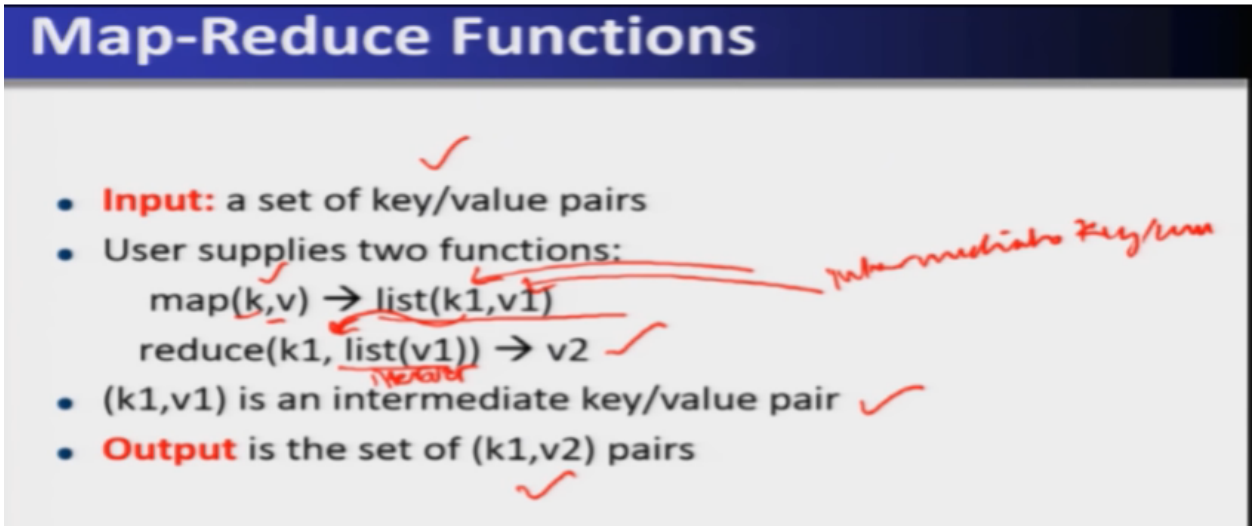
Refer Slide Time :(35: 54)



Now, we will just see, the structure of the map and reduce function which programmers used to write down and here, we take the example of a word count. Will not go in detail in explanation at this stage, explanation we will see in more detail in the further slides. So, as far as, the map function this is the pseudo code, of a map function it cannot be executed in this form, why because? It has to be programmed as per the specification of that particular language, of MapReduce but, we are going to see the, the pseudo code. So, pseudo code has to specify the map function with the key and value and where key here, in the word count example, is the name of the document and the values are the, the text, of the document which are be retrieved and given in this map function. So, for each word, which is basically getting as the value, in this particular for each word in the value, which is appearing the, map will omit, that particular world and one. Now, the next pseudo code, for reduce takes, the same format as emitted out of the map function. So, that means the world becomes, the key and the value becomes as per the value over here, that format should match then only the output of map can be acceptable that is the, the reduce function. So, here, it is explained that the key, is the word and values is basically the values, when it is grouped by key then, the iterator has to operate, over this set of values. So, let us take this set of values and we perform the reduce operation. So, reduce we'll take all these values, out of iterator and does the summation and results, in to

that, how many times that key is appearing? That is it. We'll do the summation of all the values but, this has explained .

Refer Slide Time :(38: 23)



The MapReduce paradigm and let us, see some more detail of it and here, we have given the input as a set of key value pairs and the user has to specify, the function map, wherein he has to specify, the what is key and what is value? And after that this map function will emit, intermediate values in the form of a list, which is having a key, K 1 and V 1, is the values. So, K 1 and V 1 is the intermediate key value pair, and this intermediate key value pair, is given to the reduce by the shuffler. So, once the reducer will get the key and intermediate key and value pair. So, it will get the entire list of values and here, the iterator will operate on it and applying, the reduced function it will emit that particular value. So, the output will be, the K 1 and V 2 values combined. So, this is the abstraction of MapReduce and overall picture how the program, can be written using MapReduce paradigm. Thank you.