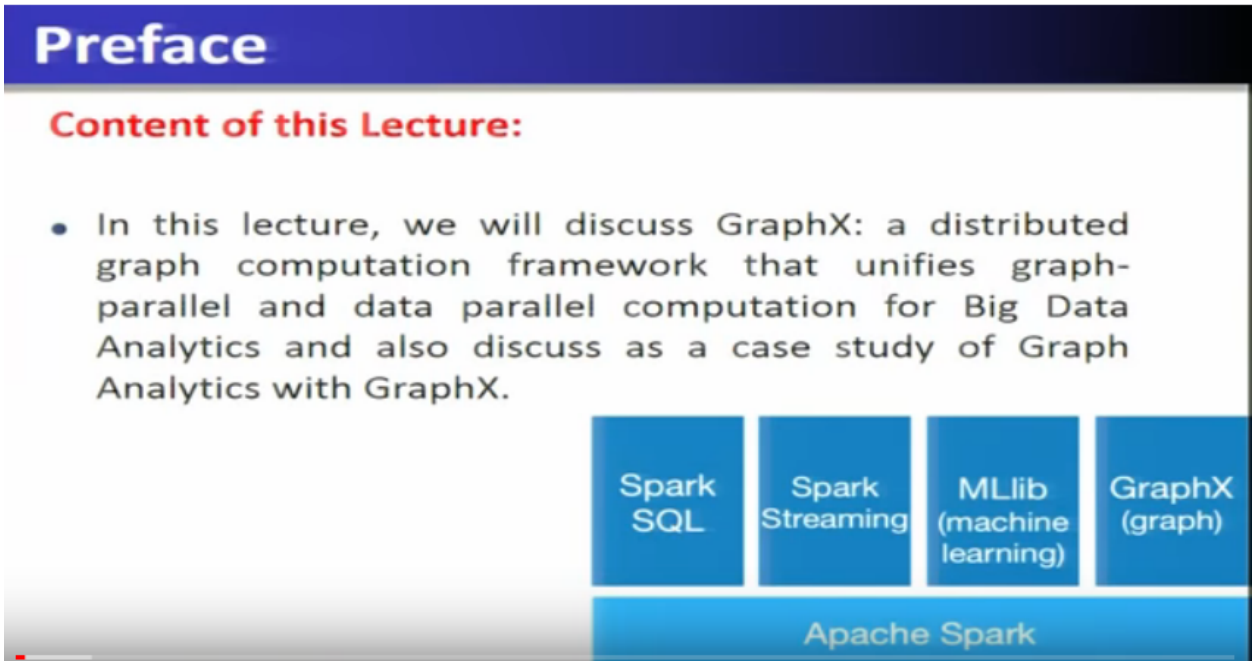
Lecture – 32
Spark GraphX & Graph Analytics (Part-I)

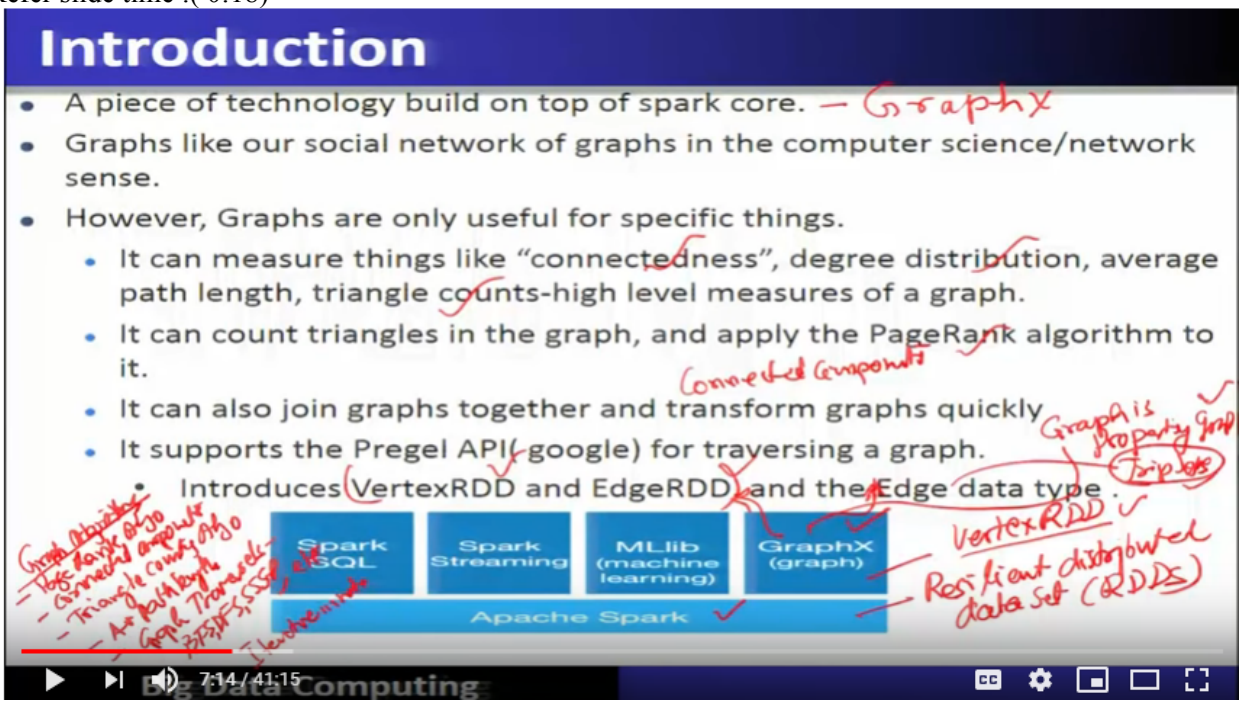
Spark GraphX and Graph Analytics.

Refer slide time :(0:18)



Preface; content of this lecture; in this lecture, we will discuss GraphX or distributed graph, computation framework. That unifies graph, parallel and data parallel computation, for big data analytics. I had also discuss in this lecture, a case study of a graph analytics, with GraphX here, on the bottom. Right? We have shown the position of GraphX, which is above the spark, apache spark, core. We are going to discuss, this component that is the GraphX, which is a part of core spark in this discussion.

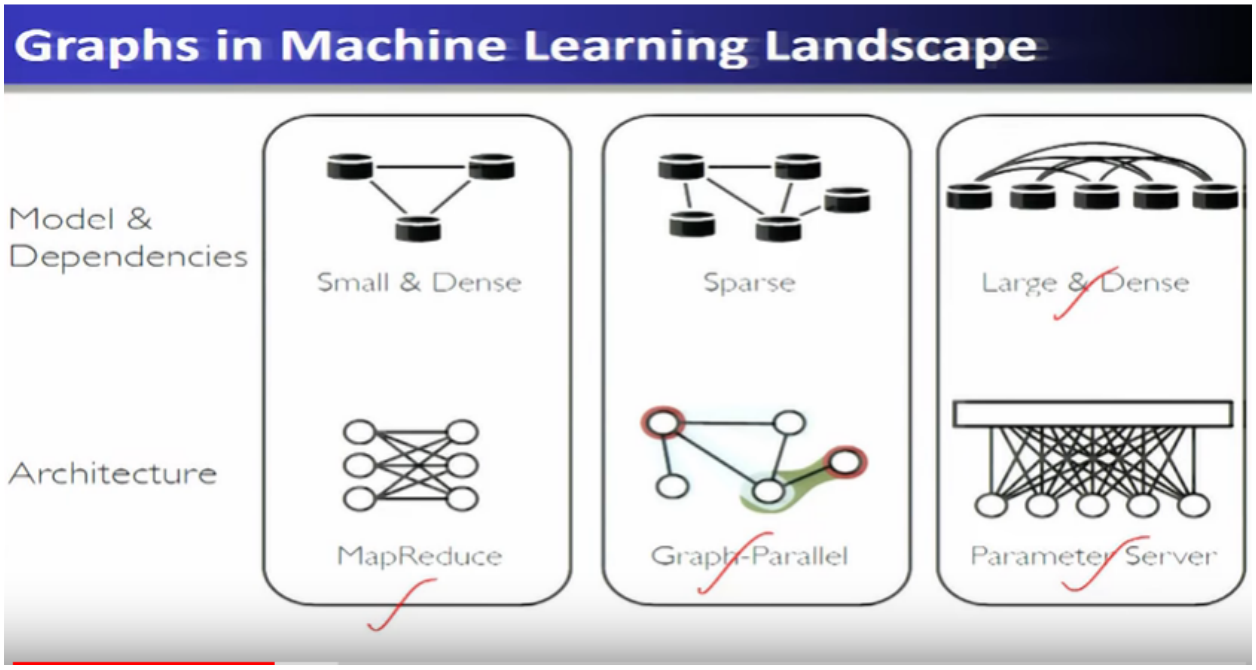
Refer slide time :(0:18)



Introduction; so, a piece of technology, which is, built on top of the spark core, we are going to discuss that is called,' GraphX', the graphs are like, our social network graphs, which are very well-known in computer science and in the network, complex networks, however the graphs are only useful for specific things, for example it can measure the things like connectedness, within a particular domain, the degree distribution, average path length, triangle counts, high level measures, of a graph in the social network graphs. The graphs can also count triangles in a graph, which basically will tell about, the cohesiveness, properties exist within a particular scenario. And also it can apply; we can apply the PageRank algorithm, to the graph to find out the ranking of the nodes. PageRank algorithm is also well known algorithm, which is by the Google to find out the most relevant pages, web pages which a user wants at a particular instance of time. So, for any search engine this PageRank algorithm is a well known, algorithm and is going is I will used in the web graph, it can also, join the graph, together and transforms, the graph quickly we will see this particular aspect of the transformation and joining of a graph together, to solve various problems, in the further slides. It can also support the pregel API that is given by the Google, for traversing a graph.

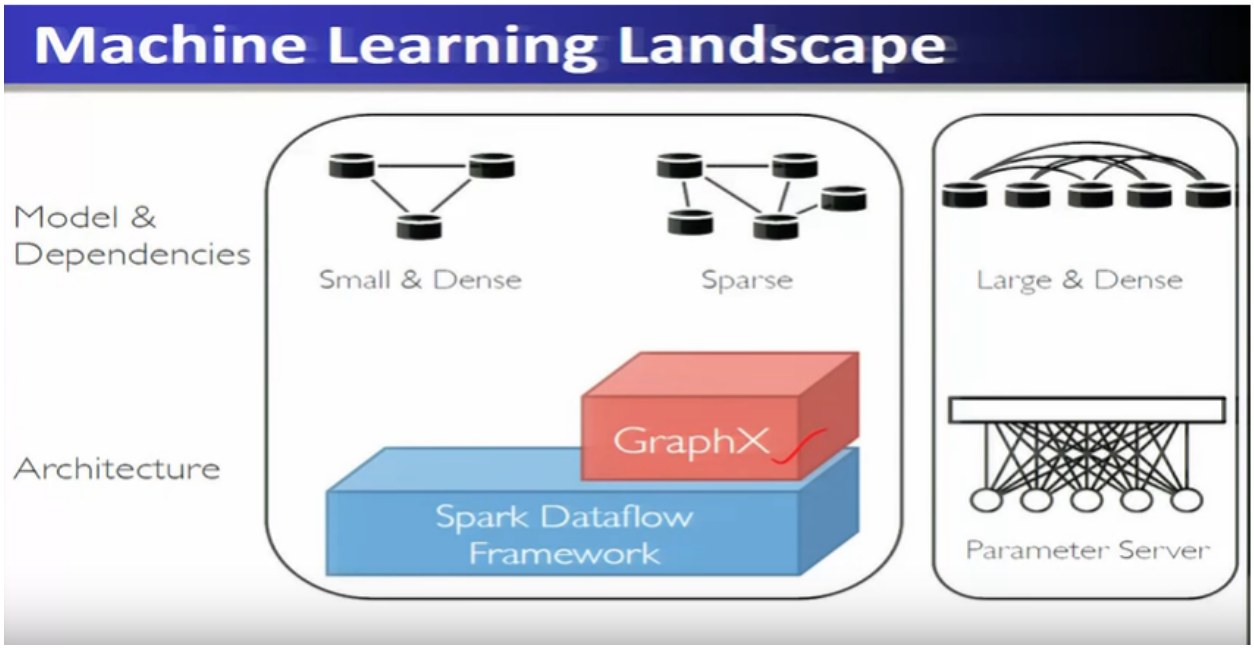
Now people also introduce here, about how the vertices and edges of a graph, what are the different data models, which are supported in the GraphX, which runs on top of the spark core. So, as you know that spark supports, the data type which is called a,' Resilient Distributed Data Set'. That is called RDD's, now this particular RDD's will use the vertex RDD. So, that means the vertices of a graph are represented as resilient distributed, vertex data sets that is called,' Vertices Vertex RDD Similarly', the edges of a graph are represented as the edge RDD that is the edge component, of a graph, which is represented as the edge resilient distributed, as data set. Together these particular components, the virtus vertex RDD and edge RDD's, which are supported, to construct a graph that is called a,' Property Graph', in GraphX. So, graph is called,' Property Graph'. Which is nothing but the vertex RDD and edge RDD, which is supported as a graph, in the GraphX, also there is another viewpoint, where in vertex and edges are combined together that is called a,' Triplet'. The concept of triplet is very useful, in building these graph algorithms, which we have discussed like PageRank algorithm connectedness and finding out, the triangle counts degree distribution and the connected components, which are some of the important graph algorithms, such as the PageRank, algorithm, then connected components, algorithm and then triangle, counts counting algorithm and degree distribution, average path length and for graph traversal, algorithms, which are going to be very much useful single, source shortest paths and so, on there are various graph algorithms. Which we are going to see and these graph algorithms, are iterative in nature, therefore, the support of the graph that is called a,' Property Graph'. And the triplets that we will, see how this GraphX will provide the important basic features and also the libraries of GraphX, which support all these algorithms, in an optimized manner.

Refer slide time :(07:18)



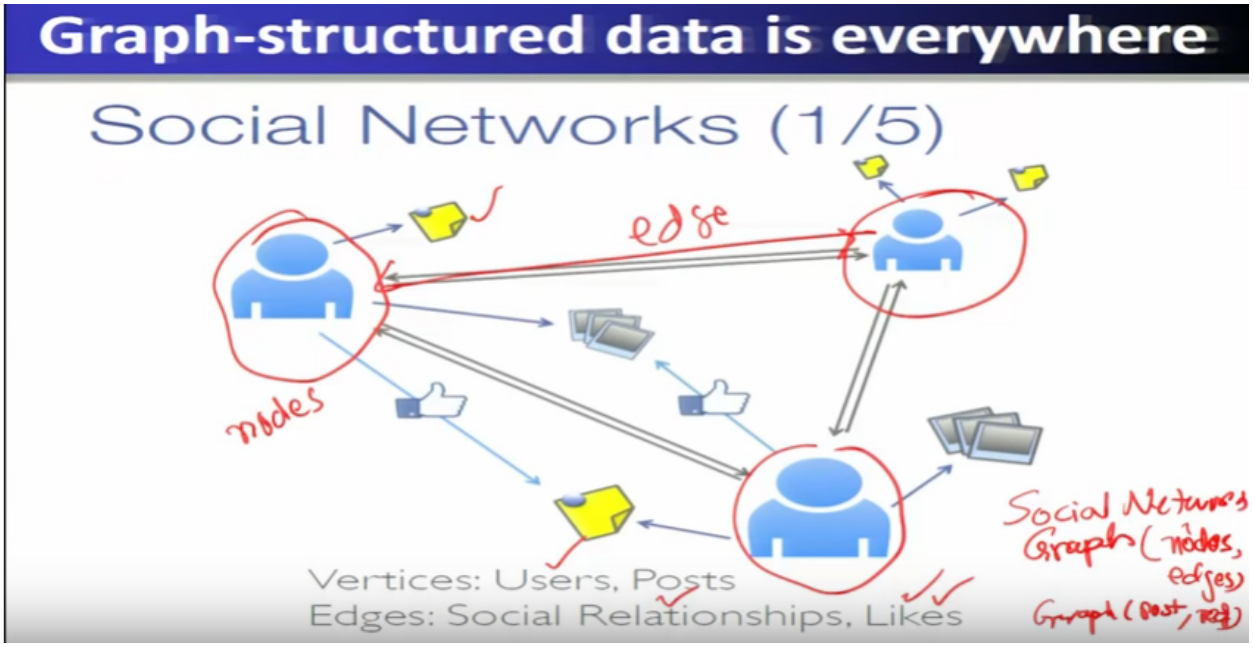
Now graphs are also used in the machine learning, landscape, for example, the models and dependencies, of the data, which can be represented in the form of a graph, whether it can be a small and or a dense graph or it's a sparse graph. So, that means once, a data is represented in a form of a graph, then machine learning, can be applied on this particular graph as a data, these particular aspects we will discuss more details, in this part of the course. So, what do we mean, by the parameter server that means, when we going to deal with, the programming of the, the machine learning using graph, we will basically encounter, with the large and dense graph, which has to be dealt with the parameter servers, similarly the operations, which are there in the machine learning, algorithms, which are implemented on top of the data sets, which are represented as a graph, has to be done in a graph parallel and data parallel operations that we will see, how we are going to support these, architectures. Similarly another architecture, which is going to be supported, for the big data landscape and which is there in the machine learning landscape, is about the MapReduce, application on small and dense scenarios, of a Big Data.

Refer slide time :(08:50)



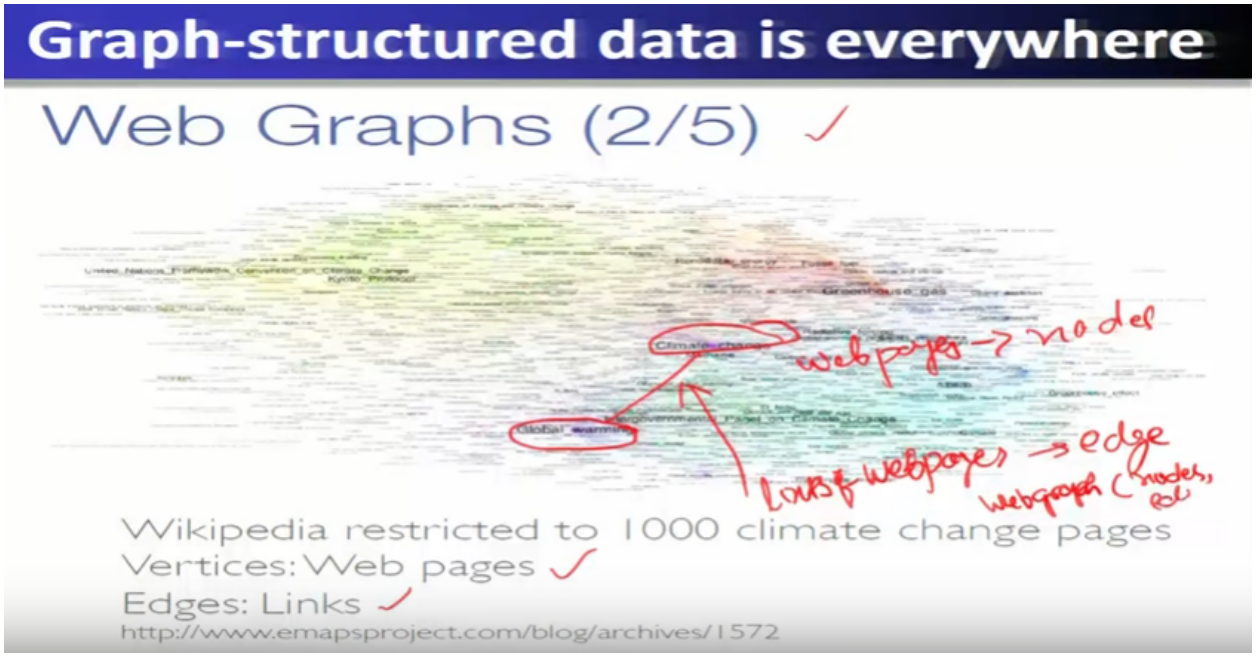
So, the basic architecture, which we will discuss here is the, the GraphX, how these GraphX, will be supported as the framework, which can also build the machine learning landscape, similarly all these part, we will discuss in this part of the lecture. So, graphs are there everywhere.

Refer slide time :(09:18)



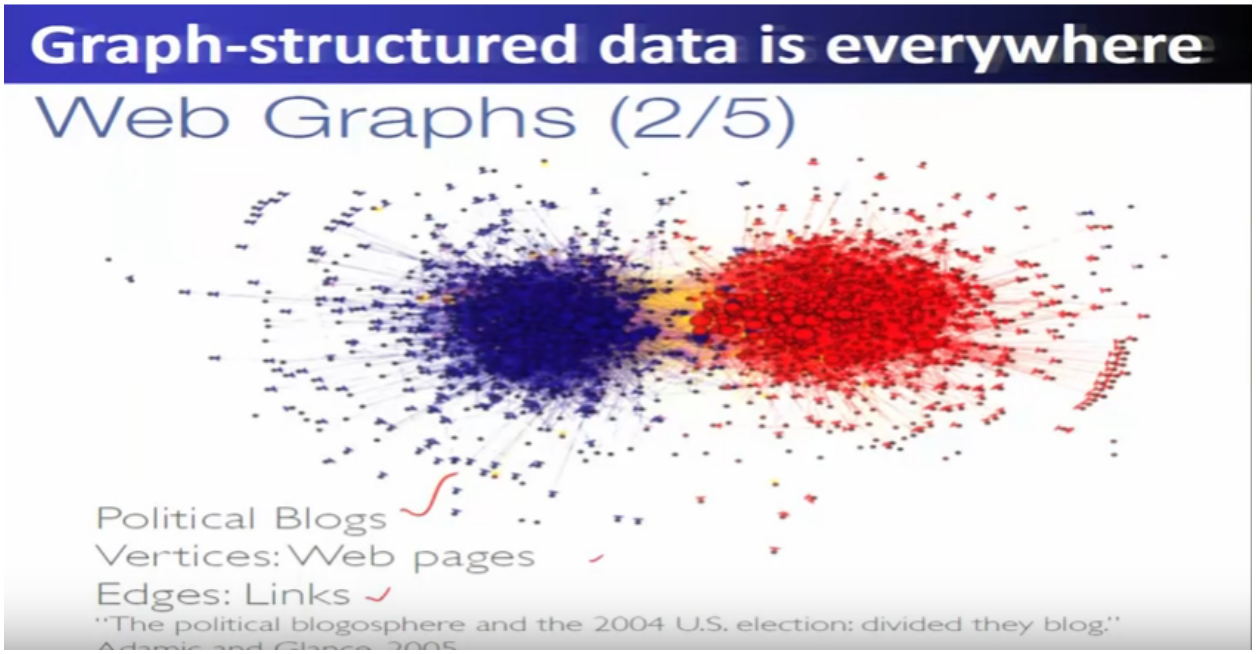
So, graphs are a structural data, which is there everywhere let us take some examples, where you can understand that the importance, of that graph, computation is very much required in today's workloads. So, for example social network, which is visualized as a social network graph, in this social network graph, the people are called as the,' Nodes'. So, these are the people, they are called the,' Nodes', of a graph and the relationship, between these people they, they represents the edge. So, hence, the graph comprises of the nodes and the edges, of a social network graph, where the nodes represents the people or the users and the social relationship, is represented by the edge, similarly the, the people who does, the post and they also, performs the like operation is also can be represented, in, in a form of a graph that represents, the post as the nodes and the likes, is basically the relationship, as an edge. So, the graph data or a social network data, you can represent, in a different form of graphs and further do the analysis, on top of it that we will discuss. So, graphs are a structured data and it's there everywhere.

Refer slide time :(11:10)



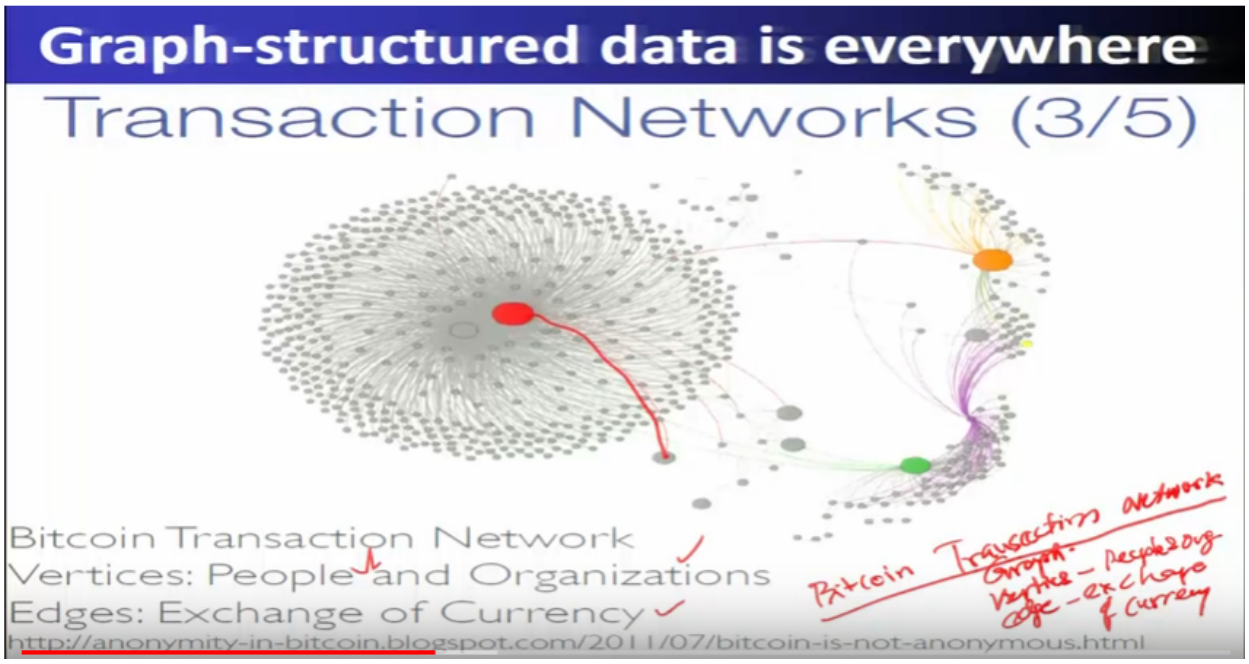
Another searching example, of the graphs is about the web graphs. So, for example, the Wikipedia has 1,000 different, climate change pages. So, if we represent the Wikipedia, as a graph as a web graph, then we can find out that all the web pages, will become the vertices and the links are among these web pages, they becomes the edges hence, this will form the web graph. So, we can see here, these are the web pages global warming and climate change and so, on they becomes the vertices and the connections between or the links between, these edges or the web pages, the links of the web pages, becomes an edge and these web pages, is are the nodes. So, this will form a web graph, which comprises of the nodes and edges.

Refer slide time :(12:24)



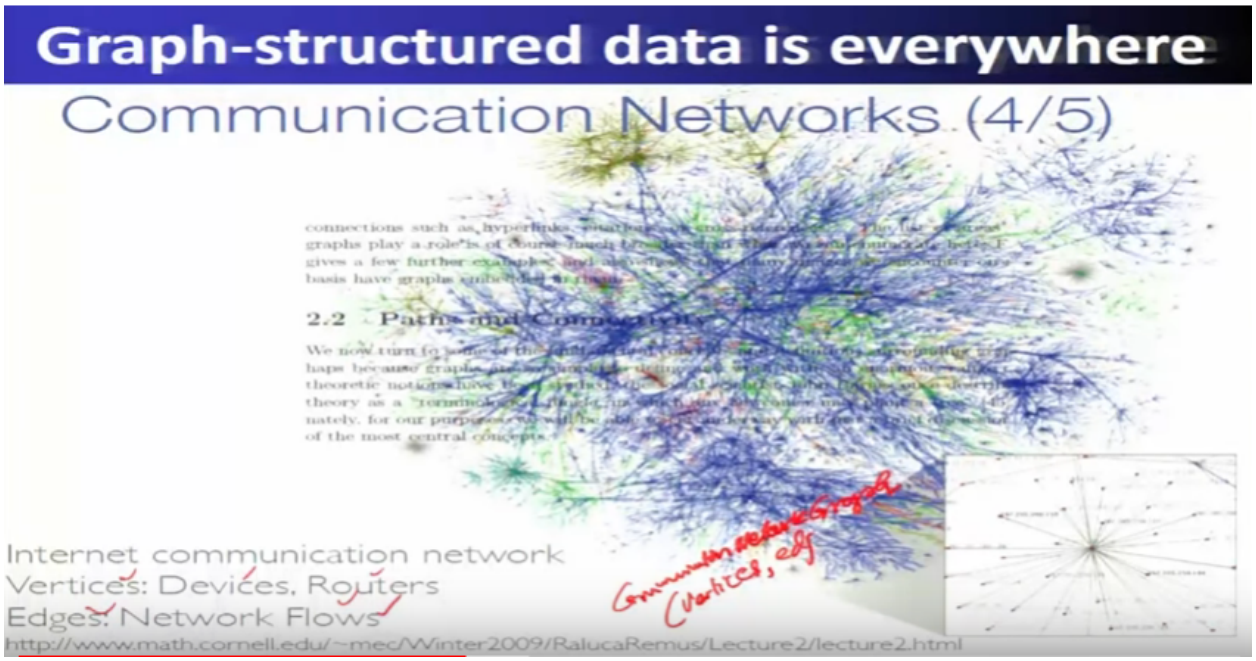
Similarly another form, of web graph is the political blogs data. So, that means the blogs, which represents the political bloc's is also represented in the form of a web graph, wherein the web pages are the vertices and the links are the edges. So, the graph which is formed out of, these web pages and the links are called as the,' Political Bloc Graphs', they are also the web graphs.

Refer slide time :(12:55)



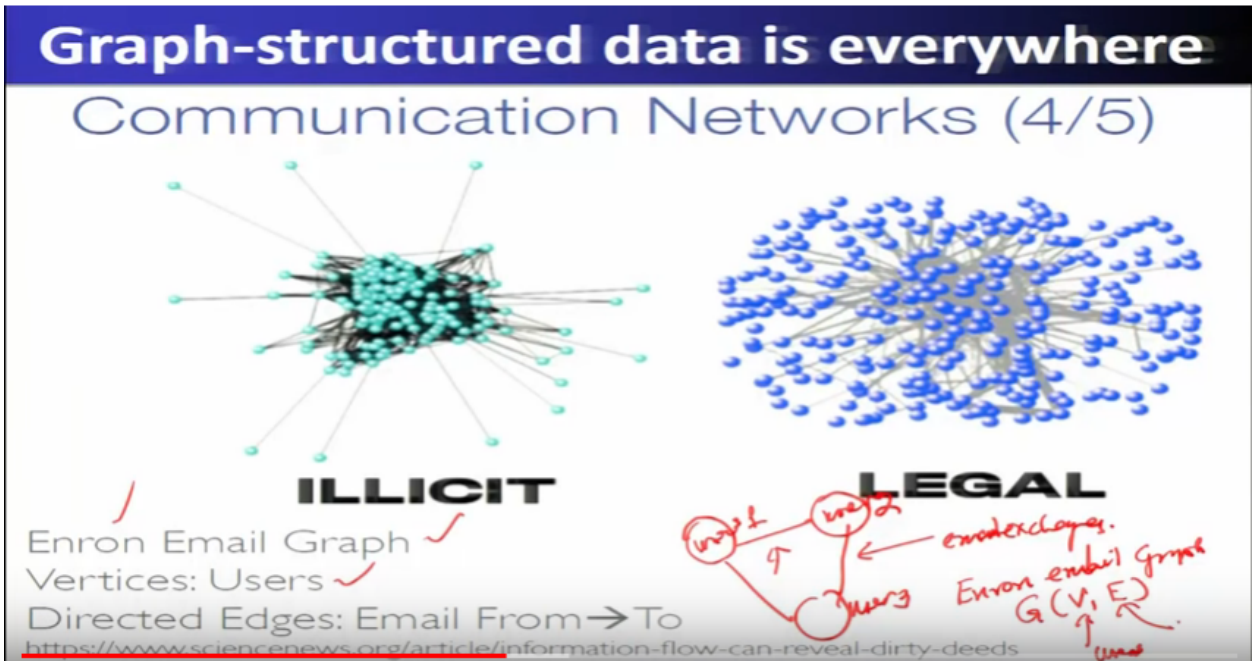
Similarly the Bitcoin transaction Network, also can be represented in a form of a transaction, graph or a transaction Network graph, we're in the vertices, represents the people and the organizations, which are executing the transactions, of the Bitcoin and the edges are representing, the interactions or exchange of currency, therefore this kind of interactions, between the people and exchange of currency, they can be captured in the form of a graph. And it's called the,' Transaction Network', Bitcoin transaction network graph.

Refer slide time :(14:10)



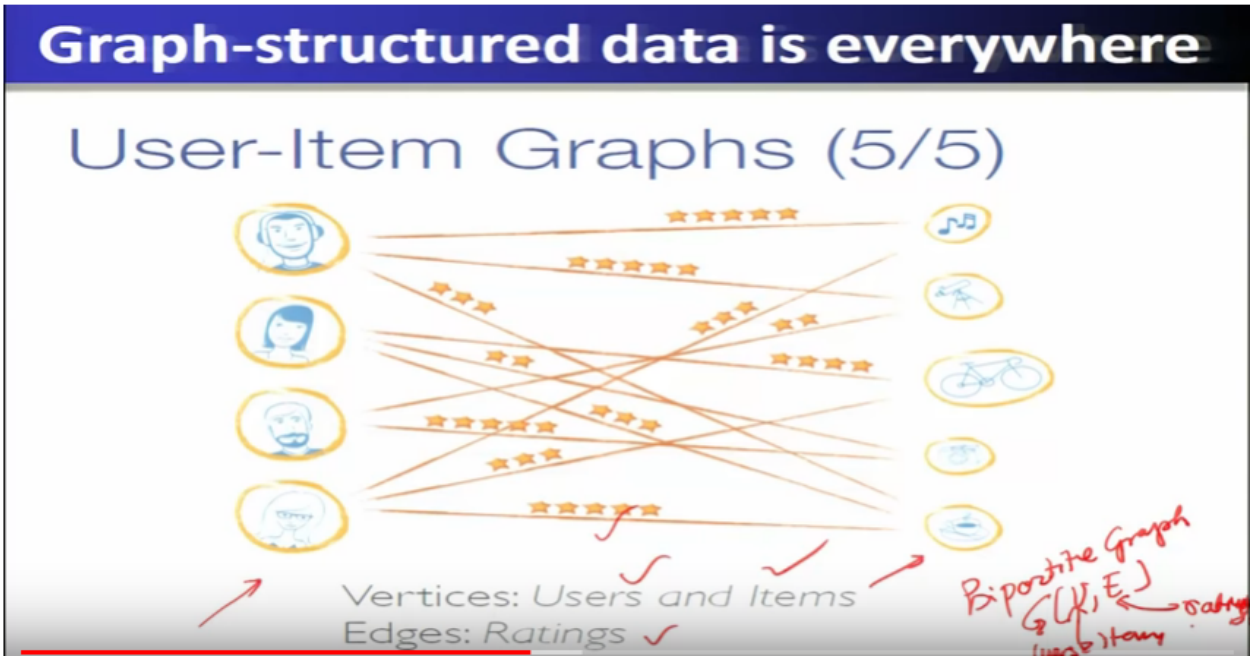
Similarly another kind of graph is, there in the internet communication network, we can also be represented in a form of a communication network graph, wherein the vertices are nothing, but the routers and internet devices. And the edges are the network, interactions or the flows of the data, between these devices is represented, as an edge therefore the communication, network can be modeled, in the form of a graph, communication network, graph, where the nodes are or the vertices are the routers and various internet devices. And the connections, between them that is also to be taken, as the network flows, is becoming an edge and this becomes a graph.

Refer slide time :(15:11)



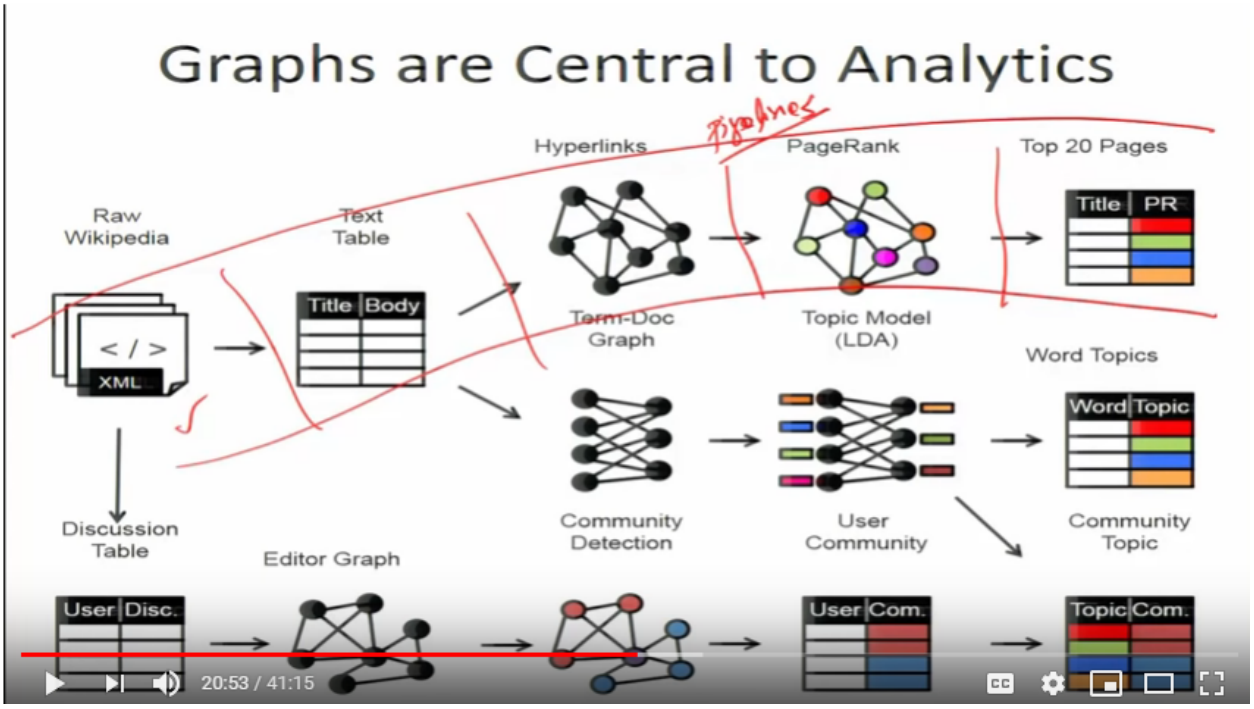
Similarly to detect the frauds among for the companies. So, for example Enron email graph can also be build up can be used. So, Enron was the company, which has committed fraud. So, their email exchanges, between different users is modeled in the form of a graph for, this kind of analysis. So, here the users will become the vertices and the email exchanges are basically the edges. So, this will form, the Enron email, email graph, wherein the vertices, when vertices are the users who have exchanged the mails. And these edges are the set of, users who have communicated, through the mail.

Refer slide time :(16:31)



Similarly the user item graph can also be visualized, as a bipartite graph, wherein the vertices are the users and items and their ratings that means the users will give the rating, to the items or the products that becomes an edge, with the weights given as the ratings. So, bipartite graph, user item graph, is a bipartite graph, where the vertices are the users and items, whereas the edges are the ratings, given by the user for different products, can be represented in a form of a graph.

Refer slide time :(17:24)

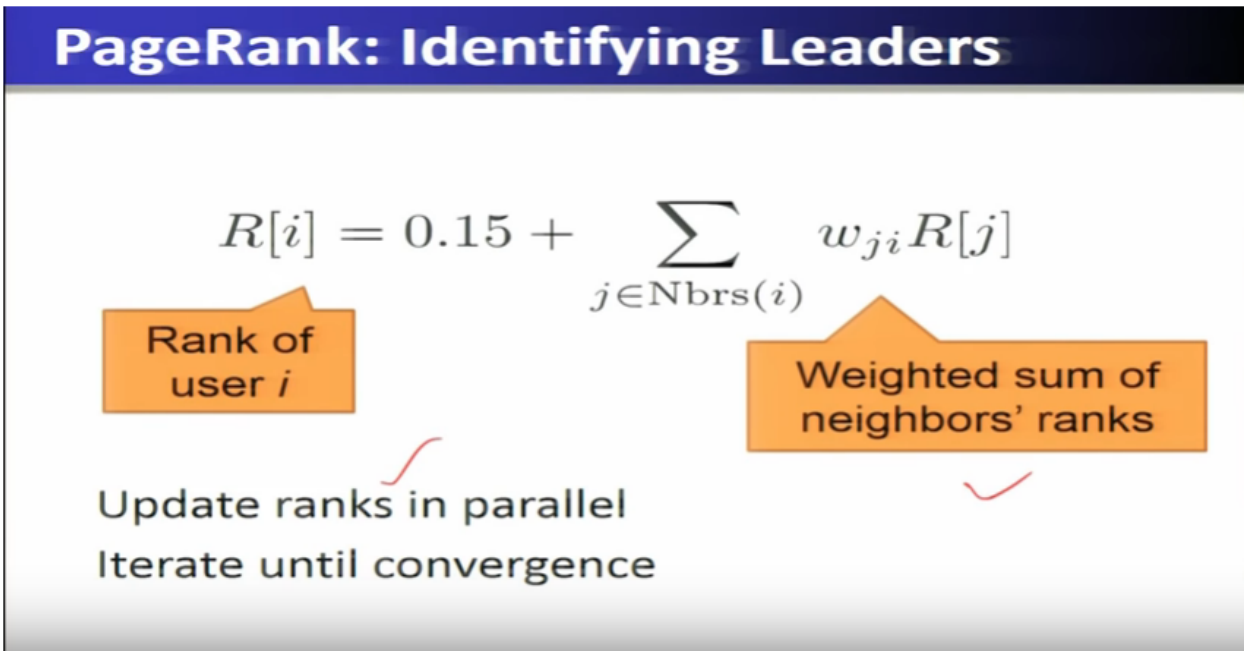


Now let us see some of the analytics, which can be performed, on the graph. So, graphs are central to the analytics also, let us consider the raw Wikipedia, purpose and using this Wikipedia, account corpus we can construct a table that is called a,' Text Table', which has the title and the body within it using this particular text table. We can construct and hyperlink graph, on hyperlink graph, we can run the PageRank algorithm. And after this PageRank algorithm, we can find out the top 20 different pages. So, their title and their page ranks are extracted in a form of it of a in a particular table. Now similarly from a, text table we can construct another graph, which is called a,' Term Document Graph'.

And this particular term document graph, we can apply topic model algorithm, I and from that we can extract, the word topic table that is word and topic table, on top of it similarly, from the raw Wikipedia again, we can construct another table, which is called a,' Discussion Table', wherein the users and discussion, topics are mentioned, from this we can generate an, editor graph and we can apply a community detection algorithm on top of it. So, we can identify the user community users and different communities, which are there in the Wikipedia text. Now we can combine both of them that is the topic model and the user community, together to get the topic community, information in the form of a table. So, these from Raw Wikipedia. We can get top 20 pages with the title and their page ranks, similarly from the Raw Wikipedia text; we can get the word and their topics, corresponding similarly from Raw

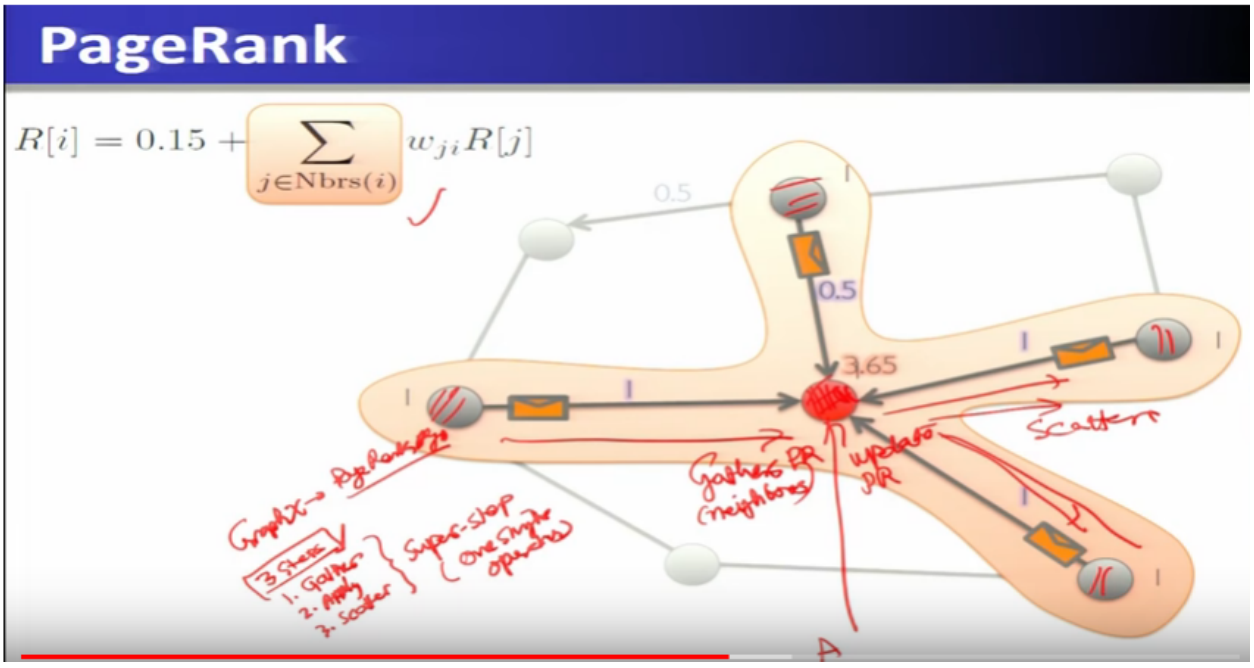
Wikipedia. We can get the topics and the different communities. So, this is basically, the start point of analytics and this flow or the sequence in, which we can get this particular, output or the analysts analysis, out of that text is basically called a,' Pipeline'. So, we will see here, in GraphX that not only it provides, the analytics, which can be performed on the raw text or in raw data. And these are the steps or the stages, to transform the data from Raw Wikipedia, into the output of that is top 20 pages and extract this insight, from the data is called the,' Analytics'. So, the graphs are also used, in the analytics and this support of making the pipeline, building the pipeline and performing, the analytics is also supported, in the GraphX that we will see in this discussion.

Refer slide time :(20:58)



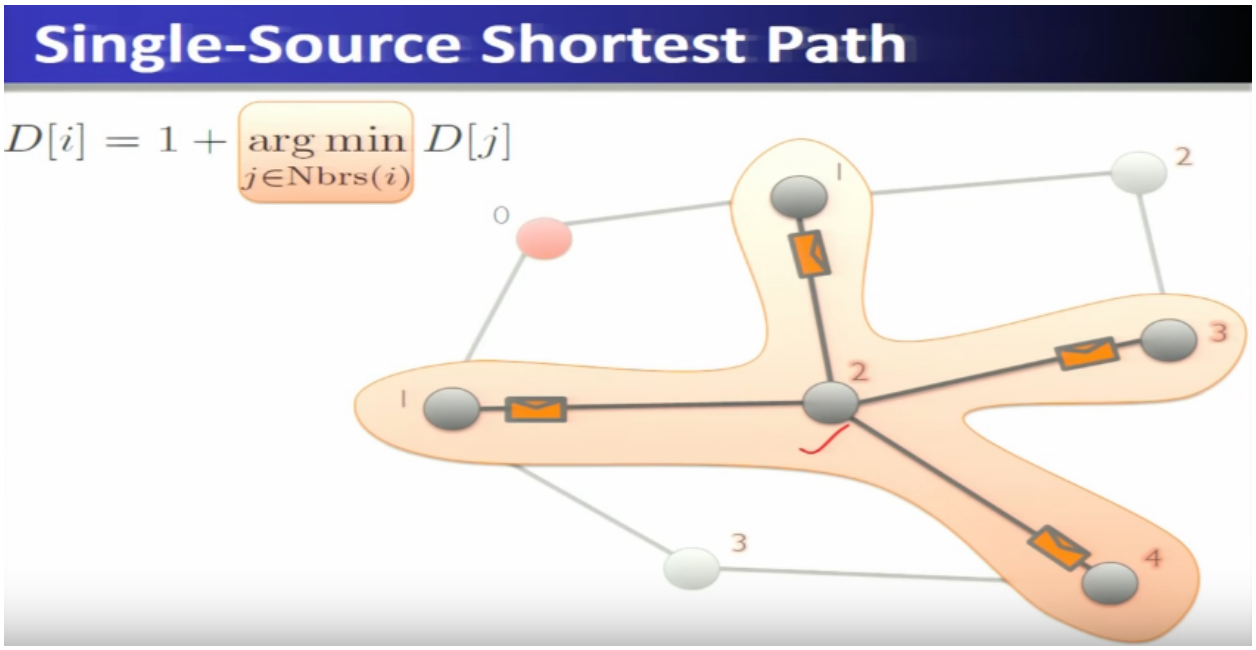
Now we will see the page rank, which is very central to the graph analytics. So, here we can understand that this page rank, algorithm is a parallel computation algorithm and it also performed, in different iterations that also, has to be done in a parallel, I and this iterations, will continue until it converges. And every iteration has to calculate, the weighted sum of neighbors rank and do the summation and find out, with a new page rank and this kind of page ranks, are updated this update of the page rank, continues over several iterations till it converges. And then only this page rank algorithm will finish its operations.

Refer slide time :(21:51)



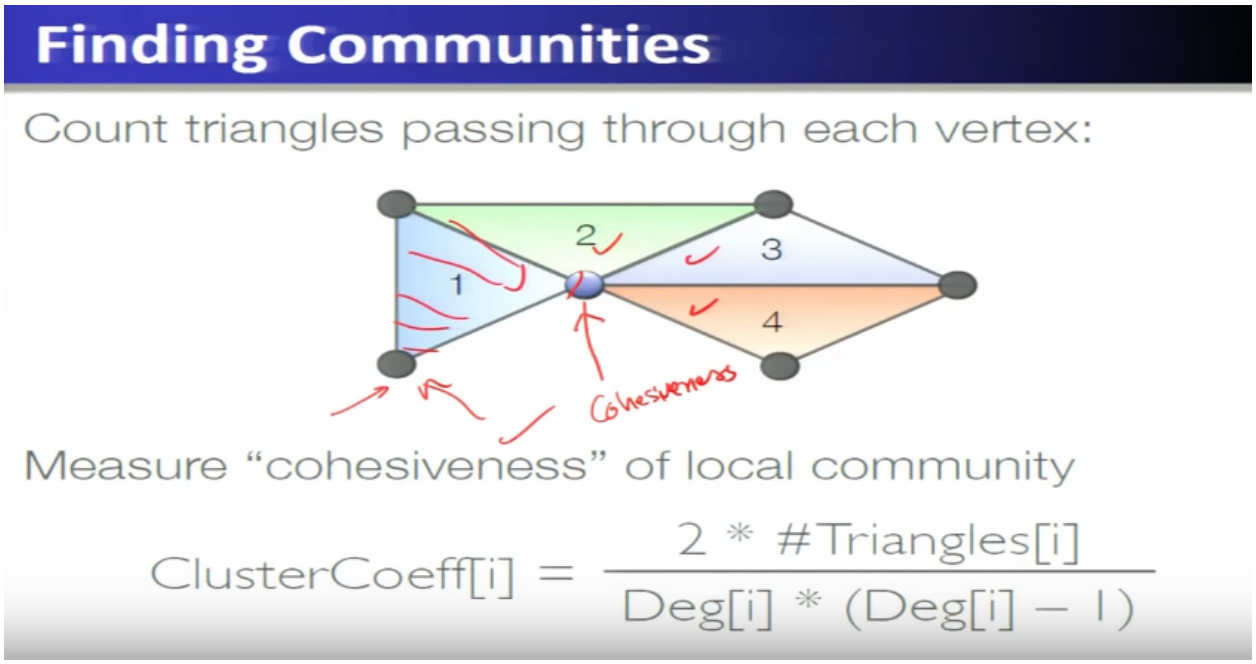
So, let us take, the internal important operation, which is required, to be supported, to run the page rank algorithm efficiently is that. So, first the neighbors or a neighbor pages, neighbor nodes, which represents the pages, they send their data, rather we can see that if you want to find out the page rank, of the node a so, it collects or it gathers, the page rank information, from the neighboring node, after gathering it, it will then compute, its update, it then it will update, its page rank and then it will again, disseminate and then it will disseminate the same information again that is called,' Scatter'. So, all these three steps, gather, apply and scatter, is treated as the super step that is it will be treated as, one single operation. And if it is supported then writing, the PageRank algorithm will be quite easier and also will become very efficient. So, these kind of constructs, are being provided, in the graph X that we will see and using these constructs, the GraphX also, provides the PageRank algorithm as its library functions. So, all the library functions, which are provided as the graph algorithms, in GraphX are highly efficient and optimized in this particular scenario.

Refer slide time :(23:59)



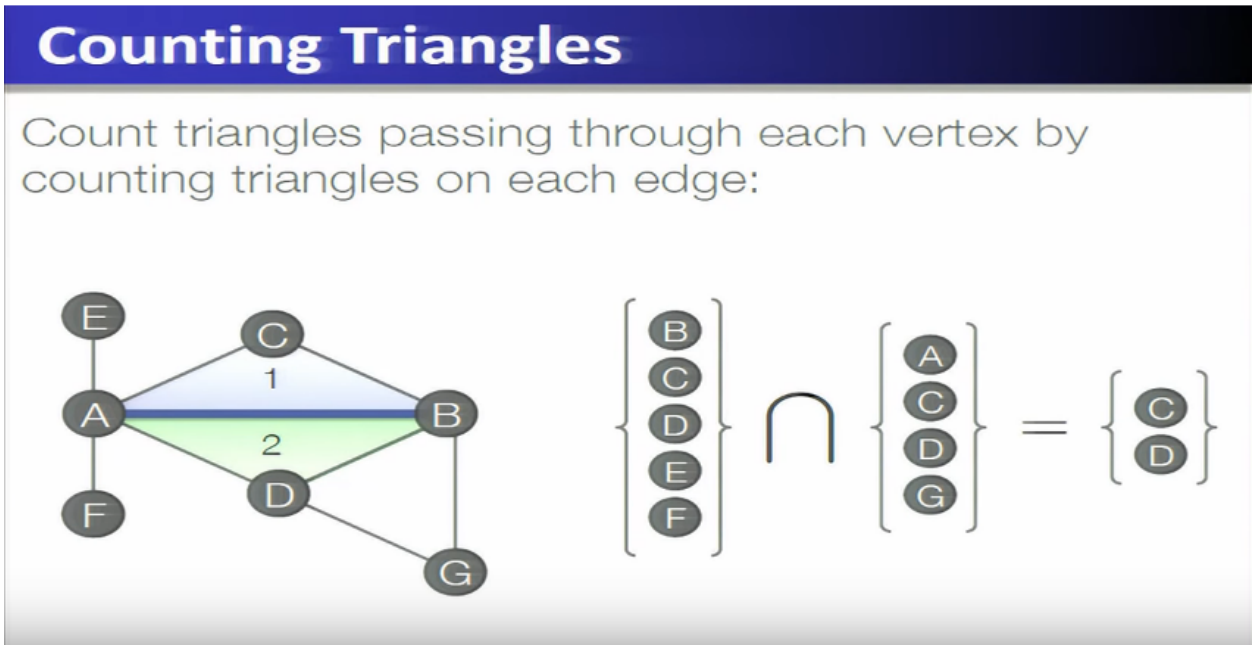
Similarly we can also see that, shortest path algorithm also exists, as the library, in the library of GraphX, is also using, this gather apply and scatter paradigm and using this in parallel, at all the nodes this particular function, will be performed ion's data parallel and computation parallel operations are being done, in to support these algorithms.

Refer slide time :(24:35)



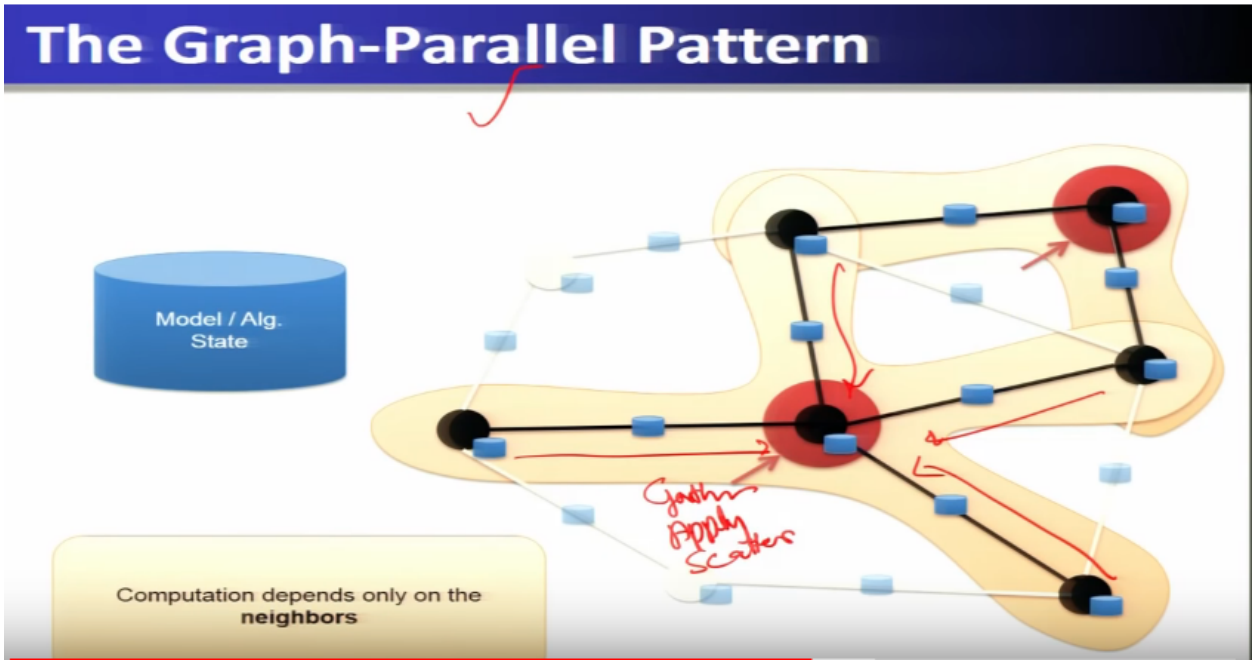
Another algorithm is to find, the triangles, is to count the triangles and in this manner, too we have to measure, we can measure the co Cygnus within that community. So, this particular way, by counting the triangles. So, here we can see that so, as far as this node is concerned one it has one, two, three, four, four triangles, which are passing through this node, one hence this particular, node is a part of large number of community hence, this particular node gives the co Cygnus compared to the node compared to the other nodes, which are not that conceived. So, we can count. So, this particular node has only the vicinity of one triangle, which passes through only one triangle. So, this particular node, which is shown over here, is more coercive compared, to the other nodes. So, here we are going to analyze, these particular behaviors, in any community, which we are going to deal with that. So, counting triangles is one of the important algorithms.

Refer slide time :(25:51)



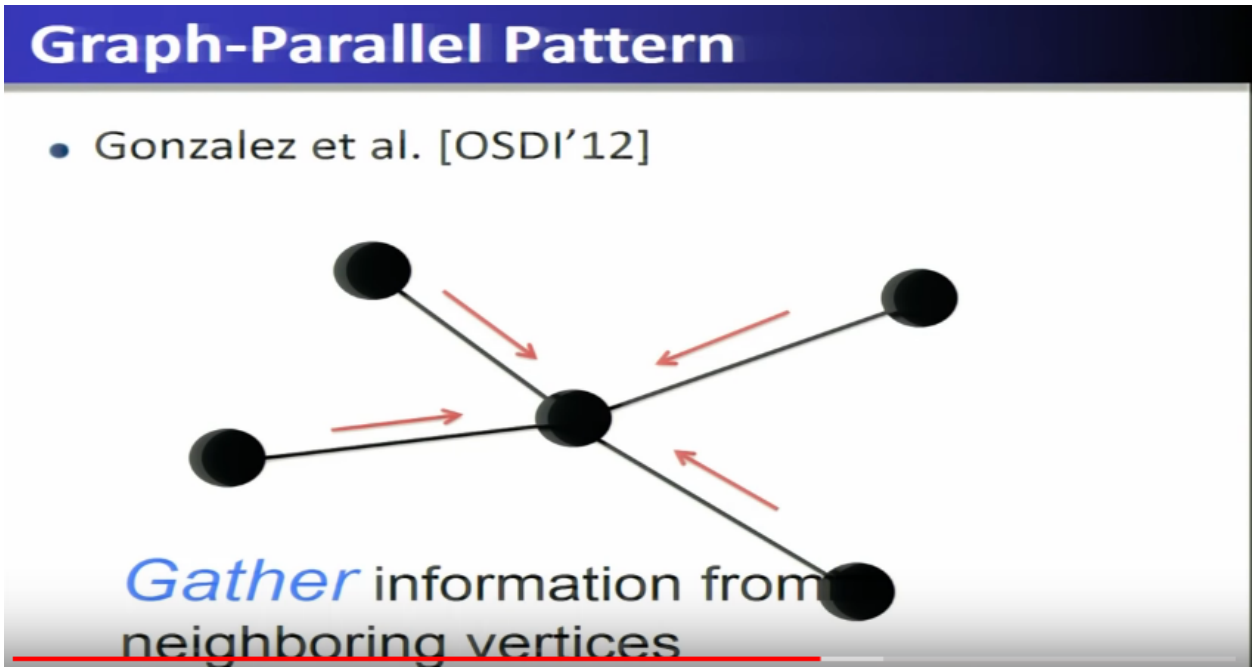
Which is supported as far as the GraphX libraries that we will see.

Refer slide time :(25:56)



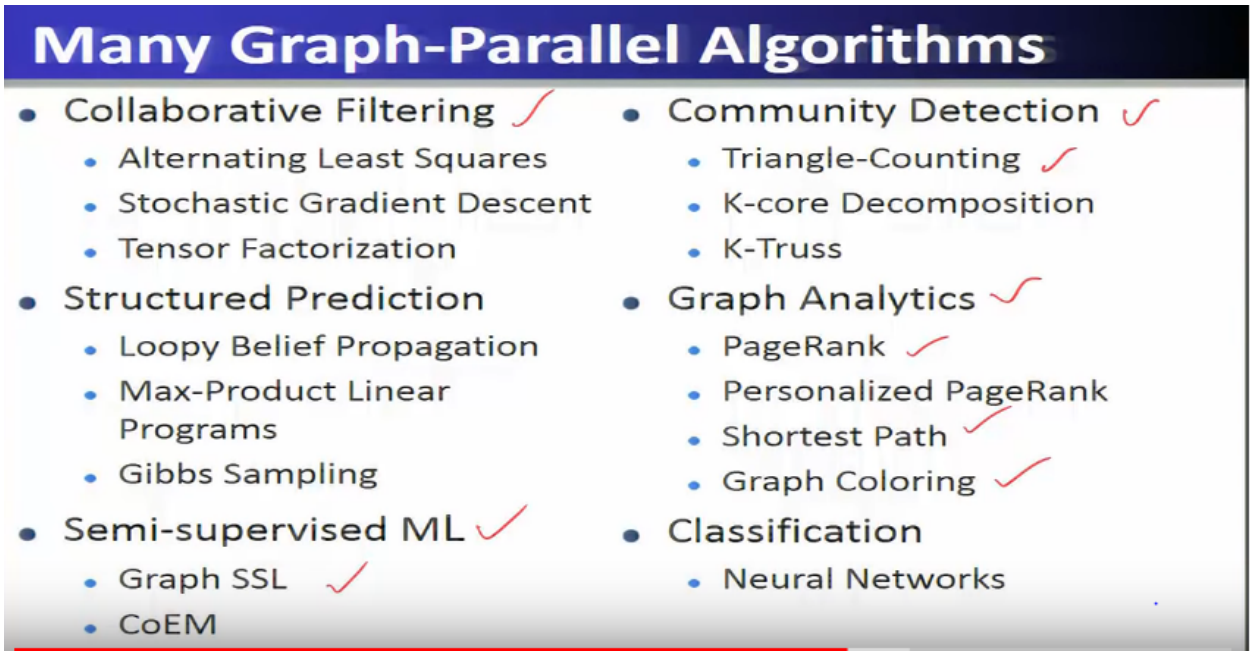
So, now the graph parallel operations, we will see the pattern, how that is all supported. So, consider this particular graph and we are going to take this particular node. And we have to run the algorithm on this node. So, if we can run this algorithm on this node, we can run in parallel at all other nodes. So, let us understand, how this particular algorithm, we can design. So, computation in most of the graph algorithm depends upon the neighbor node. So, it will gather the information from the neighbor node, perform the computation, gather apply perform, the computation and then scatter. This information, which is being computed for updation, at all the ends. So, this kind of operation, is in parallel being run at all the nodes, hence this Groff parallel, computation can be performed, in this particular manner.

Refer slide time :(27:04)



Therefore the graph parallel, pattern is a part of GraphX implementation. So, let us see the parallel, graph parallel pattern is supported by, a gather apply and scatter.

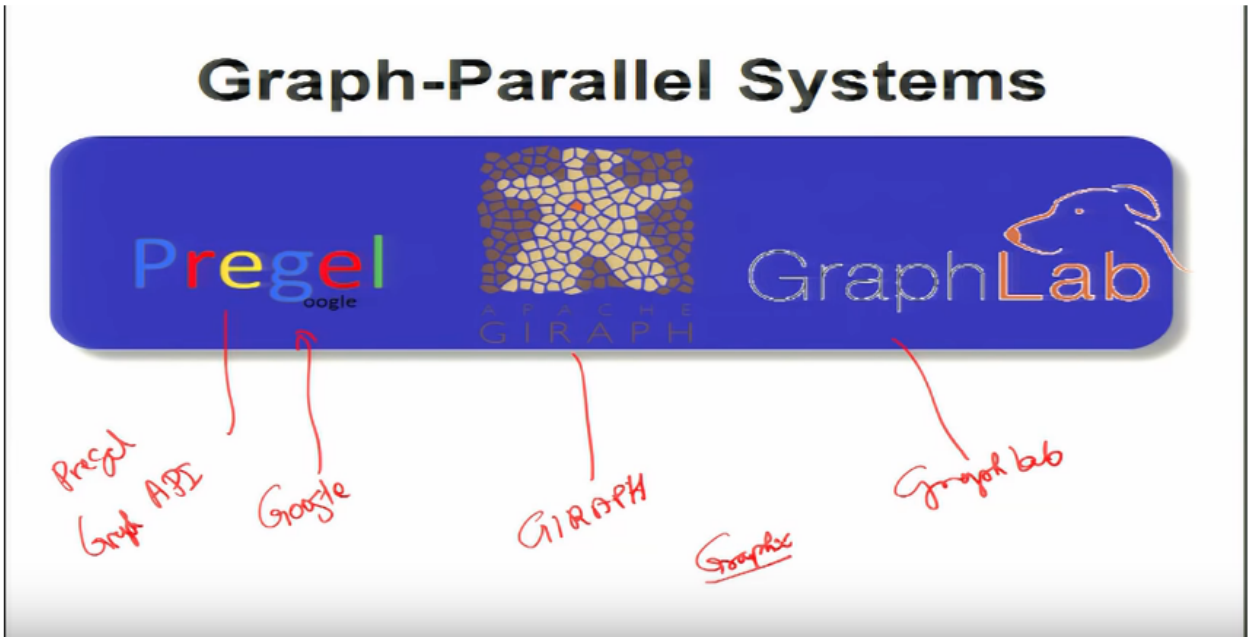
Refer slide time :(27:19)



Notion using this many graph L parallel algorithms, are now available with the GraphX, libraries some of them are performing, the graph analytics as, we have told that page rank algorithm, is their shortest path graph coloring, all these algorithms, are available, which uses this graph parallel framework. For

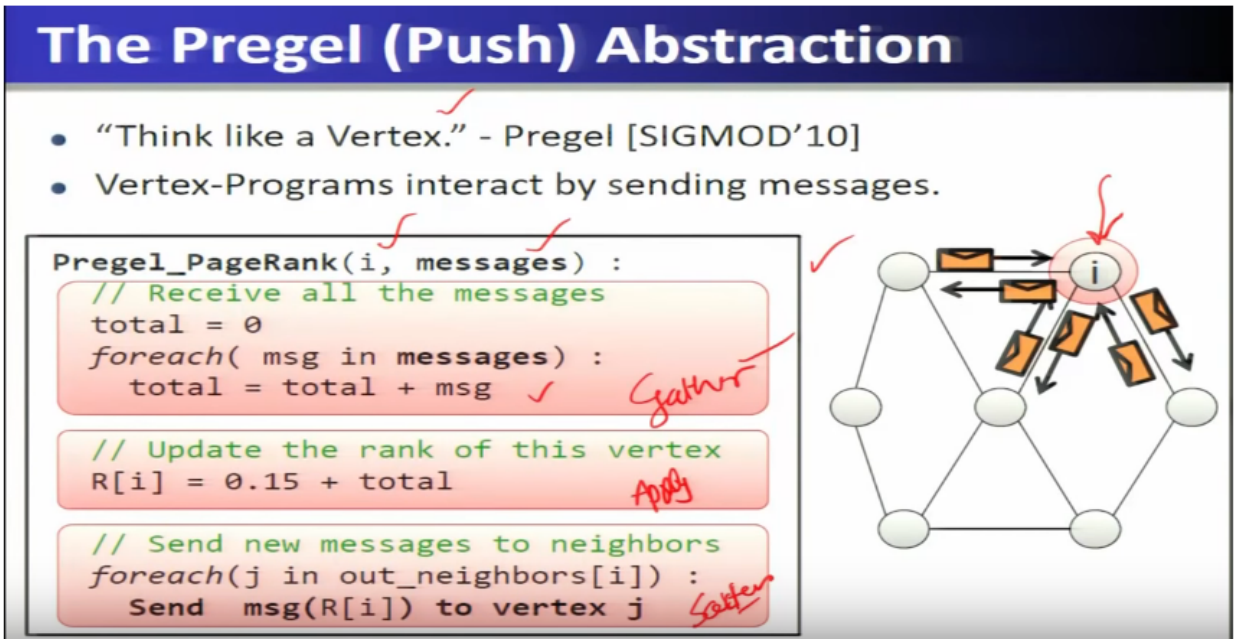
community detection, triangle counting K core decomposition, k truss all these algorithms, are available similarly for machine learning, we have graph SSL and cohere algorithms available similarly, for collaborative filtering, alternating least square, LS algorithm is also, there and stochastic gradient descent tensor factorization, these algorithms, are using the graph parallel approach for building.

Refer slide time :(28:16)



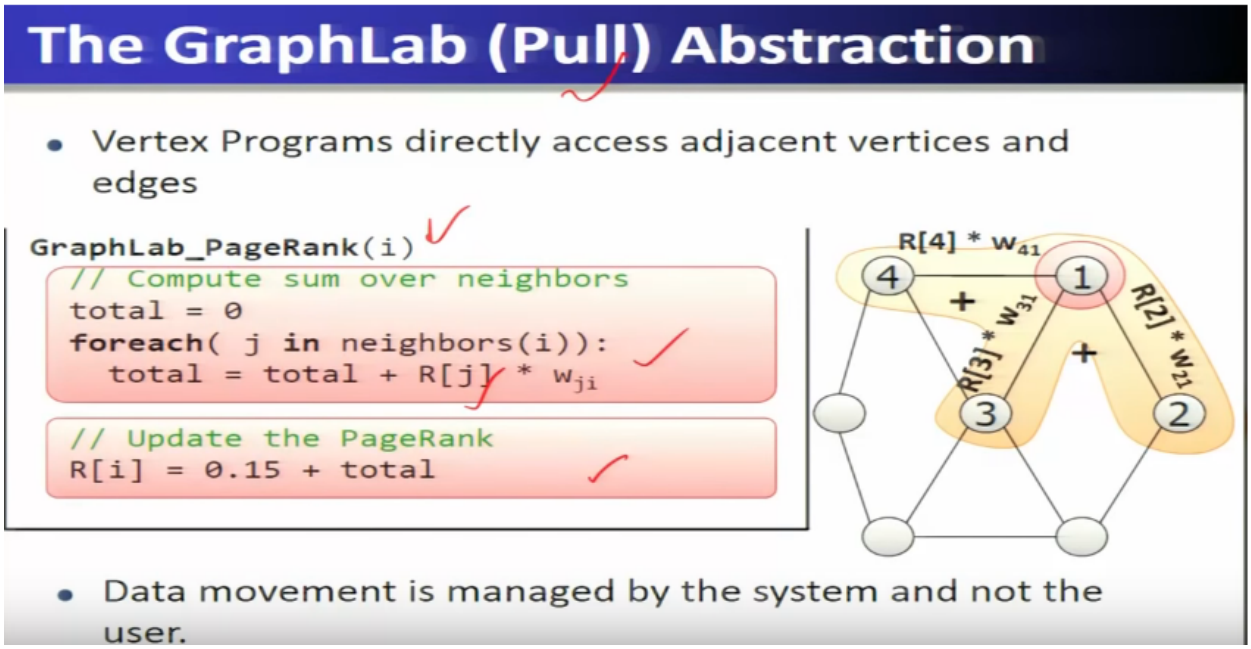
So, graph parallel systems, are available as on date is given, by the Google. That is in the form of Peregrine API that is called a,' Pregel', another graph system, which is called,' Apache Giraph'. And another one is called,' Graph Lab'. And we will see that graph hacks is performing better, in all these scenarios. So, we will expose different API is to simplify, the graph computation and that is the graph parallel computation systems.

Refer slide time :(29:11)



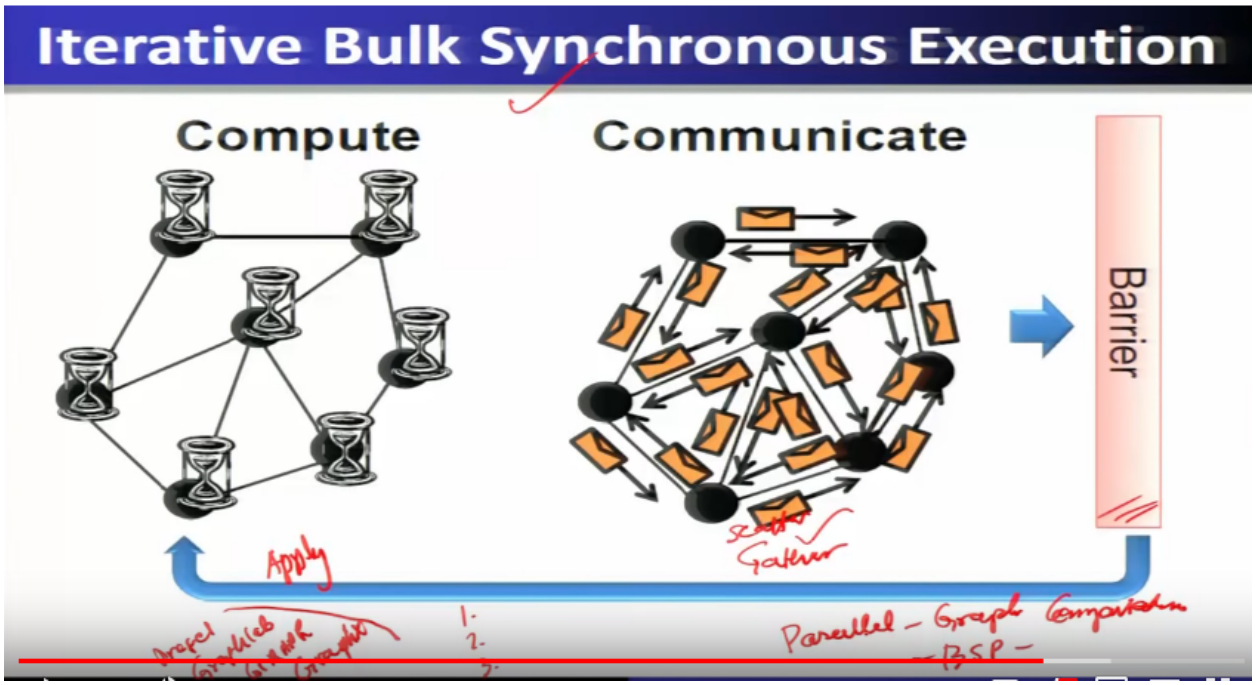
Now let us see that how the pregel, gives an abstraction and in so, that the graph computation, becomes quite easy and for the programmer, interests of the intricacies are hidden into, the framework. So, in the Playgirl, it will be war tech centric operations. So, that means you have to think, like a vertex and it does not require, MapReduce to be to be introduced, here in the program. So, only vertex centric, computations or program can be written, by the programmer hence it becomes very easy to write, the graph algorithms using pregel. Now vertex programs will interact by sending the messages. So, let us see here, in this typical example that. So, if we are going to write down, for this particular vertex I the page rank, algorithm in pregel. So, it will perform this particular page rank on, this particular node I by sending different messages. So, first it will receive the messages, from its neighbor and then it will does the aggregation, of it and after that after that, then it will compute or it will update, the rank that is called the,' Apply Operation'. And then it will scatter these two different nodes. So, this particular paradigm is being applied, at all the vertices in parallel. So, only one vertex program you have to write down a rest of the algorithm is taken care. So, writing algorithm, graph algorithms using Pregel, becomes quite easy without, without bothering about much intricacies.

Refer slide time :(31:09)



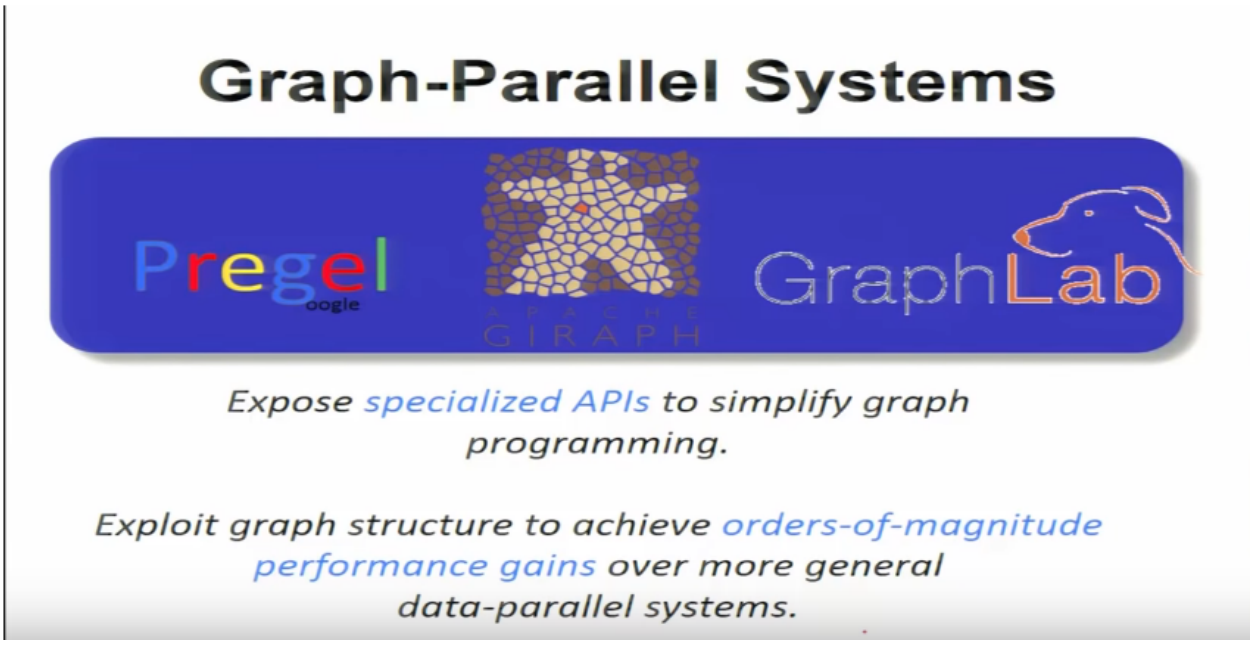
Similarly in a in a graph lab, there will be a full abstraction. So, same algorithm of page rank, if we can write down in a in a graph or lab using pool abstraction. So, it will become a vertex programmed directly access the adjacent, vertices and edges. So, Graph Lab PageRank, for the node I will can, can also be seen as it will. Now compute the sum over the neighbors, using this particular iterations and that means it will collect the information, compute the this one compute, the page ranks out of the neighboring, pages information, using this particular formula and then it will update, its PageRank and that's all. So, here the data movement is managed by the system and automatically it will be handled. So, programmer, the user has to write down for these vertex programs and restore all it will be taken care. So, that means when these, data from the neighbors are required, in the programming construct, automatically the pool abstraction, will get these values, from the neighboring node and that is the part of internal parts, of the graph lamp.

Refer slide time :(32:39)



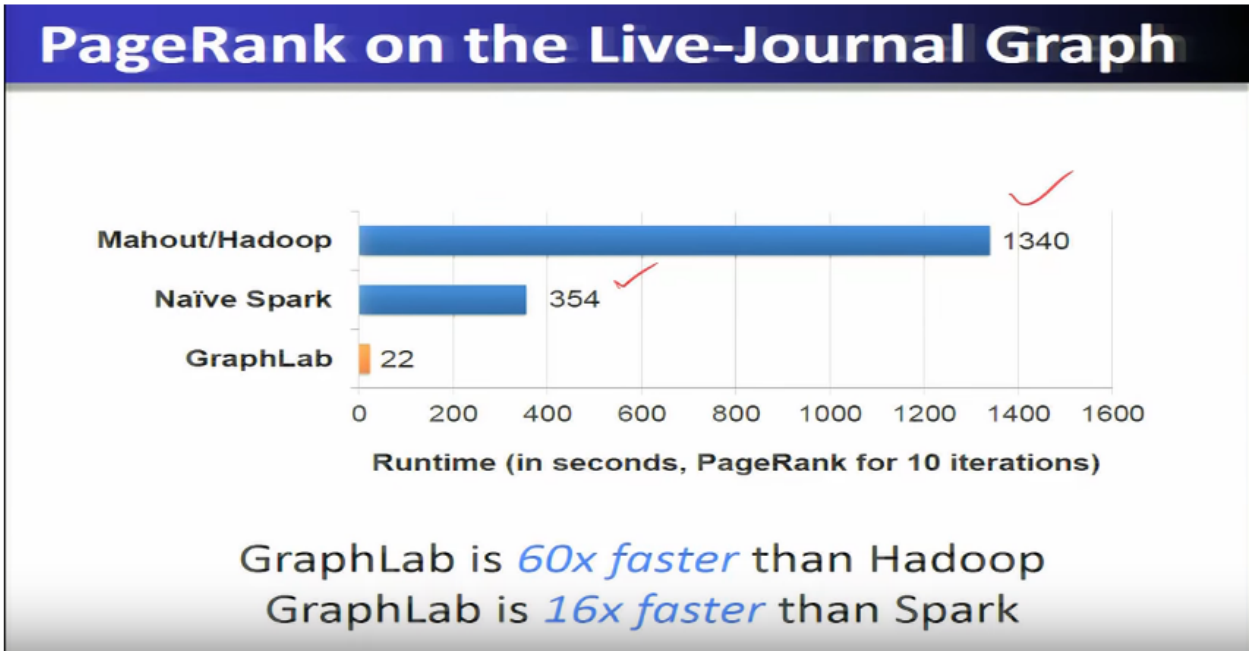
Now in short what we can what we have seen these graph, frameworks they provide, the features in which you can communicate, with the neighboring nodes, neighboring vertices and after the communication is reached, which being obtained or is reached, from all the neighbors then there will be a barrier and then the computation, will be performed and then it will then again communicate. So, this is the paradigm which is called an,' Iterative Bulk Synchronous Execution', which will provide the parallel, graph computations, this particular diagram, shows the BSP a bulk synchronous execution, of the graph computations. So, this you can understand that when you say communicate, you can gather and when you compute then you say gather apply and again when you communicate it will become I scatter. So, again I told you that all these operations, are have different names and paradigms, in different the graph frameworks, let us say pregel and then the graph lab and zero I GraphX. So, all these are performing the same set, of basic operations, to support the iterative bulk synchronous execution. And using that the vertex programs can be written, in a very simplified manner, for different graph algorithms.

Refer slide time :(34:38)



This we have already, seen that using pregel or using zero or using graph lab, it will expose, the specialized API is to simplify, the graph programming. And exploit the graph structure, to achieve orders of magnitude performance grain or the more journal data parallel systems.

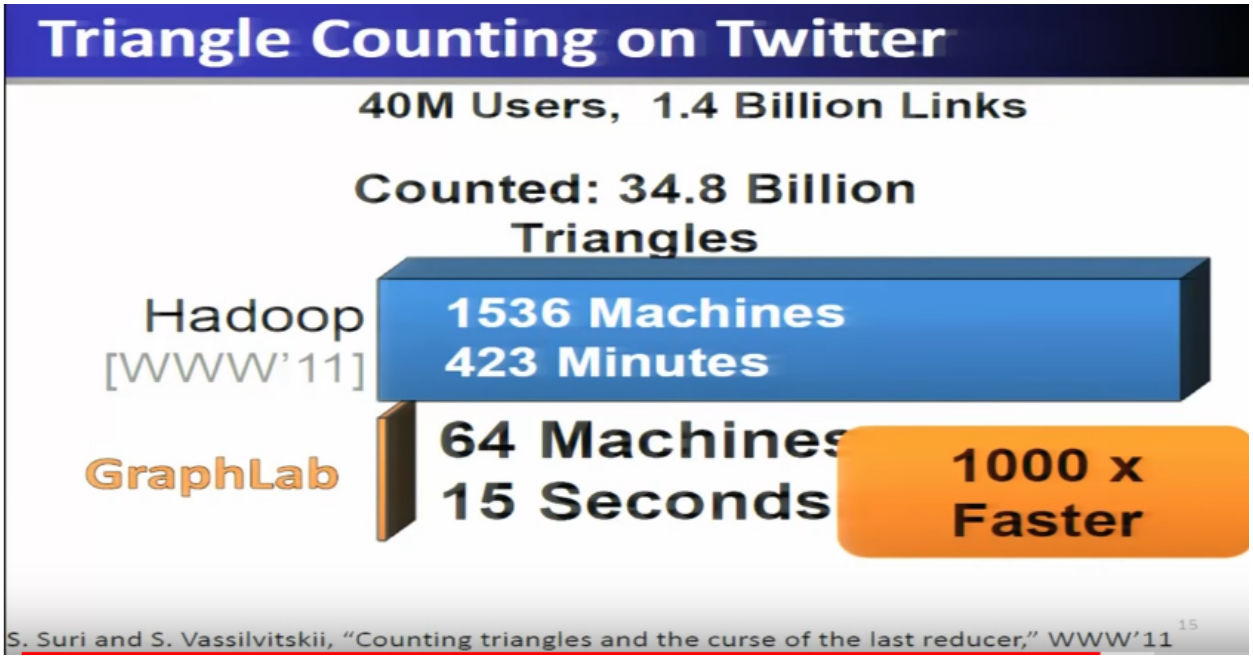
Refer slide time :(34:59)



So, for example the PageRank on Live Journal graph, if you see that, you see that if it is done through the Hadoop, it is taking 1, 3, 4, 0 of this particular time, compared to the Naive spark, if you use it will take 3 43:54 and graph lab, is 22 and even the this one, even if you see the GraphX, which is even, faster than

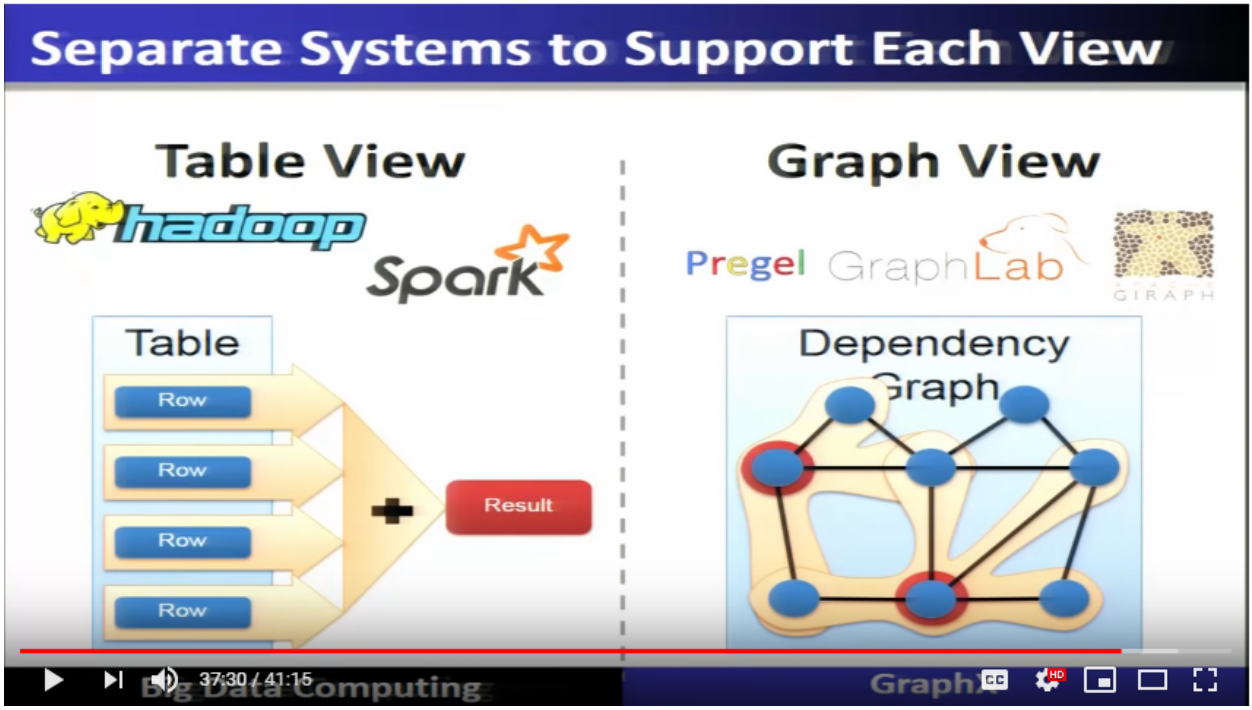
the graph lab also. So, we have seen how this particular improvement, in the performance, is gained is because, these basic operations and the algorithm, are optimized and being supported, by the core spark and optimized, by the GraphX for this algorithm. So, those algorithms, which are part of the libraries, of GraphX, are much more efficient, in the performance wise compared to any other framework.

Refer slide time :(36:11)



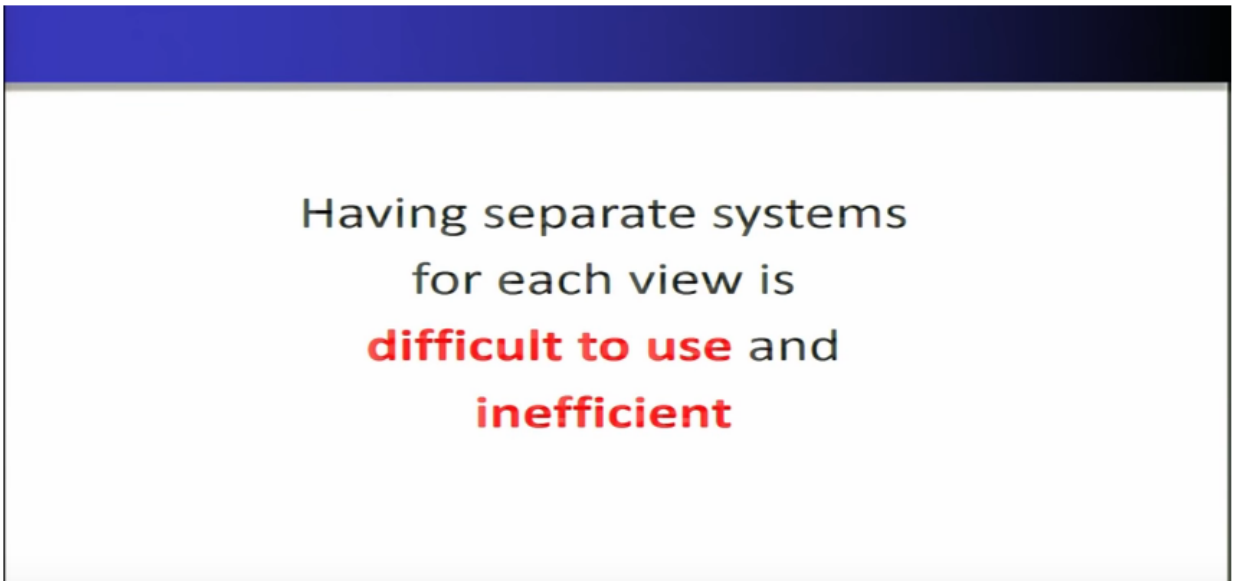
Similarly the triangle counting and if you see that it is 1,000 times faster in GraphX.

Refer slide time :(36:20)



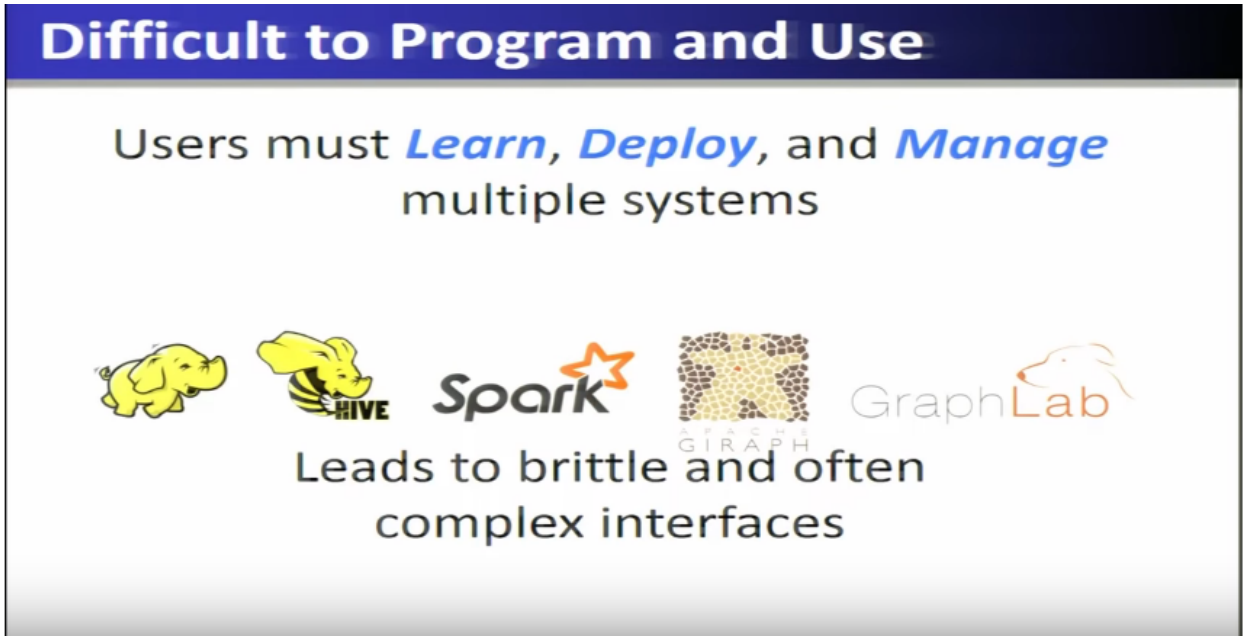
Let us see this we have seen now. So, basically now, we have to go and see, how these graph systems, are being supported, for their implementation. Now as far as these data, we have seen that the data can be easily represented, in a form of a table and that particular, view is called a,' Table View '. And the computation can be easily performed, on this view that is called,' Table View ', to get the results similarly there is another, view that from this table view we can get we can convert in the form of a graph. So, it will become a graph view and then various algorithms can be applied, with the help of pregel APIs graph lab. And giraffe so, this particular graph, will be useful for applying, different graph algorithms, to get the results back. So, we will see that there will be two different views, from table view, we are converting to graph view and then again into a table view. So, there is a dependencies, between table view and a graph view. Let us see how the different framework, they are handling this kind of views.

Refer slide time :(37:40)



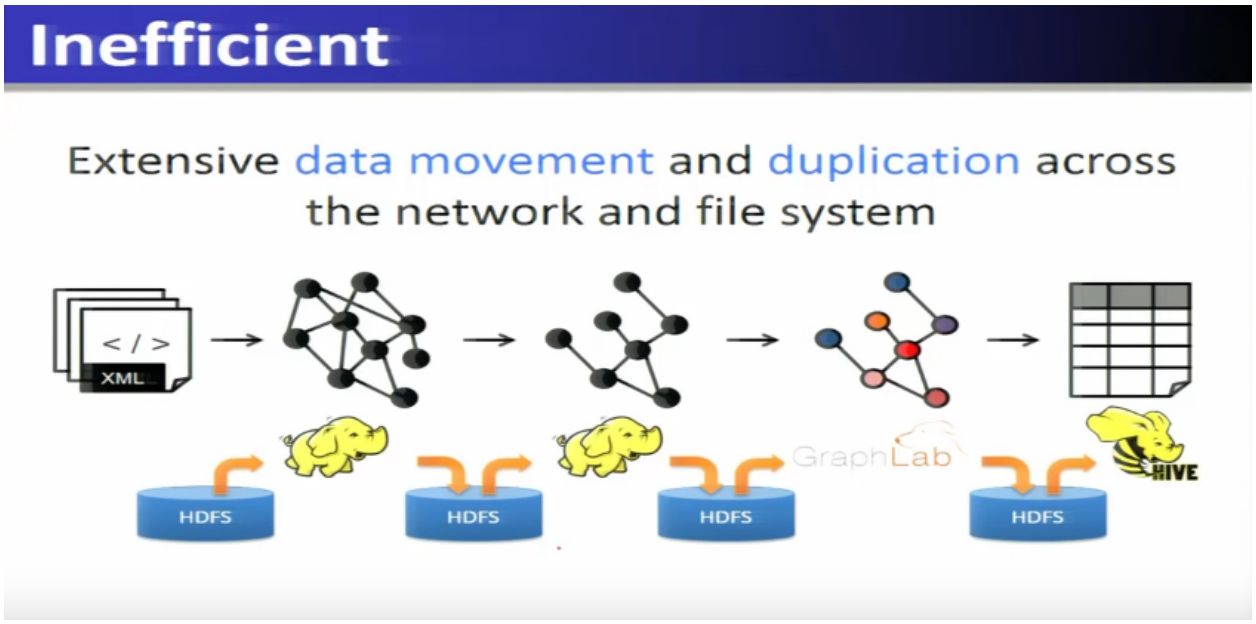
So, having separate views, for each view is difficult to use.

Refer slide time :(37:46)



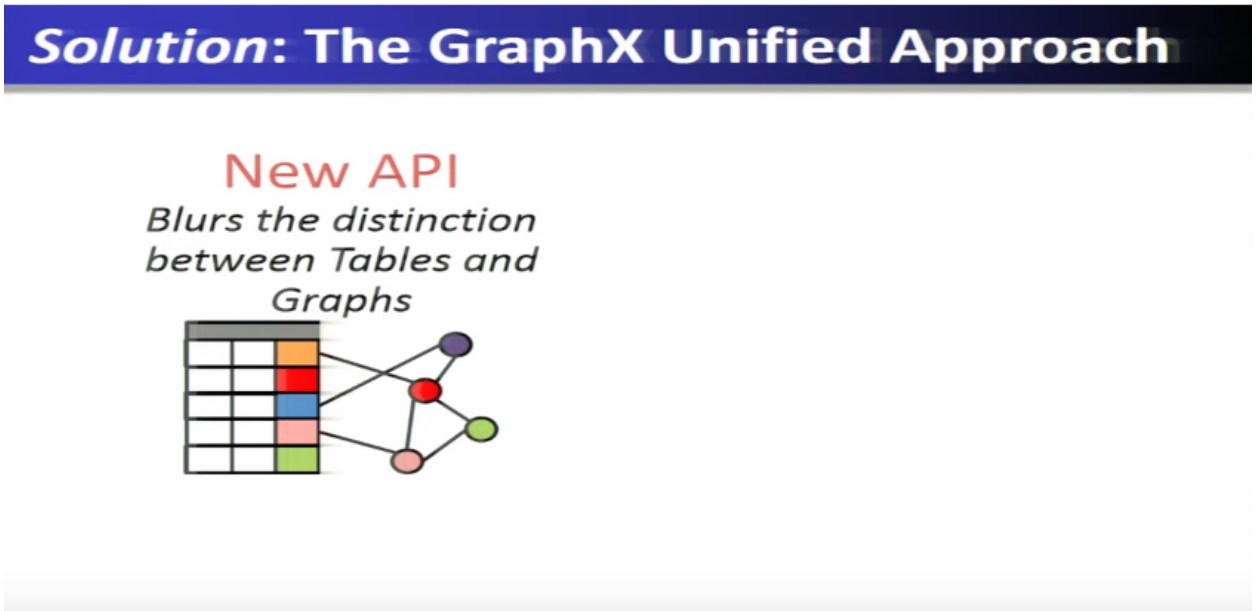
And also will become inefficient.

Refer slide time :(37:47)



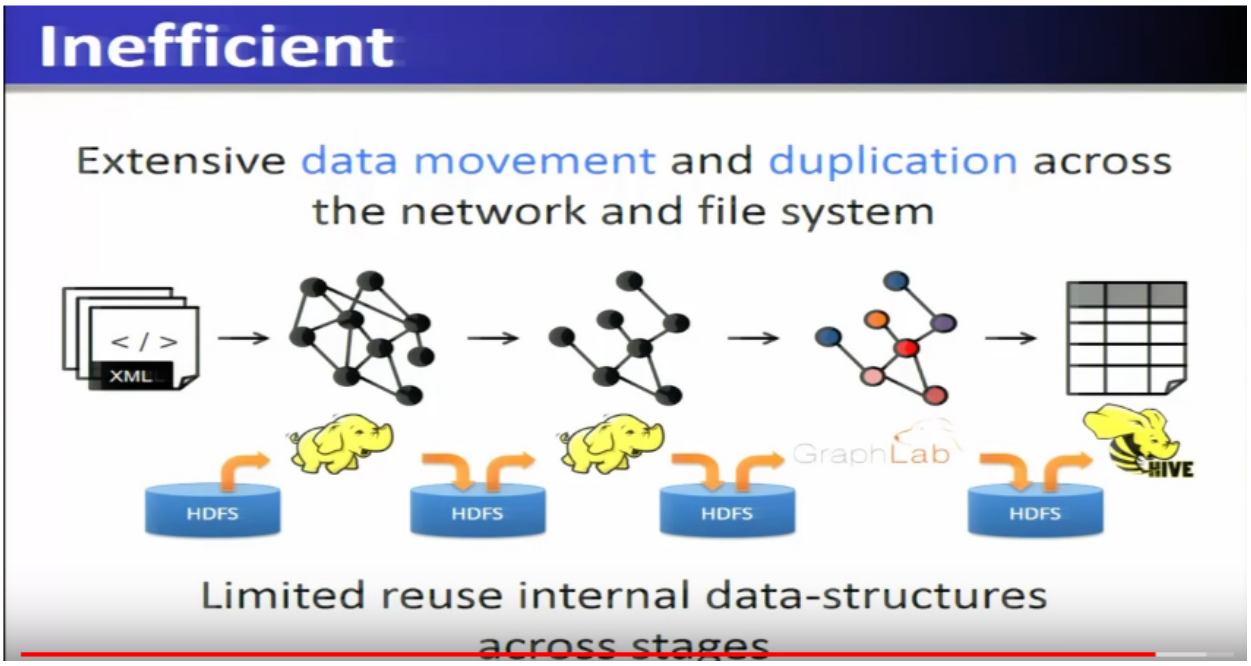
So, why because the data will be moving and also that there will be a duplication, into the network and the file system, therefore the this particular changing of views, from table view to the graph, view and again back to the table view, requires the data to be moved, through the HDFS file system hence, it will become inefficient in most of the scenarios.

Refer slide time :(38:18)



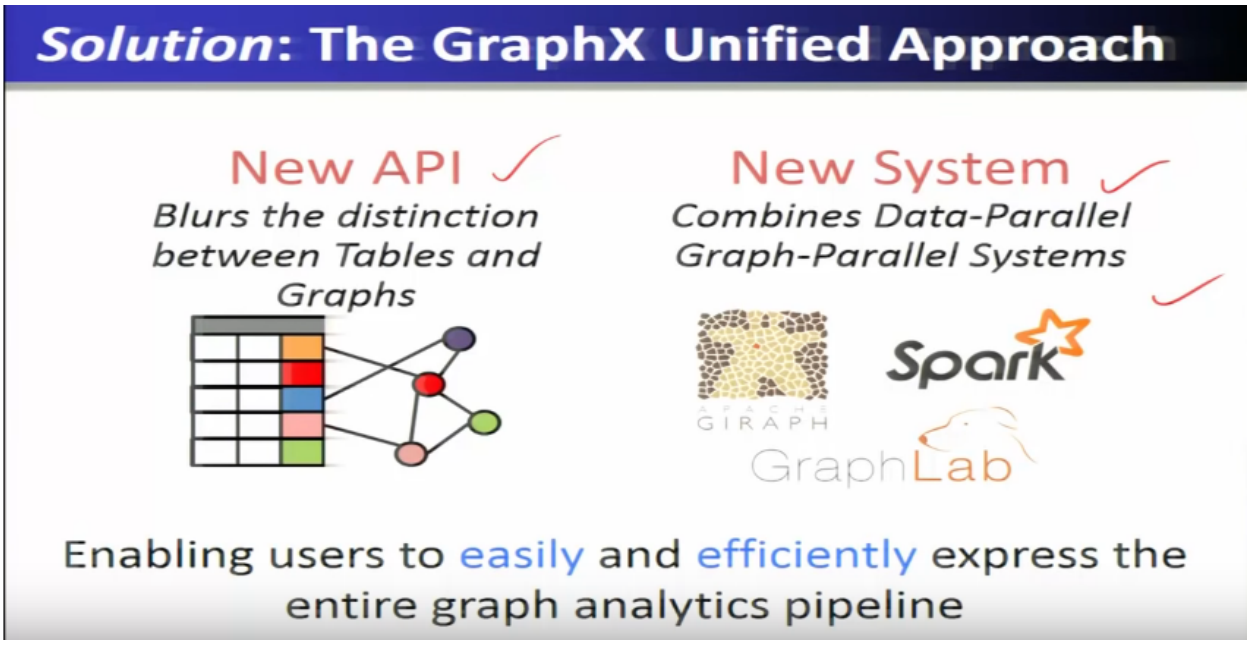
Now as far as the GraphX why we are understanding, about GraphX is that it will give a unified, approach and it will not incur these inefficiencies,

Refer slide time :(38:31)



of internal are basically, the using use of file system and the network to store the data in between.

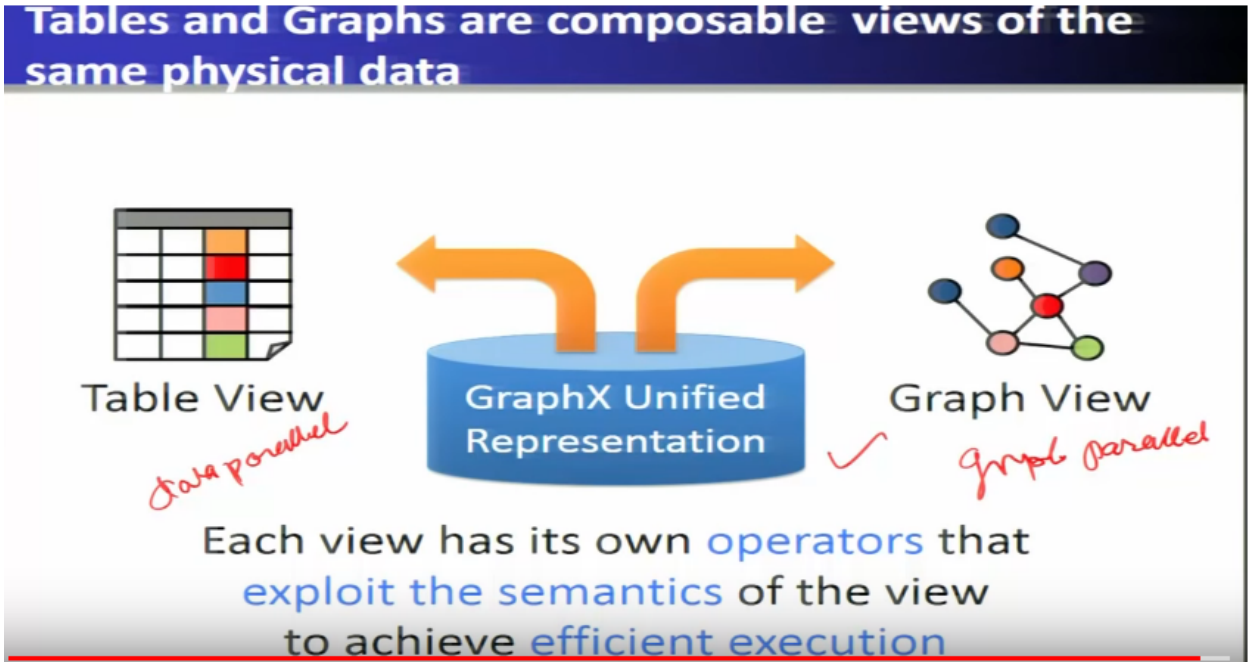
Refer slide time :(38:38)



Hence it is called a unified view. So, GraphX provides a unified approach, wherein the data, will not be required to store, in SDFS. So, between different views, automatically they are being integrated together and data can remain in memory. So, all the operations are supported, with full efficiencies. So, therefore this new API is which are supported in the form, of the graph X will, will blur the distinction between the tables and the graph. So, the new system, which will now, evolve in this in the form of a graph X as the unified approach will combine the data parallel and graph parallel systems together. So, this will enable

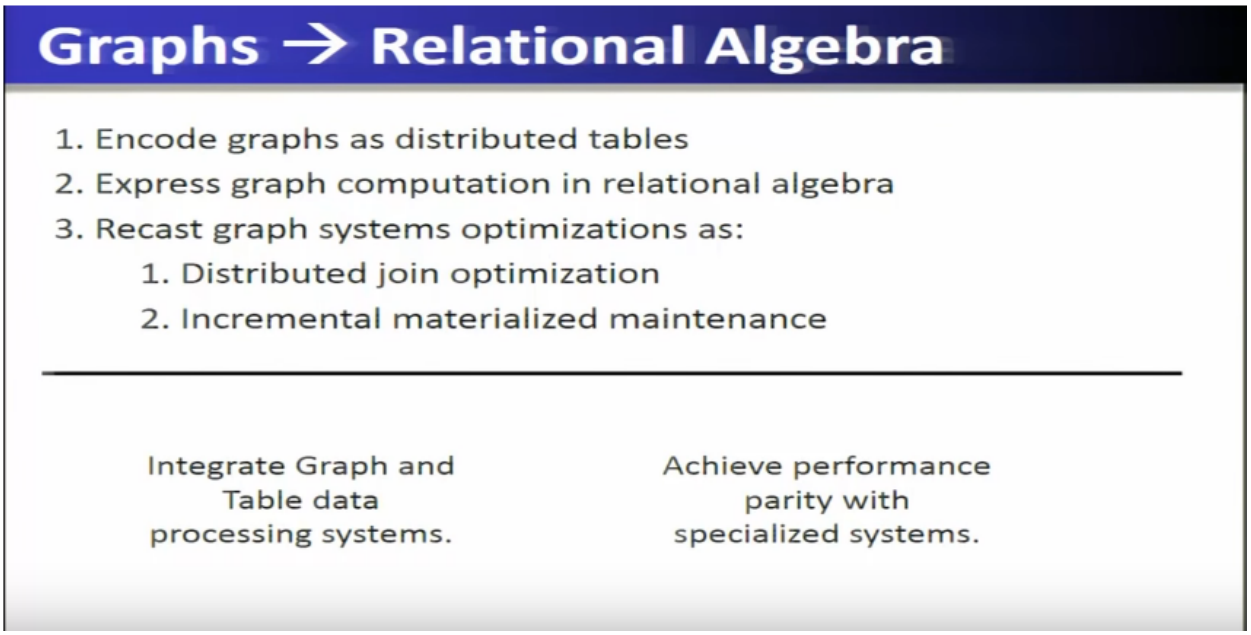
the users to easily and efficiently express the entire, graph analytics pipeline that we have that we are going to see.

Refer slide time :(39:36)



So, therefore the graphs and the tables are composable views of the same physical data. And this GraphX will provide a unified representation. So, each view has its own operators that exploit the semantics of the view to achieve the efficient execution. So, table and the graph both views are together unified and is being supported in the GraphX. So, therefore data parallel. And the draw parallel operations, are integrated together, iron is being supported, as unified view.

Refer slide time :(39:20)



For better efficiency, graphs can also be represented or map in a form of relational, algebra and therefore, we can encode the graph, as distributed tables and Express the graph computation, in the relational algebra recast the graph systems, optimizations as distributed joint, optimization and incremental, material maintenance.