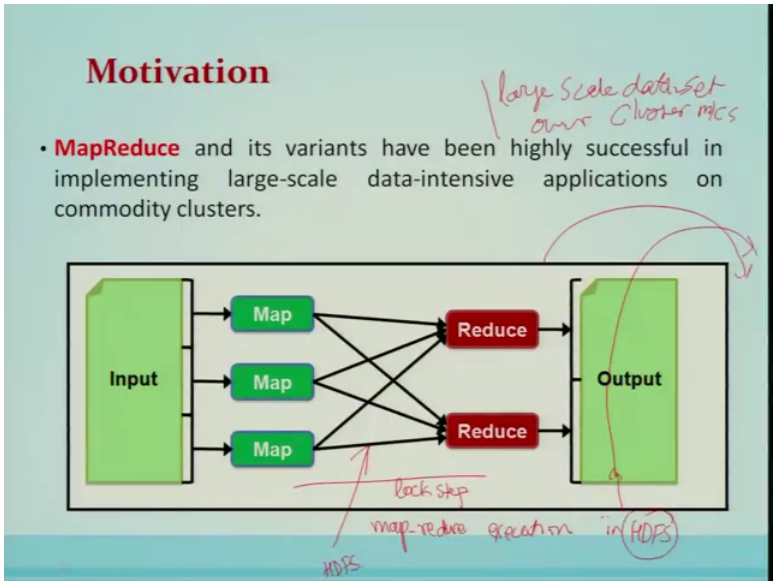
Distributed Systems
Dr. Rajiv Misra
Department of Computer Science and Engineering
Indian Institute of Technology, Patna

Lecture - 23
Spark

(Refer Slide Time: 00:18)

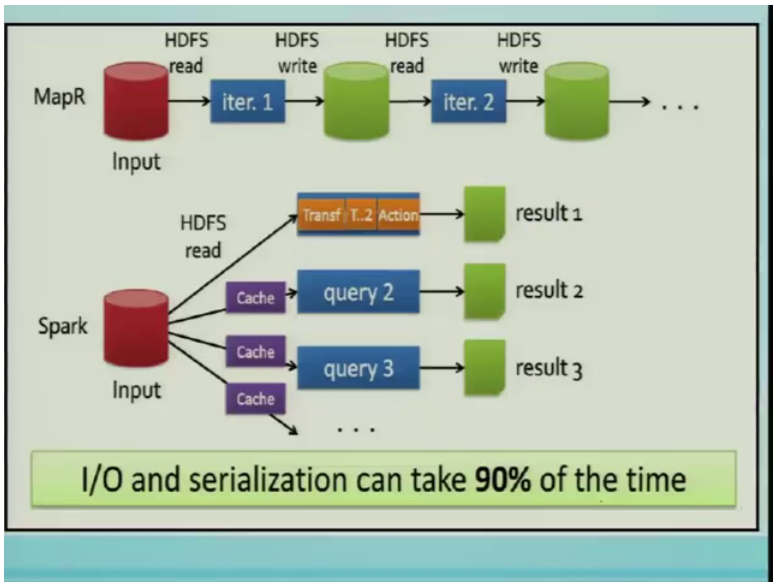


Motivation. So, the MapReduce and its variants have been highly successful in implementing large scale data intensive applications on commodity clusters. Meaning to say that, this MapReduce has successfully shown that, it is able to compute large scale data sets over cluster machines.

However, if you see this particular MapReduce execution, then we will see that this map and reduce, they work in a lockstep manner, and the output will be recorded in HDFS, here also HDFS. So, if the next iteration, if the program has more than one iteration of a MapReduce, then the next iteration has to basically come out, the input has to come out from HDFS.

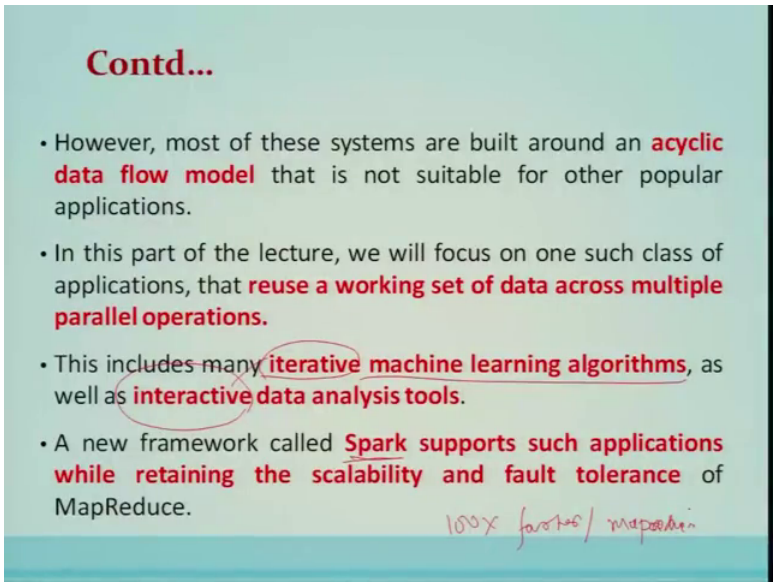
So, if the applications which has more than one iterations, then they have to each every, means the next iteration has to touch or has to access the HDFS, and that basically have an intensive input and output and serialization.

(Refer Slide Time: 01:59)



And will take 90 percent of the time in I O operations.

(Refer Slide Time: 02:03)



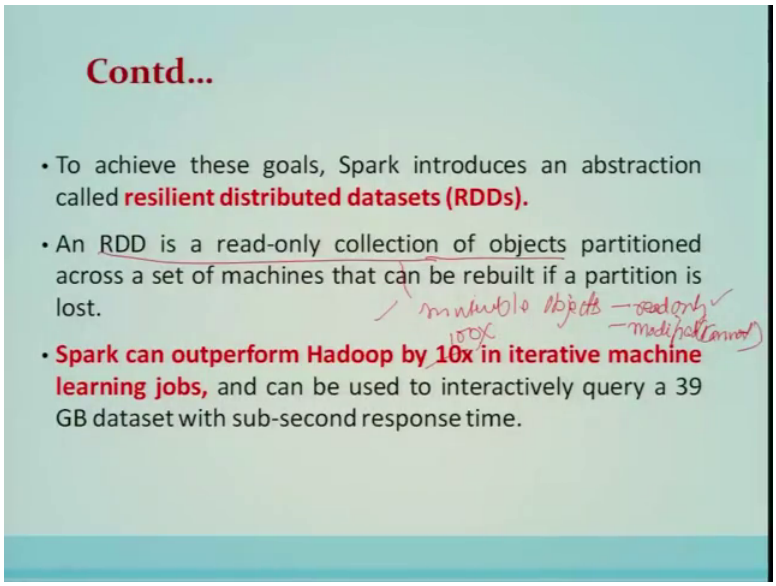
That was basically the disadvantage. So; however, most of these systems are built around an acyclic data flow model, and that is not suitable for many applications. In this lecture we will focus one such class of applications that re use the working set of the data across multiple parallel operations.

So, this includes many iterative applications; such as machine learning algorithms have the iterations, and also the interactive application such as data analysis tools. So, these

applications may require, this kind of access to the HDFS file system between different iterations or across different parallel operations.

So, therefore, a new framework which is called as sub is spark will support such applications. So, not only it will support such applications, but also it will improve the execution time; that is it will be tend hundred times faster, compared to the Hadoop mapreduce.

(Refer Slide Time: 03:34)

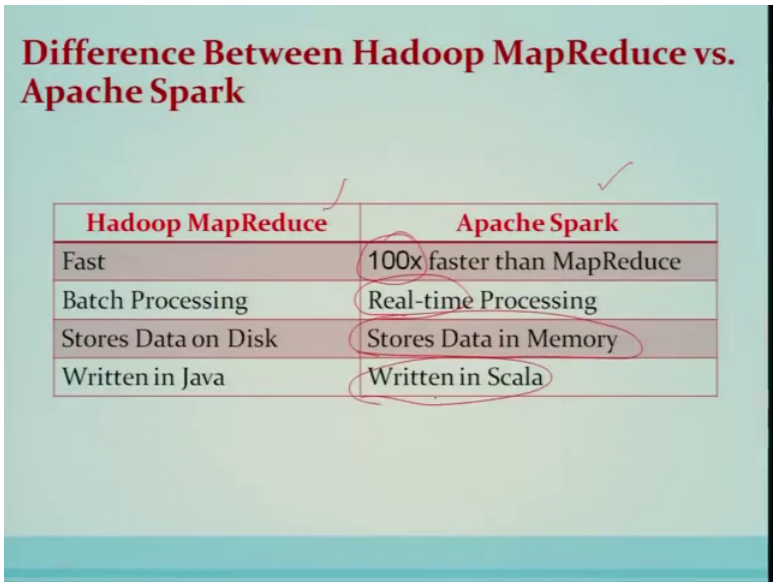


So, the new framework which is called a spark will support such applications, while retaining the scalability and fault tolerance of MapReduce. So, to achieve these goals, spark introduces an abstraction which is called a resilient distributed datasets. Resilient distributed datasets is a read only collection of objects, and they are partitioned across the set of machines that can rebuilt if the partition is lost. So, this RDD is a read only collection of objects, meaning to say they are mutable objects.

Mutable objects means they cannot, they only, they are only read only, and cannot be modified. So, if it is a read only, and if failure has happened at some partition, at some node then it can be rebuild, and can resume the execution without any much disruption.

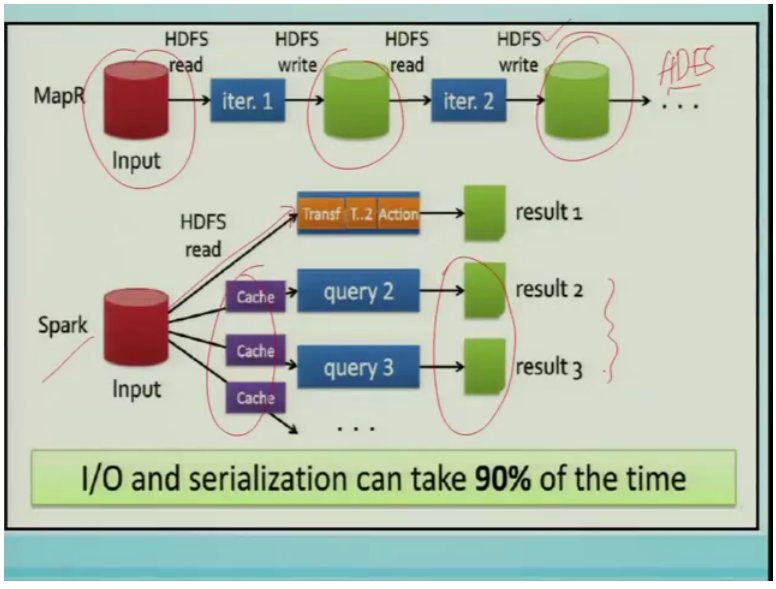
So, spark can outperform Hadoop by at least hundred times in the iterative machine learning jobs, that we will see how it achieves this.

(Refer Slide Time: 04:48)



So, if we see the difference between the Hadoop MapReduce and the spark, we will see that its speed is hundred times faster than the MapReduce operations, and here it also supports real time processing, and it also stores the data in the memory, and its particular spark applications are written in Scala language.

(Refer Slide Time: 05:14)

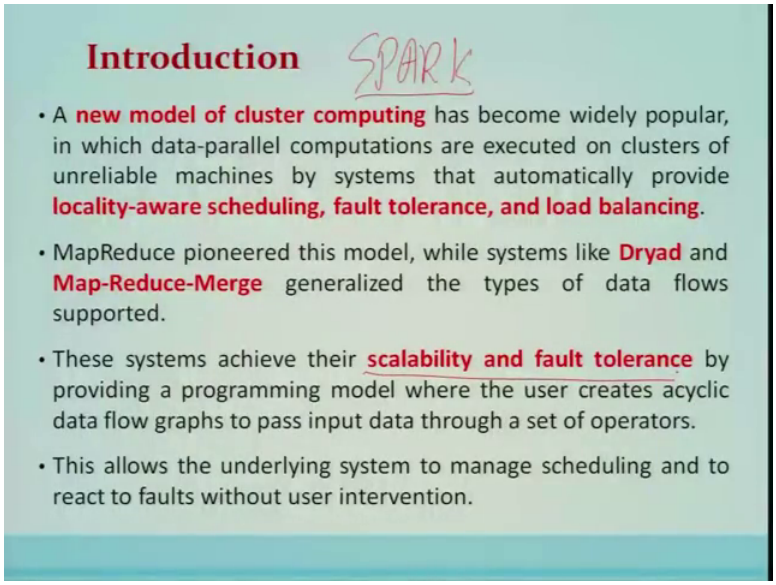


This is an example which will show that, this particular Hadoop MapReduce, will during this particular iterations, has to access this hard disk or HDFS file system, which is on the persistent memory. So, spark in contrast to the MapReduce, you can see that there

will be only one read operation from HDFS, and after that thus particular data will be cached, and across several iterations, this particular in memory cache will be utilized, and the results will be accessed.

So; obviously, it is going to reduce 90 percent of the I O operations, why because in memory operations will be most effectively utilized. And hence a newer application like interactive and iterative application, will be executing here efficiently; that is hundred times faster compared to the MapReduce operations.

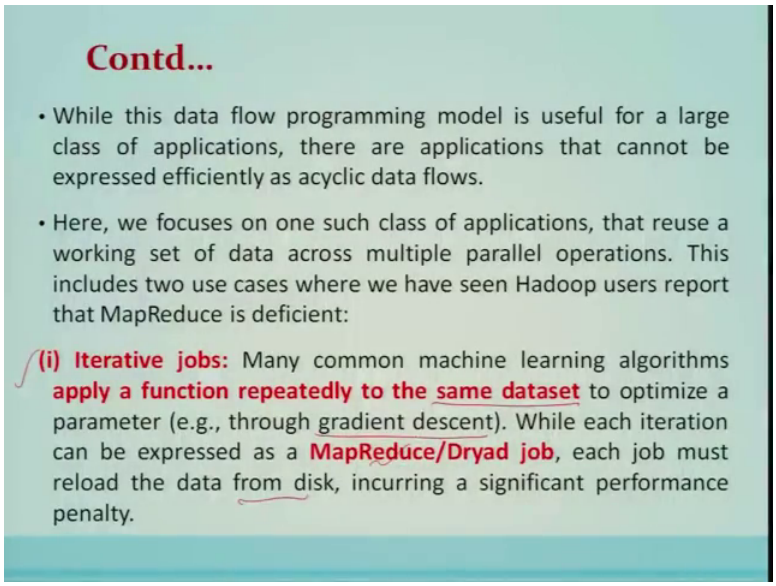
(Refer Slide Time: 06:25)



So, the introduction. The new model of cluster computing; that is called the spark, has become a widely popular in which the data parallel computations are executed on the cluster of unreliable machines, by the system that automatically provides locality aware scheduling fault tolerance and load balancing.

Now, these systems achieve their scalability and the fault tolerance by providing a programming model, where the user creates a cyclic data flow graph to pass the input through the set of operators.

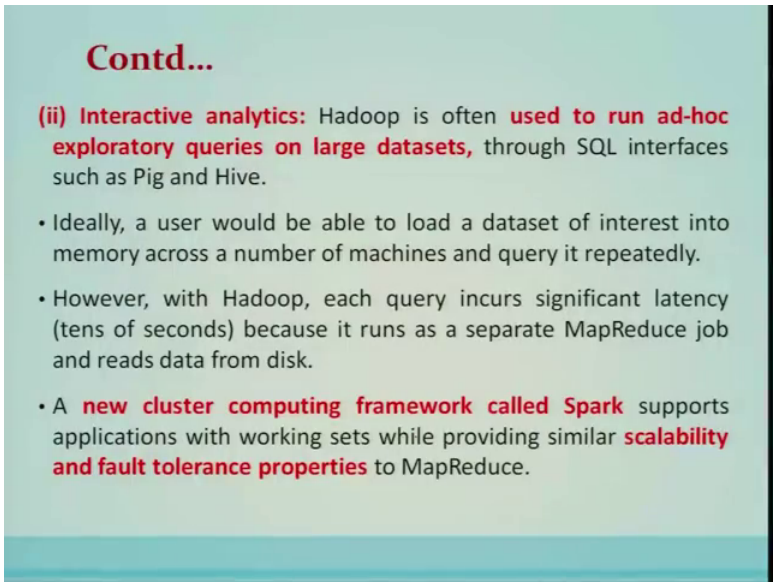
(Refer Slide Time: 07:05)



So, while this dataflow programming model is useful for large class of application. There are some applications that cannot be expressed efficiently as a cyclic data flows. So, one such application is called an iterative, having an iterative jobs. So, where many machine learning algorithms, they apply a functions repeatedly on the same data sets, to optimize the parameter.

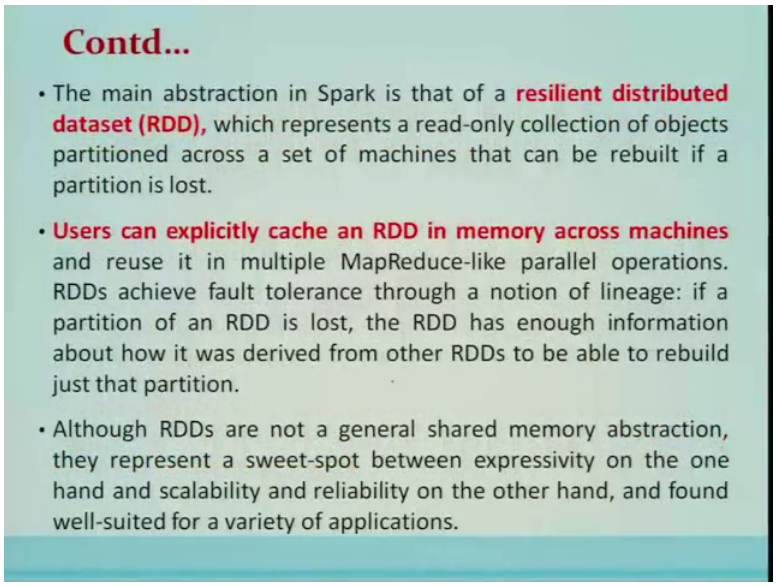
For example, in a gradient descent, machine learning algorithm, or a PageRank algorithm, while each iteration can be expressed as Map Reduce each job must be, must reload from the disk in the MapReduce. So, how this can be avoided, and spark will show that this axis can be avoided.

(Refer Slide Time: 07:58)



Interactive analysis Hadoop can be used. Hadoop is used to run ad hoc exploratory queries on large datasets through SQL interfaces. So, user would be able to load the data sets of interest into the memory across the number of machines and query it repeatedly.

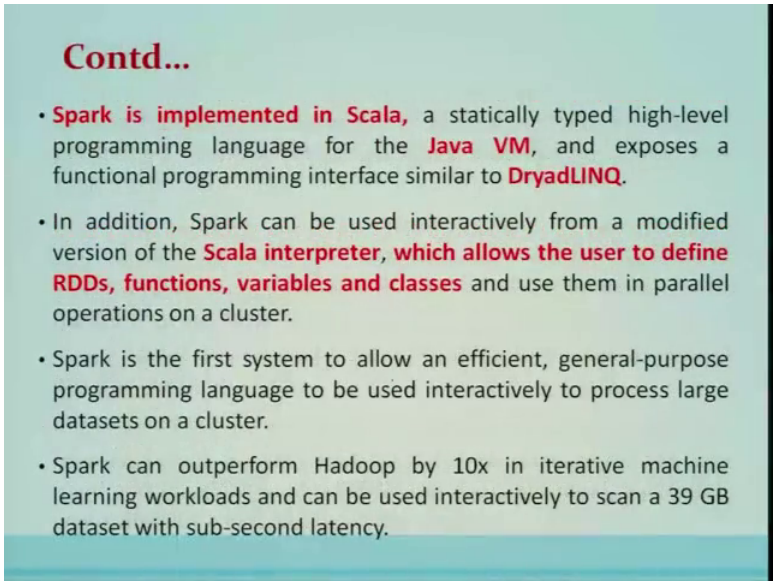
(Refer Slide Time: 08:25)



So, a new cluster computing spark will support applications with working sets, while providing scalability as we have seen. So, the main abstraction of the spark is resilient distributed datasets. Users can explicitly cache an RDD in memory across the machines and reuse it in multiple MapReduce like parallel operations.

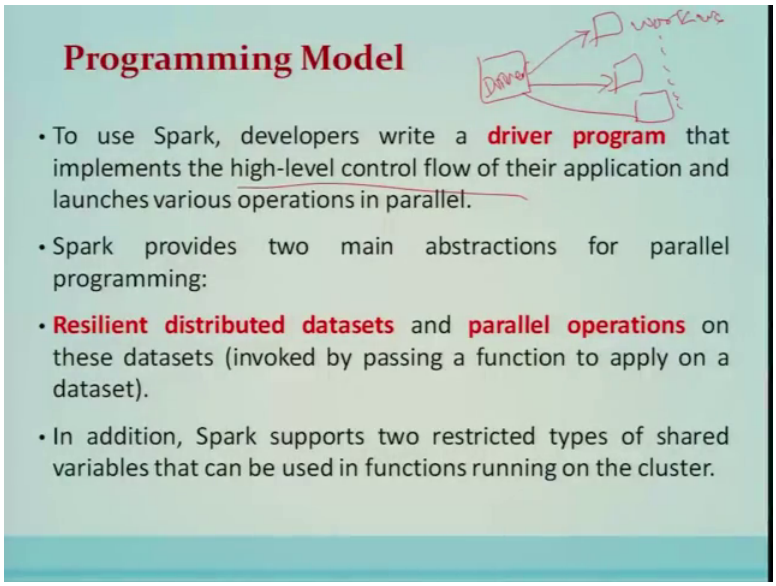
RDDs achieve fault tolerance through the notion of lineage; that is if the partition of an RDD is lost the RDD has enough information, about how it was derived from other RDDs to be able to rebuild just that partition. Although RDDs are not a general shared memory abstraction, they represent a sweet spot between expressivity on one hand and scalability and reliability on the other hand.

(Refer Slide Time: 09:14)



Spark is implemented in a Scala, spark can be used interactively from a modified scalar Scala interpreter. Spark is the first system to allow efficient general purpose programming model to be used interactively to process large datasets on a cluster Programming model.

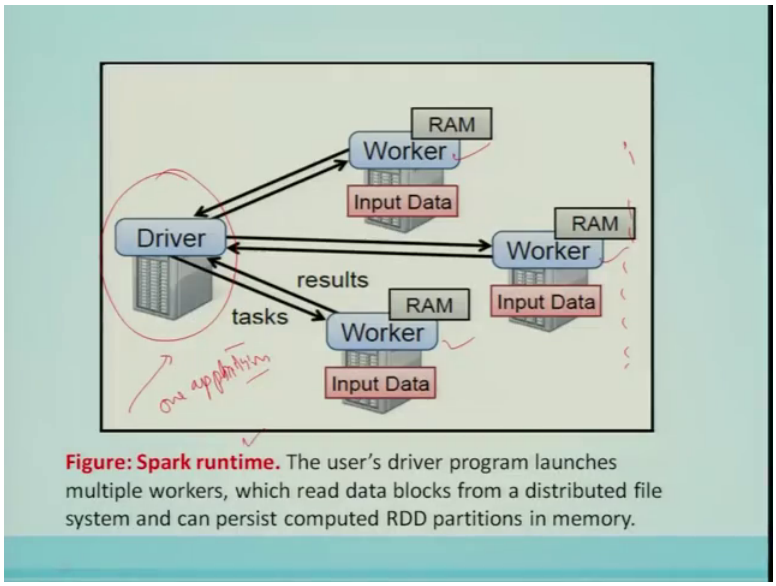
(Refer Slide Time: 09:35)



To use this spark developers write the driver program, that implements the high level control flow of their application and launches, various operations in parallel. So, for example, this is a driver program, which basically will drive the whole entire execution, and then driver will give the instructions, and that will be through a high level control flow, and they will be the kind of workers, which will run in parallel by the spark driver program.

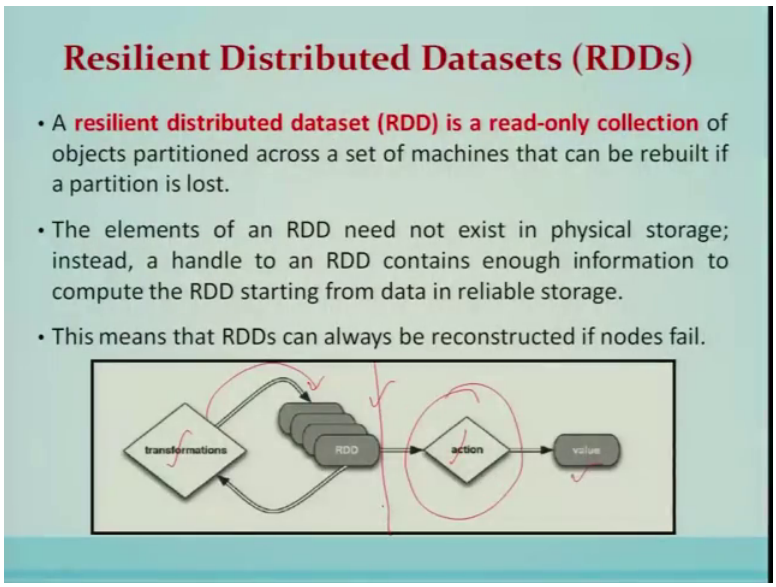
So, the spark provide two main abstractions for parallel programming; one is resilient distributed datasets, and parallel operations on these datasets. In addition this spark provides supports two restricted type of shared variables that can be used in the functions running on the cluster.

(Refer Slide Time: 10:47).



So, as I explained that the spark program spark run time will have the driver program, and this driver will distribute the task to different workers. So, there can be three it is shown, but there can be more than many, as many as the number of nodes possible, maybe thousands or hundreds, such workers can work for a driver, and driver will run one application.

(Refer Slide Time: 11:26)



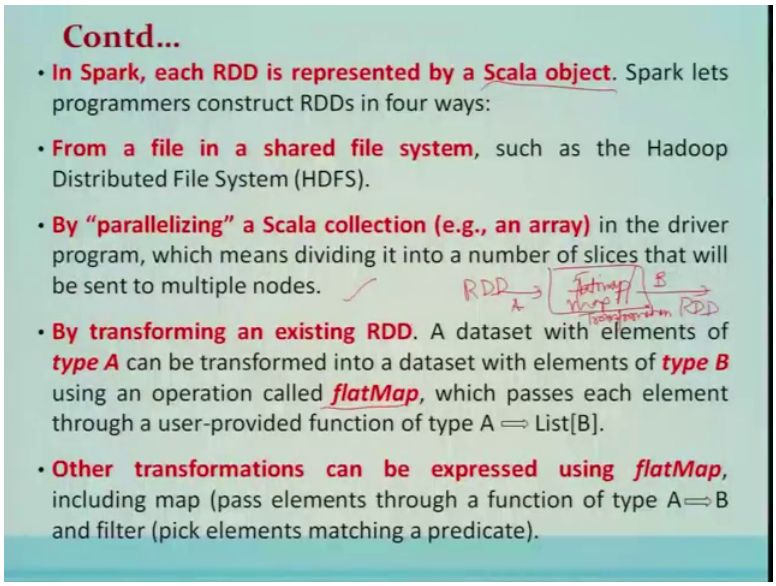
So, this is the scenario of one application execution. So, RDDs resilient distributed data set is a read only collection of object, partitioned across a set of machine that can be

rebuild if the partition is lost. The element of an RDD need not exist in the physical storage instead a handle to an RDD contains enough information to compute RDD starting from the data in a reliable storage. This means that RDDs can always be constructed, reconstructed if the node fails.

So, here we can see that, once an RDD is defined, then we can perform an different transformations from an RDD. So, we will see what these transformations are supported by the spark, and then on RDDs various actions can be performed which will give the final values.

Now, we will see that these actions are important, in between the transformations which are defined, will not be executed until actions are fired, and these actions will give the values that we are going to see.

(Refer Slide Time: 12:37)

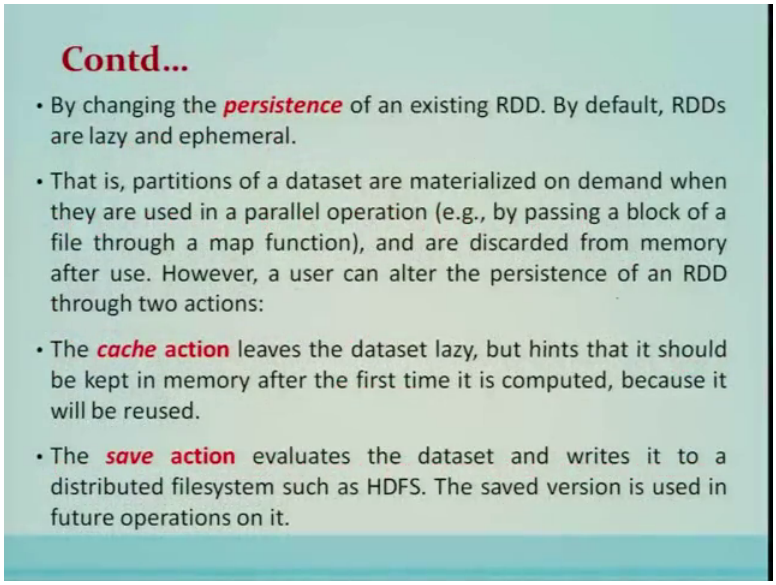


So, in spark each RDDs represented by any scale object. So, there are four ways. So, from file in a, shared file system such as Hadoop distributed file system, by paralyzing Scala collection in a driver program which means that dividing it into a number of slices that will be sent to the multiple nodes, by transforming an existing RDD, a data set with the elements of type a can be transformed into a into datasets, with the element of type B using operation flatmap, which passes each element.

Other transformation can be expressed using flatmaps. So, flatmap is a transformation which takes RDD in one form. Let us say A, and it will transform and give RDD in another form, and this is called a transformations.

So, whether it is flatmap or a map all are transformations defined in by Scala that we will see.

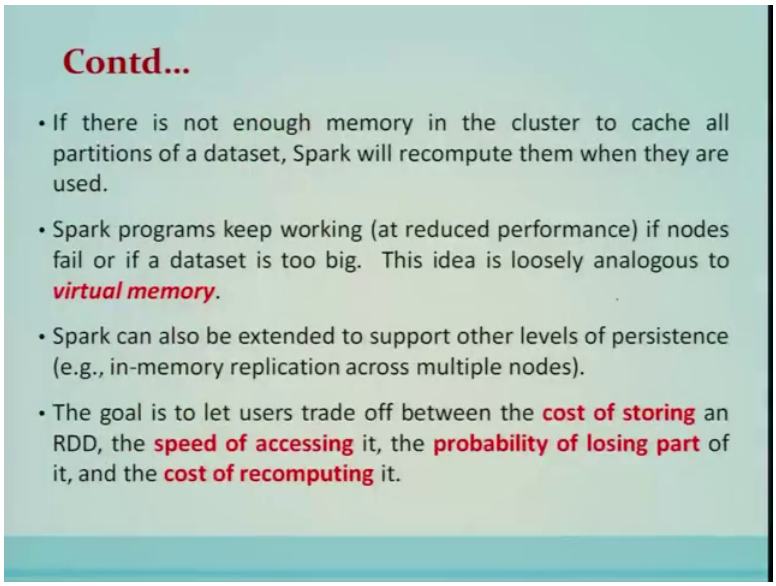
(Refer Slide Time: 13:52)



So, by changing the persistence of an existing RDD, by default RDDs are lazy and ephemeral; that is partitions of the data sets are materialized on demand when they are used in parallel operations, and discarded from the memory after the use is over

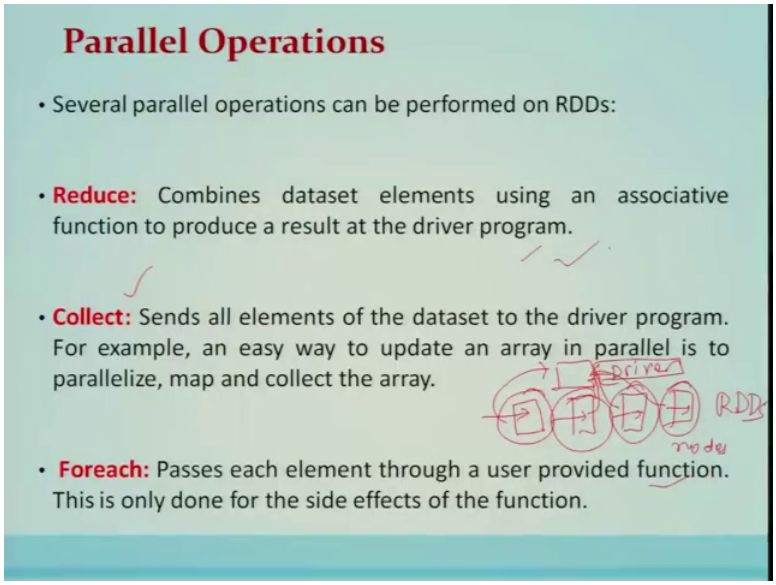
The cache action leaves the data set lazy, but hints that it should be kept in the memory after the first time it is computed, because it is going to be reused. Save actions evaluates the dataset and write it to the distributed file system.

(Refer Slide Time: 14:26)



If there is not enough memory in the cluster to cache all the partitions of a data set, spark will recomputed them when they are used.

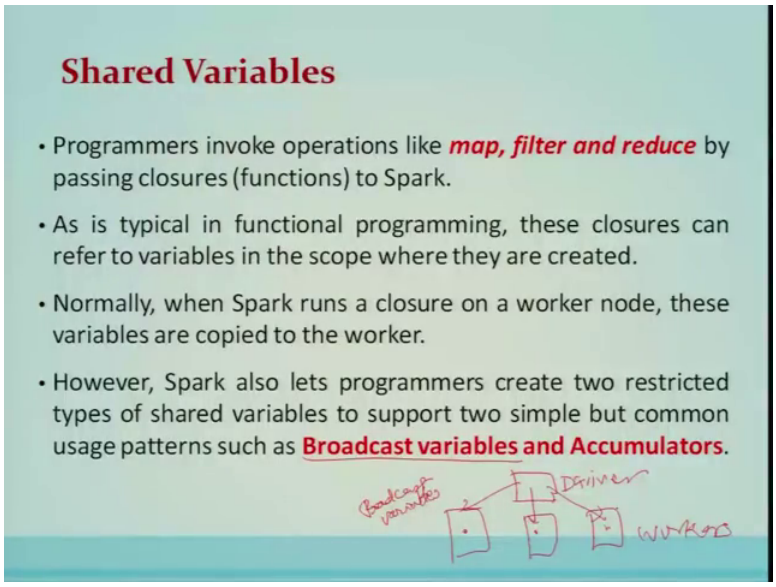
(Refer Slide Time: 14:33)



Now, parallel operations there are several parallel operation that can be performed on RDDs; one is called reduce that will combine the datasets, data set elements using an associative function to produce the result at the driver program. Collect, sends all the elements, of the dataset to the driver program, for each passes each element through a user provided function. This is only done for the side effects of the function.

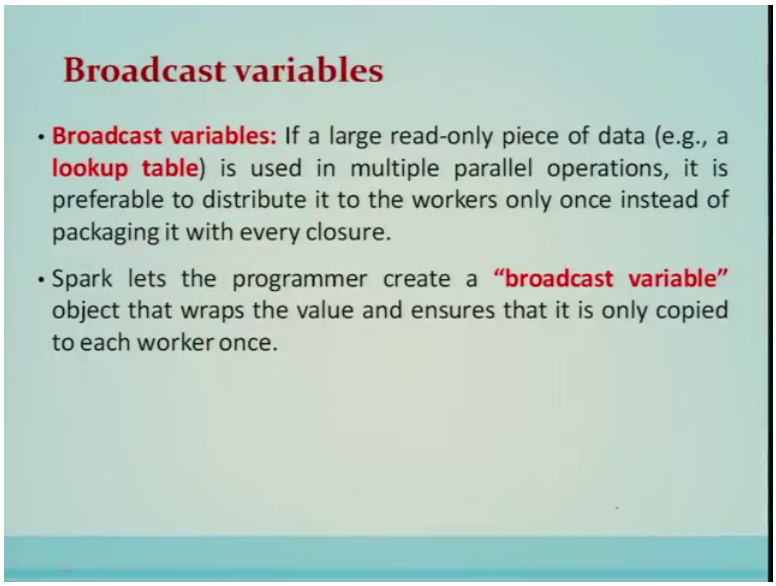
So, if you see that this is the driver program, and these are the RDDs. So, here as far as the parallel operations are concerned, when it is said as collect; all the RDDs, all the nodes, which are having an RDD, which will get an result, they will be sending the values back to the to the driver program, and for each means here at each RDD a, user defined function will be invoked and transformation will happen, and reduce also will perform the transformation on RDDs.

(Refer Slide Time: 16:03)



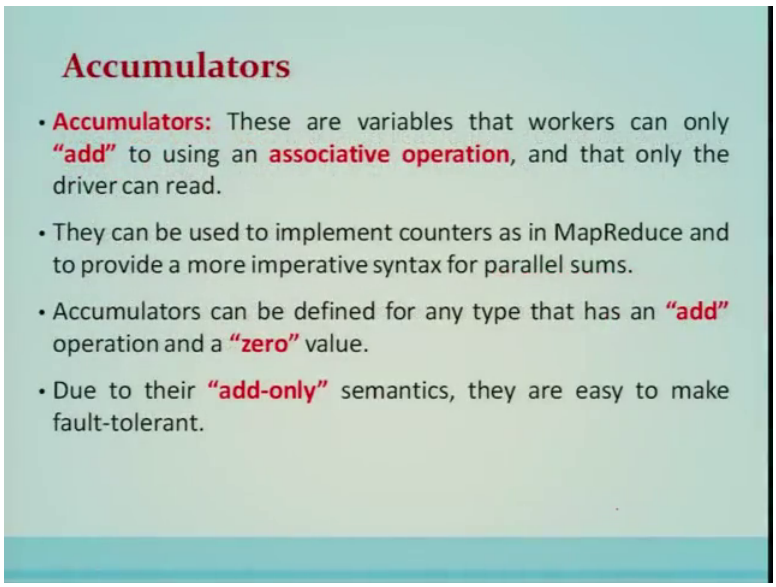
Now, these are the shared variables. Programmers invoke the operations like map filter and reduce by passing the closure functions to the spark. The spark lets the programmer create two restricted types of shared variable to support two simple, but common usage patterns; such as broadcast variables and accumulators. So, broadcast variables, if very small piece of data which is to be communicated to all the workers, then the broadcast variables are used. For example, if it is the driver and these are the workers. So, the broadcast means that the driver will communicate through the messages, through the broadcast variables, to all the workers. So, this way these share variables they are going to be utilized.

(Refer Slide Time: 17:09)



So, broadcast variable if a large read only piece of data, is used in a multiple operations, it is preferably to distributed to the workers, only once instead of packaging it with every closure. Spark lets the programmer create a broadcast variable object that wraps the value and ensure that it is only copy to each worker once that I have explained in the previous slides.

(Refer Slide Time: 17:30)

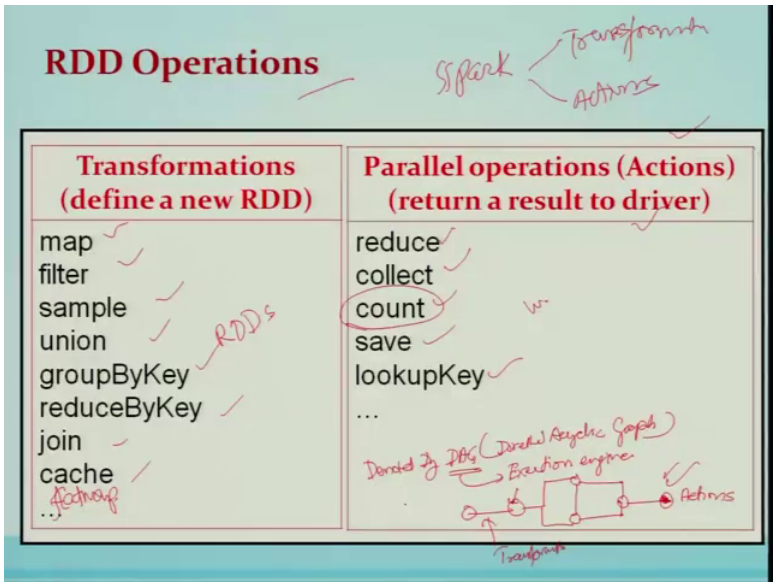


Accumulator: These are the variables that the worker can only add to using an associative operation and that only the driver can read. So, they can be used to

implement counter; such as Map Reduce, and to provide a more imperative syntax for parallel sums.

Accumulators can be defined for any type that has an add operations and a zero values. Due to their add only semantics, they are easy to make the fault tolerant.

(Refer Slide Time: 18:02)



RDD operations: So, I told you that RDDs spark provides the operation in the form of transformations, operation in the form of actions.

The plan of transformation in action is denoted by a dag; that is directed acyclic graph, and this particular dag is build in the execution engine. For example, from the start, if this kind of dag is being produced here, the edges are called the transformations, and here the nodes are called basically the actions.

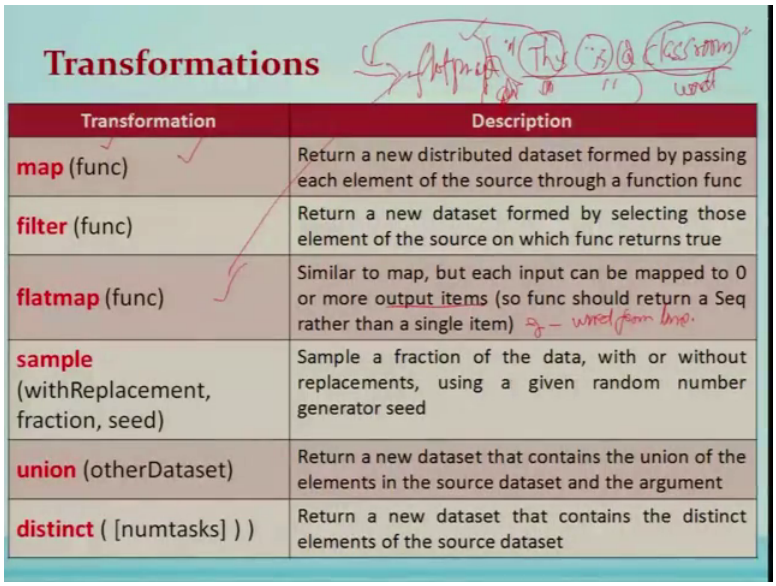
. So, transformations from one RDDs to another RDDs. So, transformation will be applied on the RDDs. For example, the map will take RDD, and do the transformation filter sample union group by key reduce join and cache. They will be performed on RDDs. And it will transform from one form to another form, as we have seen in a flatmap earlier. So, flat map is also there.

Now another kind of operations is called basically the action and this action whenever an action is encountered in a dag, then only the execution will takes place, and the values are returned back to the driver program. So, this is very important that actions have to be

fired. For example, actions is reduced collect count sale and lookup key. So, when count in the sense, the word count program you have seen in the Map Reduce, is being supported here directly by the spark.

So, the word count or a count will. So, once an action count happens. So, automatically the dag, which basically has already transformed into this kind of graph, will be executed, and the value will be returned back.

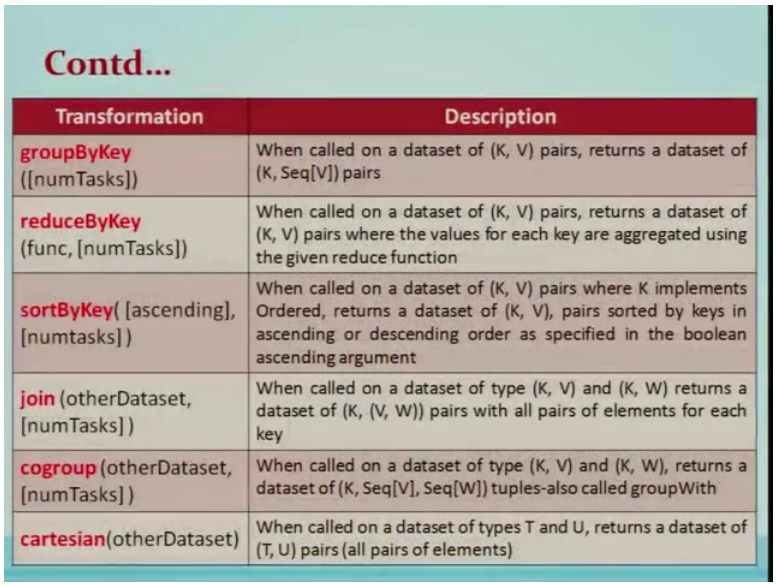
(Refer Slide Time: 21:10)



So, the transformations. Let us see the flatmap, because we have already seen, is similar to the map, but each input can be mapped to zero or more output items. So, let me give you an example of flatmap. So, if the line is given, if a line is given and flatmap is asked to split on the blanks then the input is this particular line, but as far as the flatmap is concerned it will output all the words. These words will be the output as far as the flatmap is concerned.

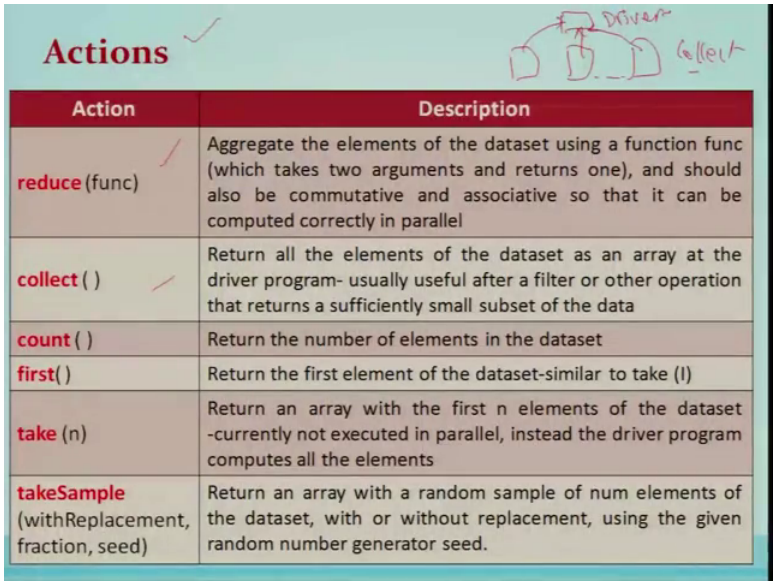
. So, the input is one, but output are basically a list of words will be output. So, similar to the map, but each input can be mapped to zero or more output items, that I explained you through an example. Similarly all other for example, map you know that returns a new distributed dataset, from by passing each element. So; that means, it has the function which will transform the RDD from a particular input to a particular output.

(Refer Slide Time: 22:55)



Similarly, other transformations are grouped by keys, reduced by keys, sort by key, join and so on.

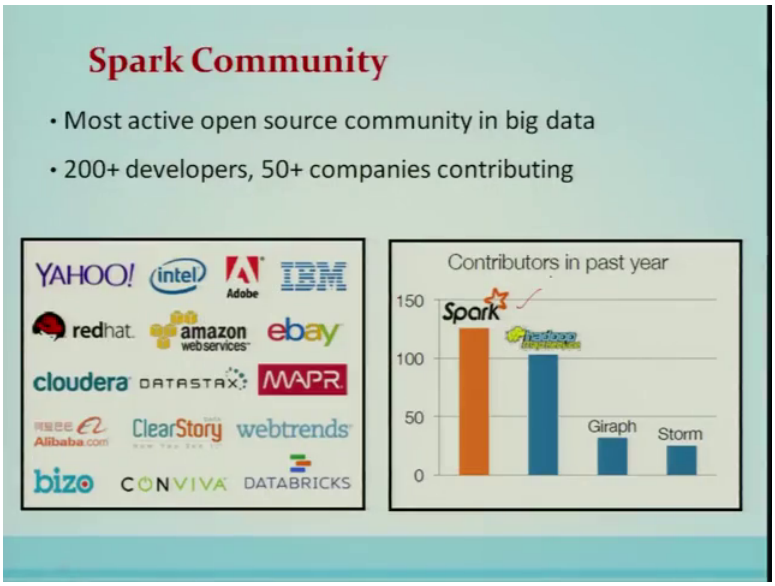
(Refer Slide Time: 23:03).



Actions: They are basically different from the transformation, as I told you actions are very important, because they will trigger the entire operations, which is build the form of the dag, and the spark will return the value to the driver, and driver in turn will get the values, and completes the entire function.

So, reduce function and aggregate element of the dataset. Collect means it will return all the values of the data set as an array, at the driver program. For example, these are all the worker nodes, and this is the driver program. So, when the collect operation, collect action will be there, then all the values whatever is computed will be collected back. So, collect is an action. So, all the actions will perform on the application. So, that is why these actions are very important, and they are listed here in this particular slide.

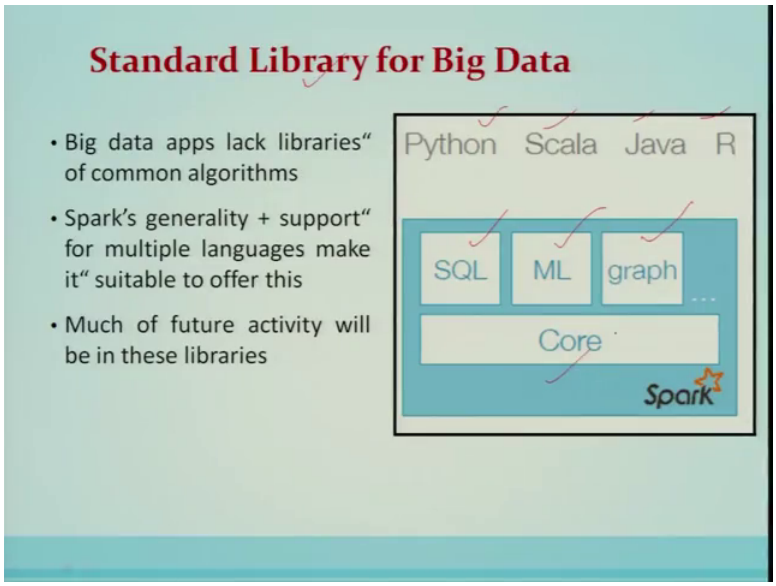
(Refer Slide Time: 24:17)



So, spark community is most active open source community in a big data, because it is going to be used in a big data scenario, because a very large dataset, it is able to compute this only platform, and it is also hundred times faster compared to the Hadoop map reduce.

So, just see that the contributions are growing in spark, built in libraries.

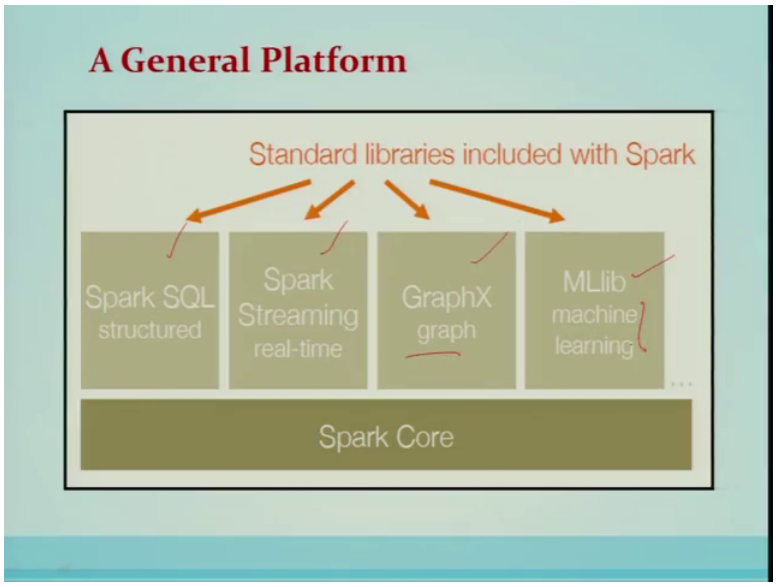
(Refer Slide Time: 24:42)



Standard libraries for big data is supported. Here big data applications, lack libraries for common algorithms sparks generality and support for multiple languages make it suitable to offer this, much of the future activity will be in the form of a library.

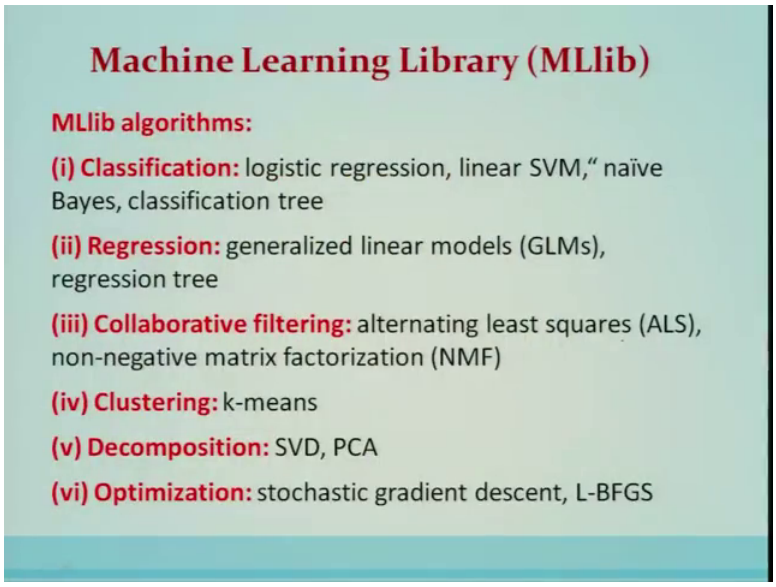
So, the languages which are supported here, are by the spark is python Scala Java R, and SQL machine learning and a graph are basically the standard library, which runs over the core spark.

(Refer Slide Time: 25:16)



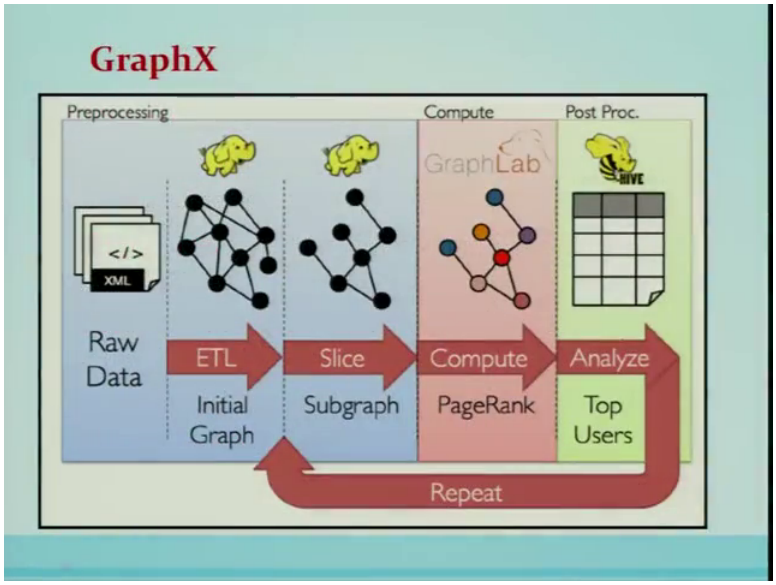
So, these are the standard libraries which are included with the spark, in the spark SQL spark streaming that is for the real time, graph X is for the graph applications, and MLlib is for the machine learning.

(Refer Slide Time: 25:31)



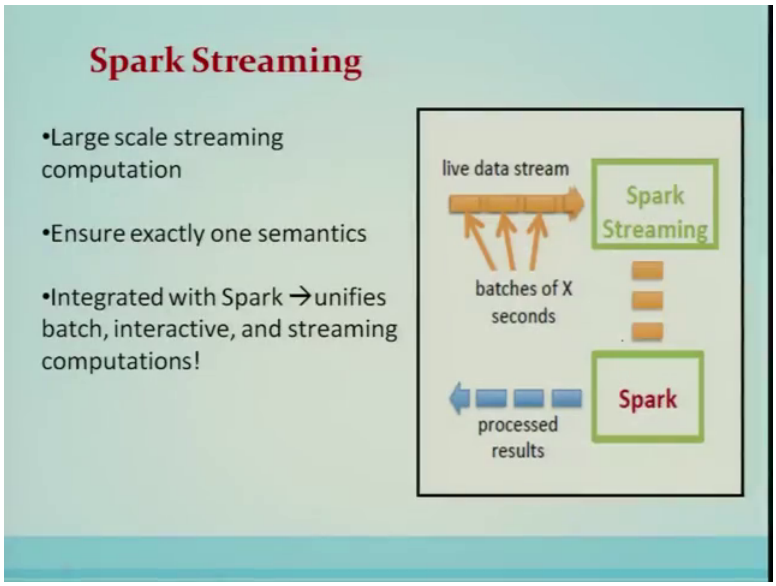
These are all written over here.

(Refer Slide Time: 25:36)



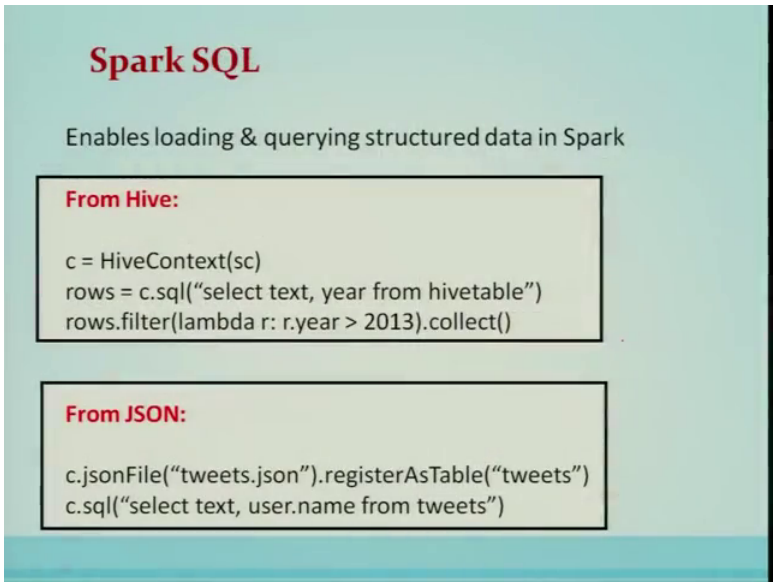
So, graph X is for the graph computations.

(Refer Slide Time: 25:39)



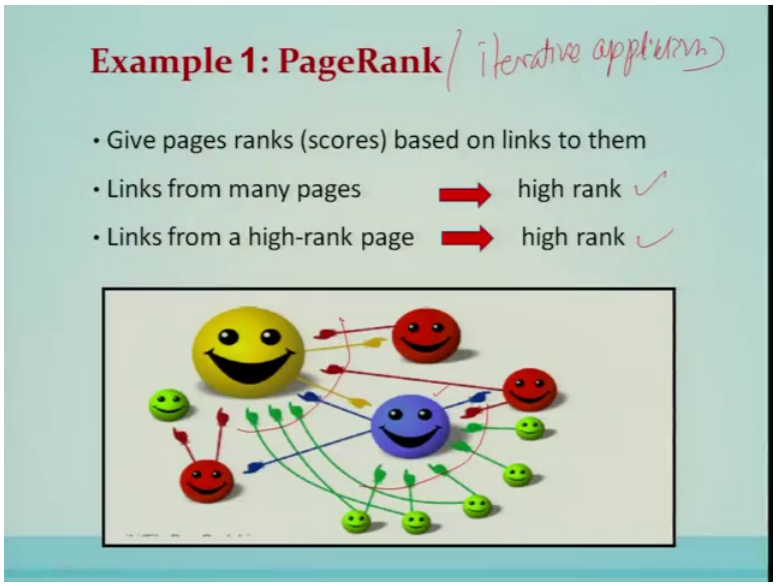
Spark streaming is for large scale streaming computations spark SQL.

(Refer Slide Time: 25:46)



For the structured data. Some examples of spark.

(Refer Slide Time: 25:52)

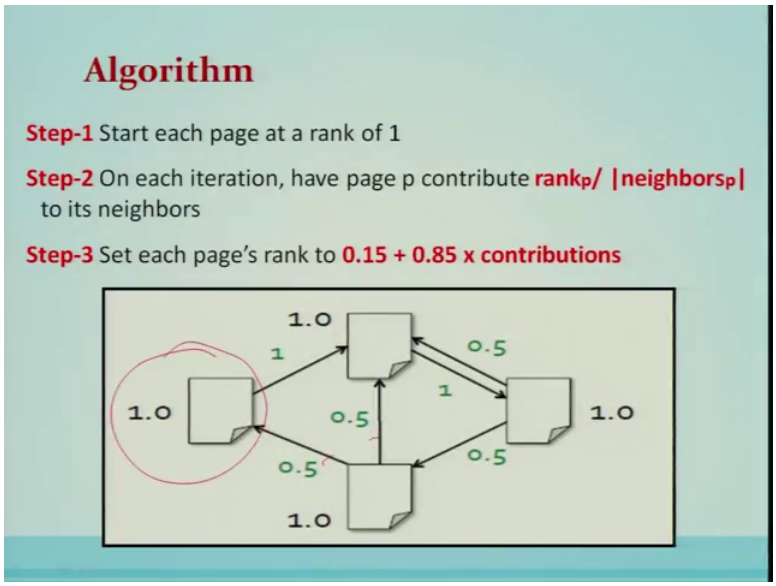


So, let us see the PageRank computation, which is to be done by the spark. As you know that PageRank computation is an iterative application and is heavily used in the Google, to find out the ranks of the web page or the ranks of the website.

So, it gives pages rank. Sorry score based on the links, which are basically pointing to that particular page. So, the links from many pages, will give a high rank, and links from a high rank page to a particular page also will give a high rank to a particular page.

So, here you can see that. So, many links are pointing towards a particular node, is going to be a high page, rank page or a website. Similarly here also now it will be ranking all the web pages, of those pages which are being leaned over here.

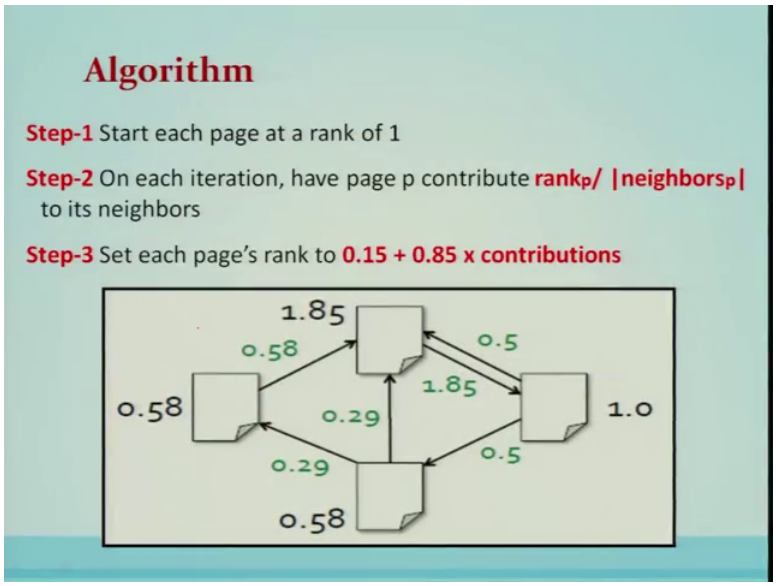
(Refer Slide Time: 27:08)



Let us see this particular algorithm. This algorithm works in the iterations. So, initially all the nodes are given the rank one, and then iteratively, it will compute the rank by iterative operation. So, let us see this, this particular node, is basically outgoing links are two. So, basically this is divided, 1 is divided 0.5 and this is also divided 0.5. Now the rank of this particular node has to be recomputed. So, how many links are basically indicating to this particular node, will be the new computation, which will be changing its link, as I told you this.

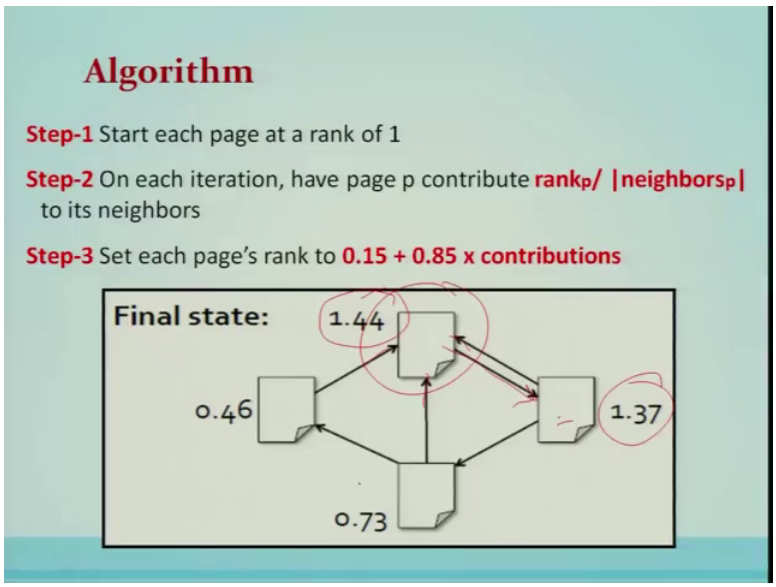
So, now, this particular PageRank of this particular node or for every other nodes, is now will change why, because the weights of these pages are going to have now changed in this iterations, and according to this formula it is going to change.

(Refer Slide Time: 28:12)



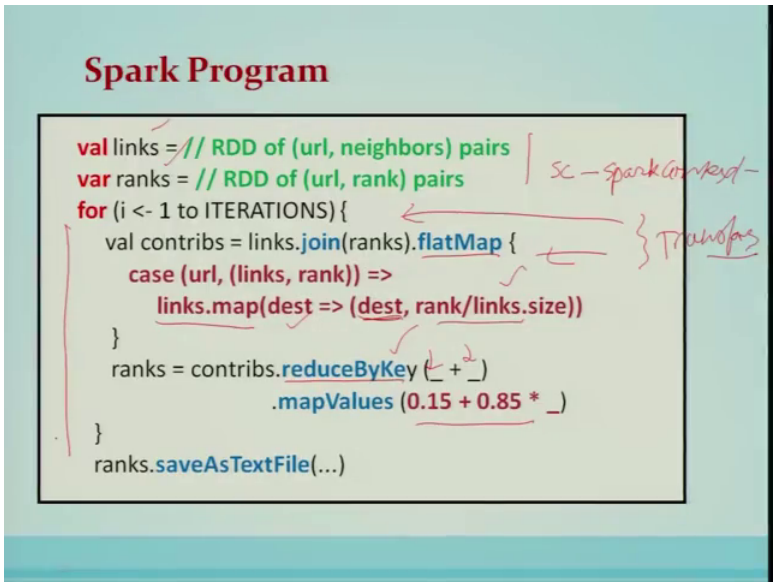
You just see that, this particular ranks are changed in the iteration number one. Similarly second iteration will again happen, and this basically will go on, and finally, here it stops.

(Refer Slide Time: 28:26)



So, here you can see that this particular node is having, the highest rank followed by this one. So, if it is the highest rank why, because you see that it has all the links, from all three nodes, and as far as this is the list because this highest rank node is now in pointing to this particular node. So, it is also receiving the higher compared to the other two nodes. So, this is basically the program.

(Refer Slide Time: 28:55)

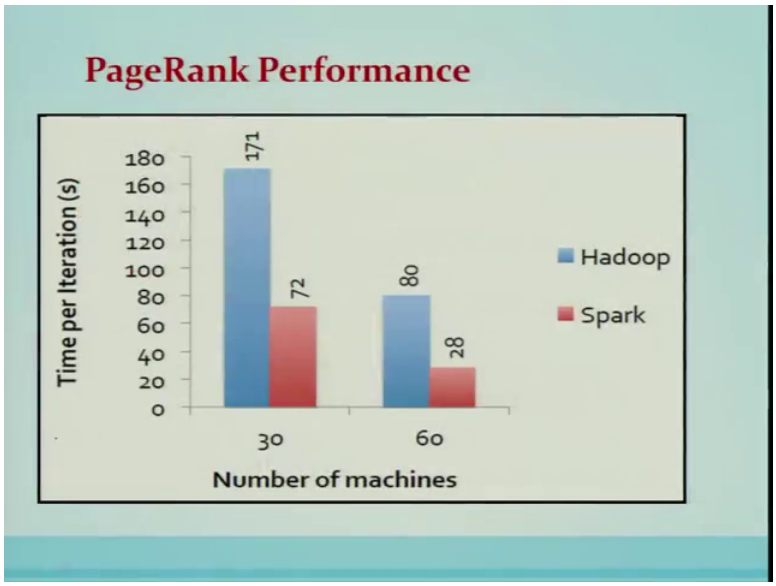


Let us go and see the program. Now here in the spark program for PageRank algorithm, first two steps if you see, it is nothing, but it is going to compute the spark and spark context; that is called SC, is going to build a spark context. And next iterations, you see it is a far loop, and this far loop will perform the transformations over RDDs. So, here you can see that. So, links and ranks, they will get the RDDs linked out of this URL neighbor, and URL rank pairs.

Then this particular transformation, flatmap will be performed, and will basically do the transformations on these particular links, and then it will be performed another transformation which are reduced by key, reduce by key in the sense, these particular rank or ranked by link size. These particular values are being given from the map transformations, these values are now being added; and that is called reduced by key. So, for the same page, all the values are going to be added, and these values will be now given as the rank.

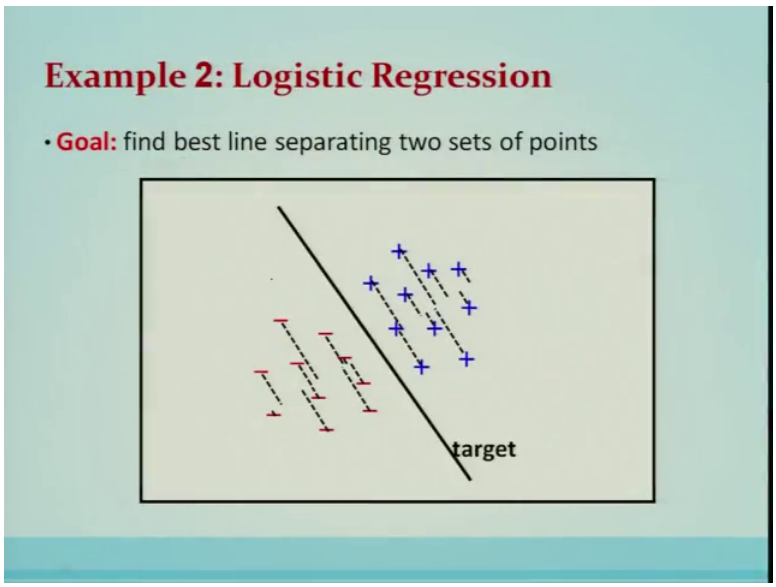
So, this particular program is written in the Scala, which will have two points; one is called transformations, most of the transformations are done on RDDs, and then it will take an action. And once an action is performed then the entire transformation will be executed, and the values will be returned. So, during these iterations the values, which are generated across the iterations, will be stored in memory.

(Refer Slide Time: 31:22)



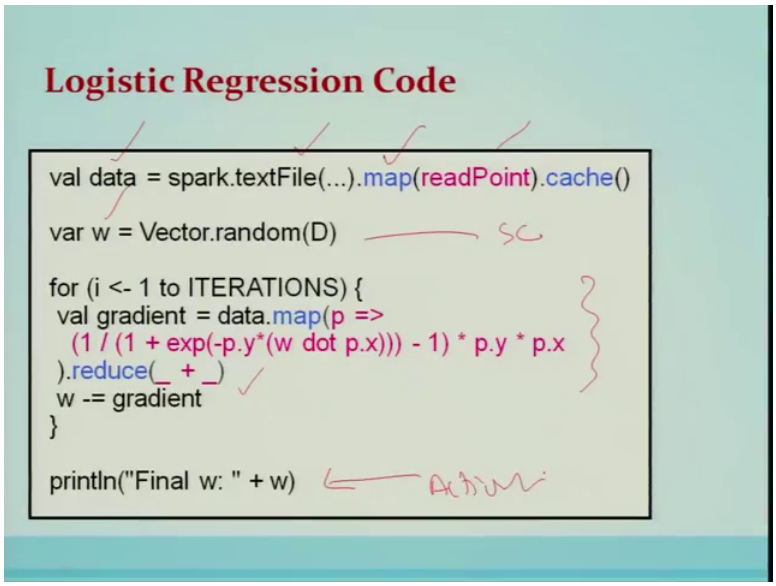
So obviously, this particular program becomes faster. So, you can see that this is the hundred times faster compared to the Hadoop version of the same program.

(Refer Slide Time: 31:33)



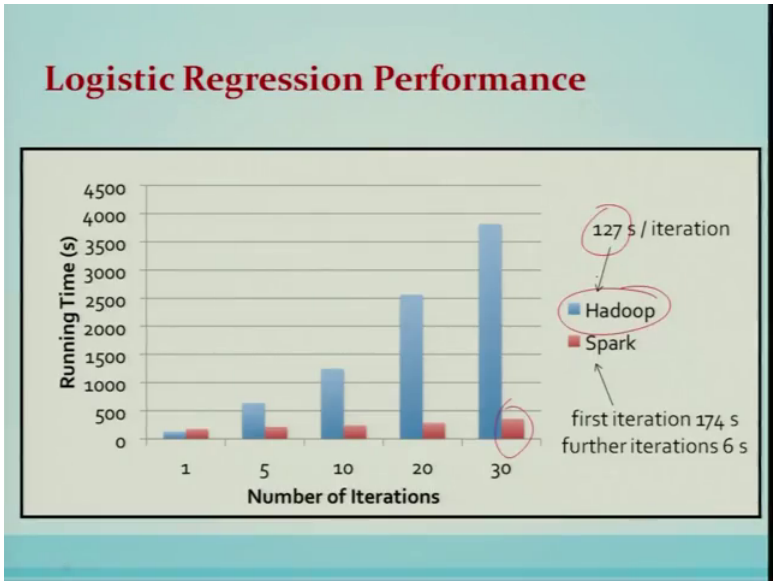
Now, another program machine learning is called logistic regression. This is also to be carried out in the number of iterations.

(Refer Slide Time: 31:42)



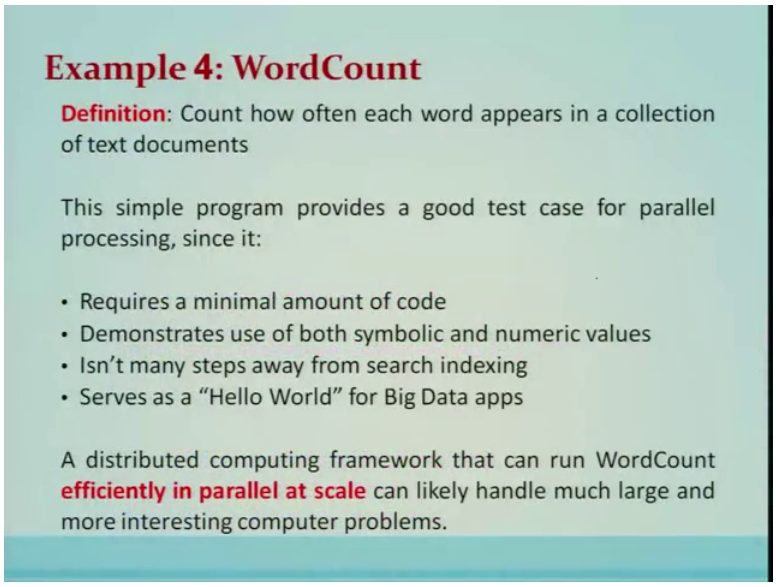
The code also you can see that, here this is, this map is the transformation, it will take a file perform a transformation, and it will generate a RDD which is called a data. Here also it will form a spark context, and then it will perform another transformations and up to reduce it will do, and finally, this is the action; that is this action will be performed in the local disk.

(Refer Slide Time: 32:17)



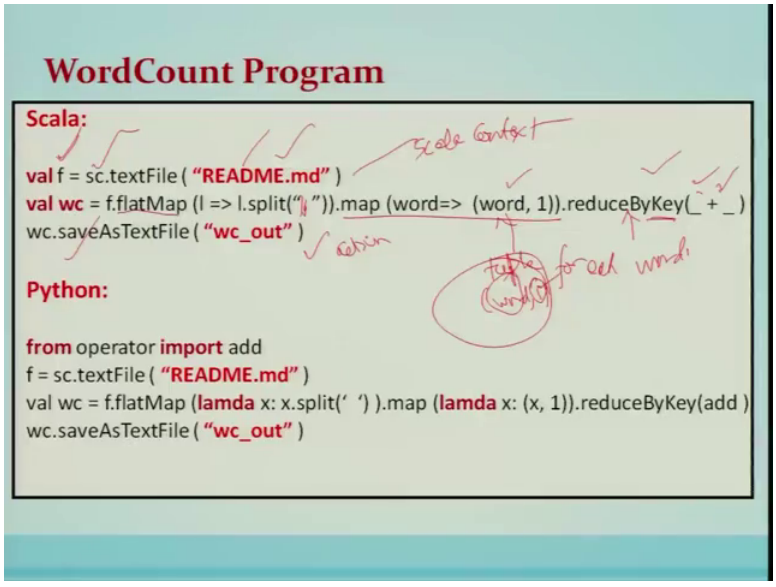
Why, because the result will be in the local form of the disk. So, here also if you see the running time, this is faster compared to the Hadoop version. Now another important program is called basically word count.

(Refer Slide Time: 32:40)



Word count is important why, because in the line the words which are most frequently occurring, is not going to impact in the page rank, as it is seen for example, the is and or, they are not going to contribute, significantly as far as the page rank is concerned of a particular website.

(Refer Slide Time: 33:05)

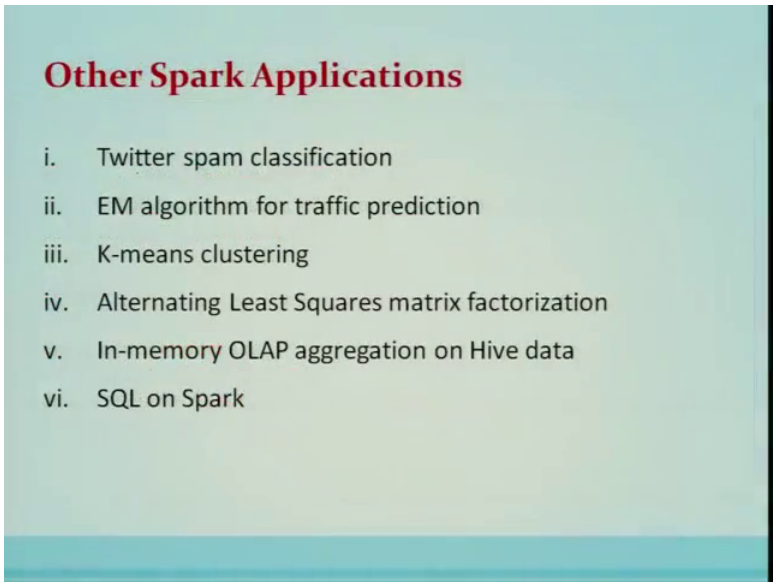


So, basically that is why the word count is the most important application, and it is also supported here in the spark.

So, just see as I told you that it will take a file, it will create a Scala context, it will read a file, and it will generate a file in a variable f, as far as the flatmap is concerned, it will take these lines which are read from the file, and it split according to the blank. So, basically it will generate the words, and these words. As far as the map transformation is concerned, it will output for every word comma one; that means, this particular tuple, will be output for each word, a tuple will be generated. Tuple means word comma one, and then reduce function will be performed reduced by key is a transformation on this particular RDD. The reduce will take the; that means, all the words having this particular value one, and they will be added up

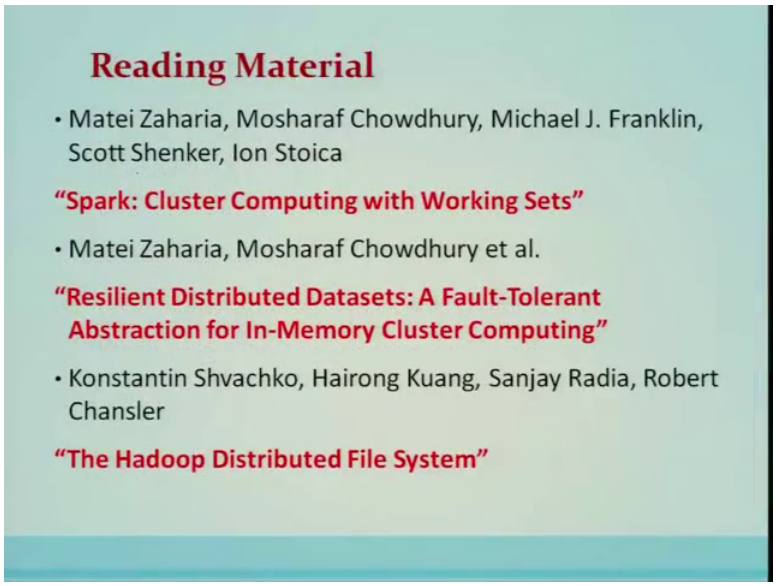
So, for a particular word t h e the, if it is appearing, many number of times it is appearing. So, that many number of ones will be added, and it will be reduced by key. So, these are all transformations. So, this particular word count, when it is saved this is an action, save is an action, then the entire program will be, the entire output will be saved and it will return.

(Refer Slide Time: 34:46)



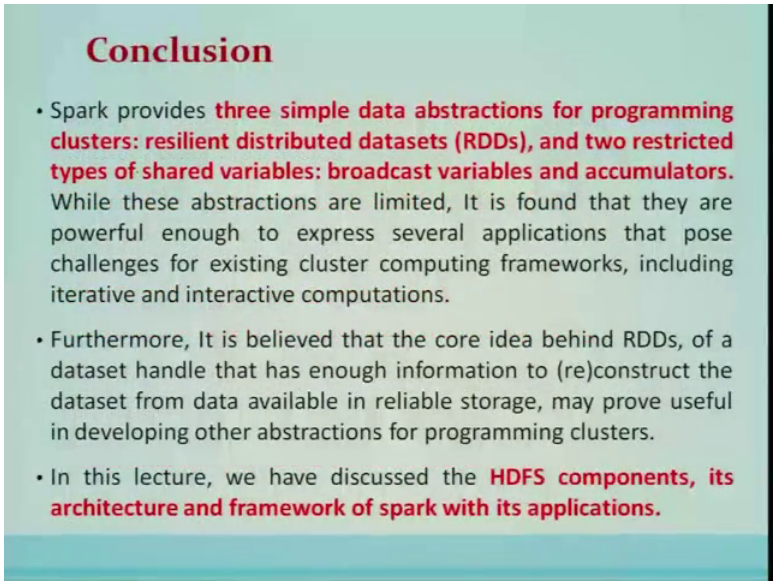
Other is spark programs and applications are there. For example, twitter spam classification and so on.

(Refer Slide Time: 34:56)



Various machine learning algorithms. Now lot of reading materials are available, that can be referred.

(Refer Slide Time: 35:06)



Conclusion: Spark provides three simple data abstraction for programming clusters, resilient distributed data sets, and two restricted type of shared variables; that is the broadcast and accumulators. Why these abstractions are limited? It is found that they are powerful enough to express several applications, including iterative and interactive computations. Furthermore, it is believed that core idea behind RDDs of a dataset,

handle that has enough information to reconstruct the dataset from data available in the reliable storage. So, in this lecture we have discussed the HDFS components, architecture, framework of spark and its application.

Thank you.