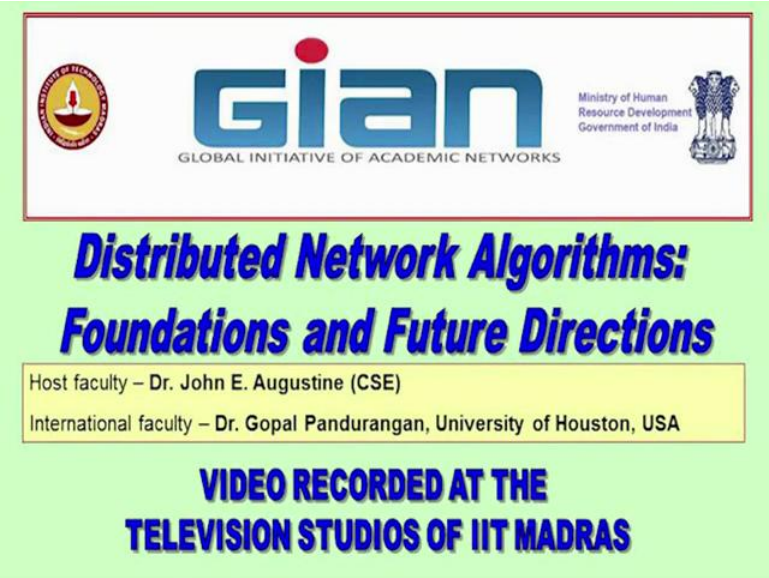
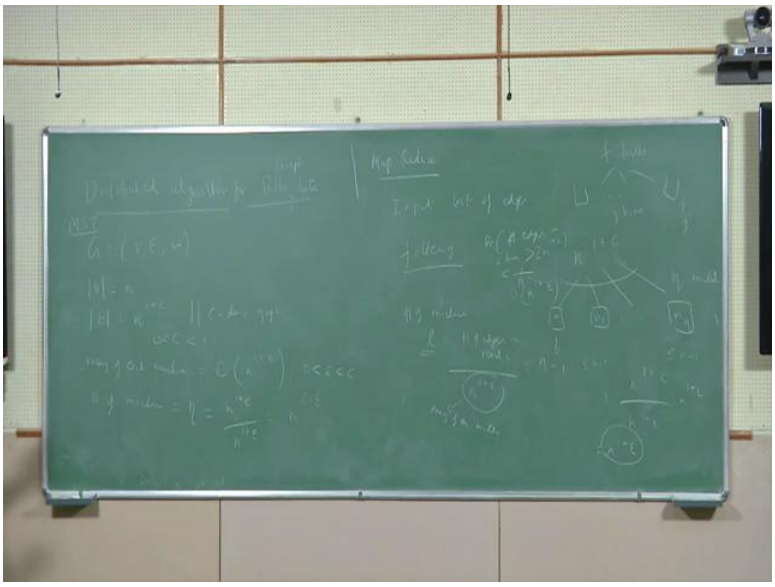
Algorithms for Big Data
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture - 48
K-Machine Model (aka Pregel Model)

(Refer Slide Time: 00:15)



(Refer Slide Time: 00:20)



I want to now go the next model, which I think is really motivated by looking by motivated by looking at the performance of graph algorithms in map reduce. So, people now it is well know that graph algorithms even actually Google people are first probably to notice that map reduce is not very good for doing algorithms. Because this is good, but these sort of set up in the fashion setup so that I get constant terms, but you can see that in general it is very artificial model, because I am basically this model artificial because you can see that the modeling assumptions depend on the input.

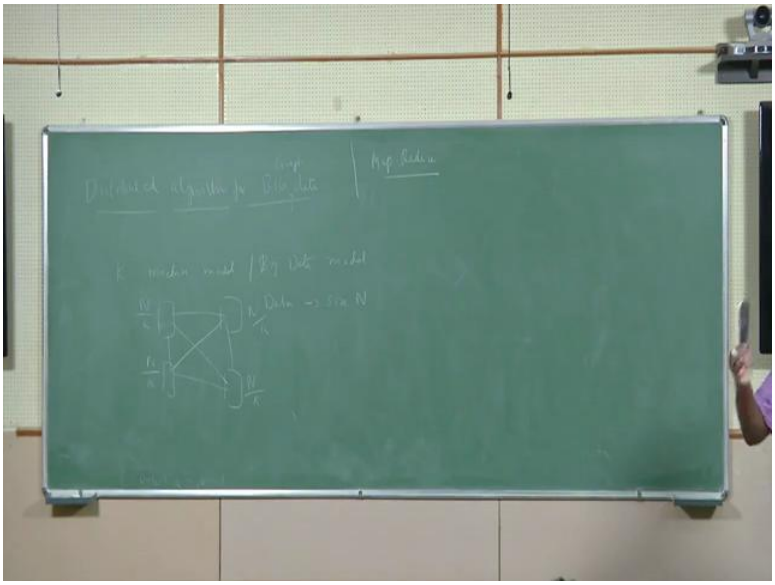
If because I am assuming it is then and then it is 1 to the n plus epsilon so for example, if the graph is already parse then this does not make sense. If the graph is sparse then I can; that means, they only order n edges, then I cannot assume these many number of things then it my number of my memory size would be little o of n, because it is always should be sub linear. So, my model changes with the data so that is not a good model that is not very good model.

And either still do a but the number of rounds will be much more then people also find there is in practice that actually map reduces not very suitable for algorithms that take lot of iterations, lot of rounds. And many graph algorithms are like that whether you take MST, even worst algorithms like (Refer Time: 01:52), shortest paths, they are much more iterative, they take probably diameter rounds or something like that. So they come with another system called Pregel, so again Google guys.

Student: Is it pregel?

Professor: Pregel, pregel. So, now the lot of other systems are based on that, but the there is a different, but all systems that are that came up the pregel and other systems are came after those they are based on a some more different paradigm. So, basic that paradigm is basically is very close to distribute computing paradigm that you have seen till. There it is what it is there you have a paradigm which is so let me go to back.

(Refer Slide Time: 02:36)



So, the paradigm is much more natural for naturally suited for distribute computing. So, where your bunch of machines because your bunch of machines the machines cannot change machines is always fixed, I mean you cannot change the machines means a memory based on the input. So if you have liked super dense I cannot say the memory is this much this sparse, I cannot make the memory smaller and so on. So, machines are machines and they have some memory what all they get. So, you have bunch of machines let say k machines that is why it is called the k-machine model, it is also called the big data model.

So, bunch of k-machines, let say poor machines here and they are catered by a network that means, they can communicate with each other. This matters that not it is completely transparent. So, this in Google will be the data center network, so they compute they exchange messages.

So right now we are for simplicity, we are going to assume that it is a complete that means, every machine can talk to any other machine and you have your data, so the data is there, so maybe it is of size n. So in the graph setting, it will be the number of edges plus a number of nodes that is the size of the input that is distributed among the machines, because you have to distribute. Because you have obviously, overall although

one particular machine cannot hold the full data overall it should be able to hold them otherwise you cannot do anything right.

So, each machine will get approximate it is a balanced distribution approximately. So, each machine will get this is the basic model, we will come to that little bit more little bit more specific shortly, but this is the whole thing, so they are distributed. There is no memory and all because I am assuming by default that there is so much memory to hold at least one by kth fraction of the input, obvious I cannot do anything. So, each machine has enough memory to also the memory is not even issue here, memories implicit because we test through a every machine the only k machines that can try to process the data each machine should have at least 1 by kth fraction of the memory to hold the data.

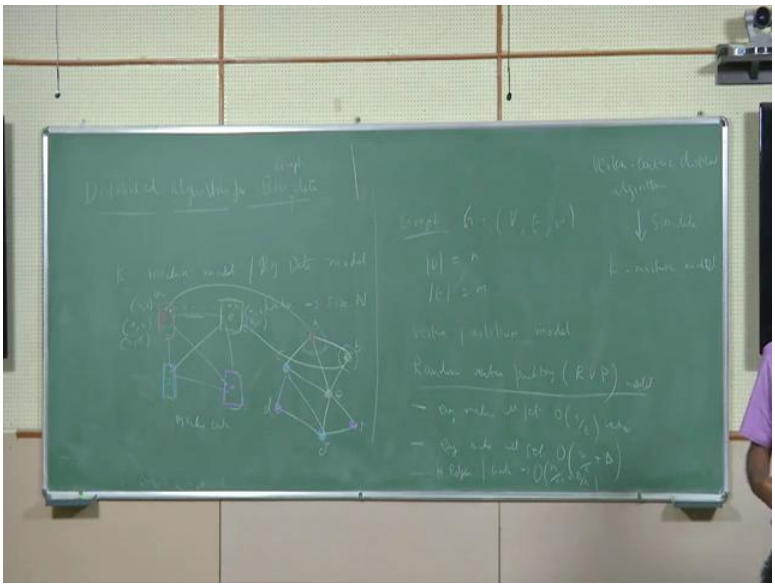
And they jointly process whatever computation you want to do MST, or may be this sorting from this n may be list of values you want to sort them together whatever be the problem your favorite problem graph problem or a non-graph problem whatever and they jointly compute, so that is a model.

So this model is you can that it is quite different from the map reduce. For one thing thus it is not state less so this data which is there in this machine is always there. Of course, it can get an so it computes this so this very much like it is exactly like your distribute computing model, so that is basically the model that is inspired by pregel. So pregel is false, so this kind of model captures pregel and other systems that come after that giraffe now those so many systems spark, so all the systems are in the setting this captures all these thing. So even map reduce (Refer Time: 05:46) make sense, but not in that flavor so here it does not model memory, memory is implicit.

So there is state of course, each machine has it is own memory it has state for example, if it is get n by k this state is already there. There is no need to move it around, so it has this initial input and that part your input is free for it, you already have this. So, only it has to pay for getting messages and things like moving like that, but this is very much straight full. So, every round it is exactly your distribute computing model; so it gets some input, initially the n by k fraction of the input, does some computation local computation, sends messages receives messages and so on.

At the end, you have to jointly compute something let us say MST means, if we look at the overall output of all the k machines n minus one edges should be output. So, ideally here I would really want the load balance because you do not want each machine to be a bottle neck so if there are n minus one edges in the MST, ideally I want all the machines to output one by k fraction of that. So everybody will output 1 n minus 1 by k edges. If I allow one machine to do lot of things then that is a bottleneck; I want to completely balance the computation and communication everything.

(Refer Slide Time: 06:58)



Let us see an algorithm in the in the k machine model. So, I am going to design an algorithm k machine model, but before that let us look at this paradigm of graph algorithms in the message passing sensor, and that is sort of motivates this whole model. So, this kind of model is very natural when try of think very natural to model graph algorithms, because graph algorithms you can think of them as actually message passing algorithms even centralize algorithms, you can think of like that like I was told in this object oriented way.

In fact, if we write a code in this completely object oriented sense, you will not treat graph a node as like as an array point in an array, but you treat as an object. So, each node is an object and it computes by sending messages those are the methods, methods is

basically sending messages to it is neighbors. So, and that is actually the model that was initially the motivation for this pregel, and also it is exactly like that.

So, now but let us come back to this and let us go little bit more specific and look at a particular thing. For example let say we looked at distributor. So, we can think of every graph algorithm in the distributed sense. There for example, for MST, so in the MST, we looked at the distribute algorithm for MST, where each node perform some computation.

But now I am going to think of this model as what I am going to call as so all the graph algorithms that we have seen till now like last week right, there you can think of them as vertex centric. What do you mean by that? Think of the graph as a network. The input graph here think of this network, if the graph were actually a network and you want to compute let say MST that is what you would have done. So, each node is a processor and the edge graph edge is actually a link and that is what we did.

So, you can see that initially the it has the same input model, so each processor only has it is local knowledge it means it is the incident it knows about itself may be it is id and it is label or whatever and it is incident edges. And it communicates by sending messages. At the end, every node should know what edges which of incident edges belong to the MST which do not, so of course, there is another way to look at centralize algorithm in a completely distributed way.

In fact distributor algorithm are exactly, so they are vertex centric, because each vertex you can think of it is processor all though it is a kind of a dummy thing right; if you think about an object or a thing, but it is you can think of it as the processor. That is means think of the input graph as a communication network and apply the distributor algorithm and you still get the result.

So, a distribute algorithm is also centralized algorithm where you try to sort of simulate it on the thing. But here the vertices are not process, the vertices are dummy right. So, the processor actually the machines, the machines are actually doing the computation. So, you can try to simulate a vertex centric distributor algorithm in the machine model, where the computation done by a vertex is actually simulated by the machine that as the

vertex. So, you take a suppose I want to solve the MST problem, you look at the distributed MST algorithm let say we looked at the GHS algorithm, the pipeline algorithm the other the d plus quite of an algorithm, whatever the algorithm you have, they are all distribute algorithms they are all vertex centric. Because they assume that each vertex is now in processor as was a processor in the links are the edges are links.

I am going to try to simulate that in the k machine setting. So, this model is machine centric because the vertices are just dummy they are just data right. So, but in the vertex centric model I am pretending the vertices as process, but here they are they are just data points the actual computation is done by the machines that are holding the data. So, this is machine centric. So, the main thing so one way of implementing algorithms in the k machine model is to take your vertex centric algorithms that we have that has we known for like (Refer Time: 11:28) we studied that last class, and try to simulate them in the vertex centric machine. So, simulate vertex centric algorithm in the k machine.

So let us do that. So that is gives you so that immediately gives you way to write like algorithms in pregel and giraffe, because you take your favorite graph problem, and take the distributed algorithm for that problem and you write the code for that, as though as the vertex is executing the code. The machine will take care of that of course, if we say at this vertex does that it is not the vertices will do that seamlessly the machine that has the vertex will do that. So that is means in the distributed algorithm it may so in the distributed algorithm, this is the input graph, but you pretend this input graph as a network. So it will be sending, let say a computing MST, maybe I am in what happens in the MST computation, in any distribute algorithm, there is always this each it goes in rounds, receive, compute, send; receive, compute, send.

So, I have to simulate every step every round of the distribute algorithm the vertex center distribute algorithm that is the vanilla distributed algorithm you have seen standard in the machine model the k machine model. So that means, if this in the vertex centric algorithm if a send the message to b, it should be simulated in the k machine. Of course, a local computation is easily simulated, if whatever local computation done by that machine which has a will do that computation, one a(s) behalf then. And if a sends a message to b this machine will send a message to b. How will it now where to send

because it works in a hashing that is why the hashing is important.

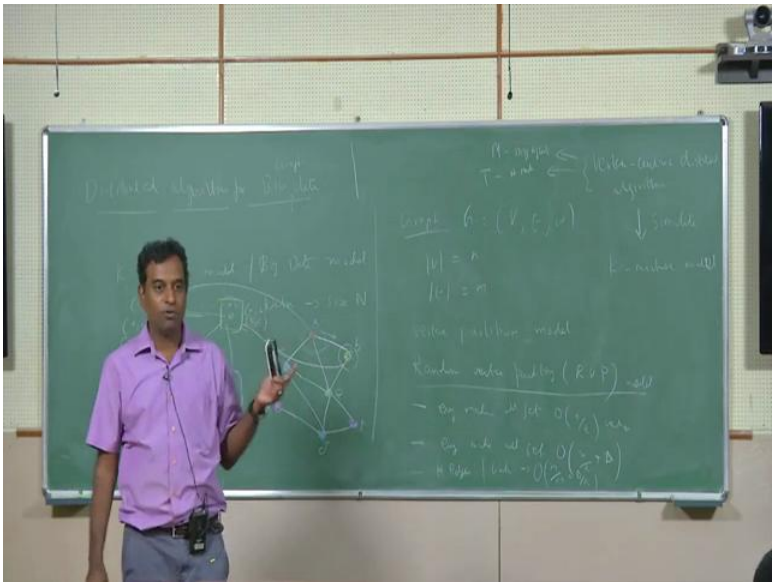
So, in the original vertex centric algorithm, if there is that algorithm set send this message to neighbor b this machine will the machine that whole this holding a it has all the incident edges; that means, it will have a b edge. So, it will look at b, hash the b, and it knows where it is mapped to of course it can do it once, it does not have to do it every time it has this table of things so it knows where b is asked to send that that machine will deliver it to b.

So, that is so this is a very basic way of simulating. So, let us do this steps take this, this is the standard way that is it. So, what is so of course, here we know the complexity here in the complexity in the vertex centric let say I have an any arbitrary algorithm you can think of a MST as running example, but think of an arbitrary example so in the vertex centric thing. What are the communication parameters we were in interested in the complexity measures?

Student: The messages and rounds.

Professor: The messages and rounds, OK.

(Refer Slide Time: 14:26)



So, let us look at that. So, in the vertex centric world that is you are distributed, so you add what are the parameters you add the message complexity. I am going to denote that as big M, do not get confused may be I am used because that is what, we have so big M is the message complexity, small m is a number of messages, small is the number of number of edges sorry number of edges. So, small m is the number of edges, big m is the message complexity. T is the number of rounds. These are two things.

I also need one more complexly measure that we usually did not bother to introduce in when we think when we studied distributed algorithms, but it becomes relevant here. And what is that measure it is this measure, what I am going to call as delta dash this is the communication we call as degree complexity communication degree. What is that that is the number of messages that the maximum number of messages that any nodes sent or received in one round.

So, if we look at a if you look at this distributed algorithm the bunch of rounds look at any round see what should see the number of messages send by any node, and take this over the maximum or all nodes and all over all rounds. Why is this important, it was not so important the distributor set a why because does not because in one round I can I can send delta messages I can only send maximum delta because that is the it has delta and neighbors what are neighbors degree neighbors in one round that is not a problem. But here that is a problem right because if node has large degree I have to send lot of messages through those different links, so that becomes an issue here.

Because why because lot of edges there in the in the original model, if even if this degree is large I can send let say even if it is degree is n minus one I can send all the n minus one machine in one machines in one round. There is no conjunction; because they are all going through different edges here different edges are mapped can be map to the same link that is going to cause conjunction in this edge in this link. So, I have to look at that complexity also that might be of course, that is maximum how much delta dash cannot be more then delta. It cannot be more than delta, so delta also will come to the picture. It cannot be more than delta the maximum degree of the graph. So, delta dash will not be more than delta the max degree.

Student: (Refer Time: 17:24)

Professor: No, in the original algorithm this is the parameters of the vertex centric algorithm that is so I am just looking at the parameters in the vertex centric algorithm and looking at what is the complexity in the k machine algorithm. But first of all I did not say what is the complexity metric we are interested in and that is quite natural. What is the thing that you want to minimize in this model so number of rounds again, number of rounds so that will be just said. So we want to minimize the number of rounds.

Again one round what is the round, again in one round; I can get some messages, receive compute and send. But of course the thing is there are bandwidth restrictions, so there no bandwidth restrictions are trivial. So, this whole thing is completely trivial, because I can just send all the thing to one machine, because I am not even assuming any memory restriction. So, send all the data to one machine, and locally compute that, so everything I can do in one round.

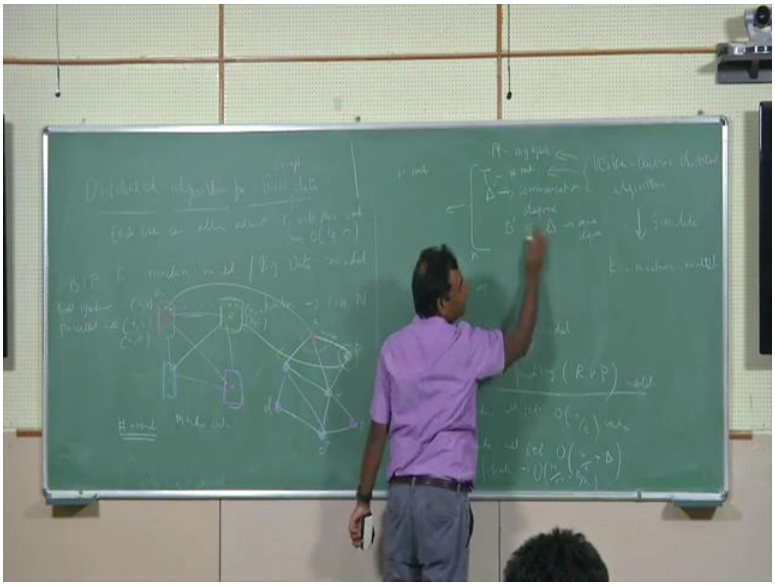
Student: (Refer Time: 18:24)

Professor: Correct, correct, but that is true if you do the random partition, but I am not explicitly assuming any memory thing so memory is not really constraint here of course, if I send that that is very bad because I am going send lot of messages. So, the real constraint is the band width or you can think of that either you can put a constraint on a band width or you can put constraint on the amount of messages and node can receive in one round. So, basically a node can receive only about k messages or k minus 1. So, let us put the constraint that in one just like we had in the vertex centric model the bandwidth this like poly log or log n ok.

So, I can only send log n bits per message, log it is message per round during one in one link. Why log n because at least I have to send the I because that is the log n is the you need at least log n bits to identify a label of the vertex, the n is the size of the input data. So even to send a label I need log n bits and note that this is very this is quite realistic because the bandwidth is much smaller compared to the input data size. So input data size is n the bandwidth is only much smaller poly logarithmic.

So, of course, in general you can do this thing with the bandwidth b can put as a general parameter and that b you will always scale it up so you have to divide by b, but I am not going to divide by log n log n is I am going to just forget about log n factor. So, we forget constant factors I am going to forget log n factors. But in generally, if you want a specific b you can put a parameter b as the bandwidth that means what you can send so you will have a bandwidth that is the crucial thing.

(Refer Slide Time: 20:09)



Each link can allow at most b bits per round. And typically I am going to assume b to b o of log n or poly log n, because I have to send the at least the label t let us just like in the origin model. The motivation is the bandwidth is much smaller that is the whole point that is why that is models the real life scenario. The links are much smaller capacity than the data that it can carry that is what creates the bottleneck so that is why you want to reduce the number of communication rounds.

So, the goal is simple so it is very clean model number of communication rounds. So, if you know parallel computing here it is this is sort of a very simplified version of the bsp model, so the bulk synchronous parallel model. (Refer Time: 21:02) we have seen that called bsp - bulk synchronous. This is the somewhat complicated model. It was introduced by Valiant, he got the Turing award for that, but still, but still it is a very

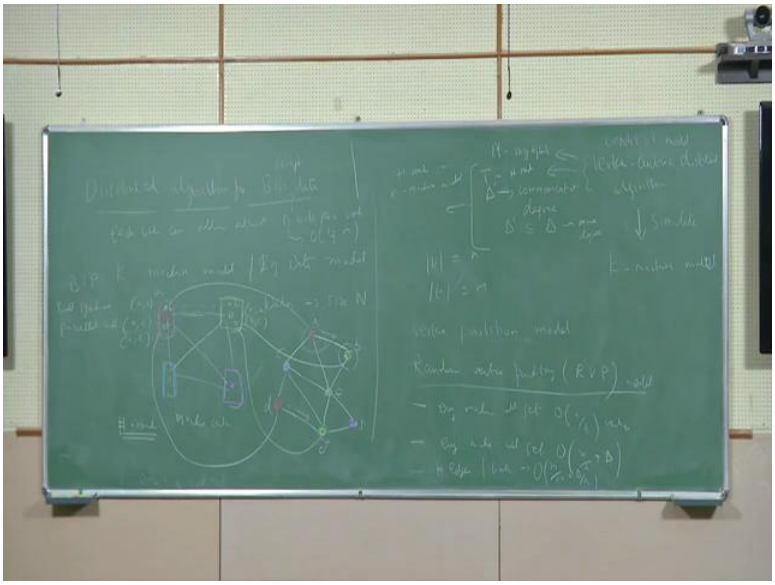
complicated model. So people are not developed a theory for that because there are so many parameters in that it is no real good theory for this unlike here, because it is only one parameter here. One parameter is just k a number of machines.

So, what even understand if the k is the only parameter I want to see so this is like a parallel computing, because I want to see how this the number of rounds scale up with k. Because if I increase the number of machines, I presumably want the number of rounds to be reduced that is parallel computing. So if I want to increase if I increase the number of I throw more machines added then I want to get sparse; that means, the number of round should be reduced. So, I want to really study since you are mathematicians you want really study what is the function, so is the number of rounds growing linearly that is like one by k or quadratically inversely linearly or inversely quadratically or inverse whatever function.

So, what is the function of k that is what you understand, so that will obviously, might depend on the problem, it is that is what you want to study. So, number of rounds is the complexity measure and rounds basically define the same way that we did before. Number of each round is one send, receive, compute sends data and only I can send limited amount of information. So, now, given this I want to see how I can translate this, so these are the complexity measures of your classical vertex centric algorithm.

So, we have studied all this algorithms broadcast, MST, shortest path, all these algorithms are there you can think their distribute algorithms. So, if you had studied centralized algorithms we take a little bit of conversion to interpret that as vertex centric algorithms and that is what pregel does. But if you have already studied distributed algorithms you already interpret with them like that because we already given them distribute algorithm we already thought of them as decentralized distribute algorithms. So, it is very easy to convert them into this model.

(Refer Slide Time: 23:28)



So, I am going to now see what is the complexity in this thing, because it is a number of rounds here there is only one parameter the number rounds. So, what I am asking is if I have an algorithm with this complexity measures in the vertex centric model, what will be the complexity measures in the k machine model that is the question I want the answer. So, this will be a generic thing. So, you take any favorite algorithm or for any favorite problem you just simply the exactly you do the simulation the standard simulation that means, whatever local computation you do, local computation is done simulate by that machine in that round, so that is free in that thing, here also it is free. Whenever message is sent by a vertex in this machine to vertex in that machine it is simulated by sending actually that message.

Note that in the in the original thing I am looking at the congest model right it is a congest model, so it also send log n bits here also I am going to send log n bits. So, this is a log n bit. So, it is obviously, I am going to so it is important that is a log n it is a congest model local means it completely point less. So, here also one message will be in the congest model, so log n bits, here also I have to simulate it by sending there.

But the problem is I might have to simulate multiple things, for example, c might be sorry not c may be may be let us make this red and may be let us make it yeah, so let us

make it red so d will also be assigned here d will also be assigned here. So, let us say in one step of this algorithm a is sending message to b, and d is sending a message to g. So, g is blue so g sorry not g sorry let us make this green so that means, g is assigned to here. So a is here d is also here b is here g is also here.

So, in the vertex centric algorithm may be it might happen that a is sending a message to b, and d is sending a message to g. Both of them have to be simulated across this link, because both these edges are sort of across this link. They are mapped to this link because this simulation a is sending to b, and d sending to g, I have to be have to go through this link. So, there is congestion in this link so if I if I send two many messages there is congestion so that is going to so I will do it in two different rounds now. So, in one round I can only send one message and the next round I have to do it in send the next message.

Student: Log of capital n and that is a (Refer Time: 26:25) log of small n.

Professor: Log of small n.

Student: Here also (Refer Time: 26:28)

Professor: Yeah n, n is a number of the nodes.

Student: Still smaller than log of (Refer Time: 26:33).

Professor: So, what is capital N?

Student: (Refer Time: 26:35).

Professor: Data is size n right, here it is n or m may be. So, data is n so here is the size of data is this small n or here is the m plus n rather, but it is still m is n squared, so log of m squared is also log in that is why I said. But it is not log of k though, it is log of n because the only k machines it is not log k, but log of n. So, now, it is really question

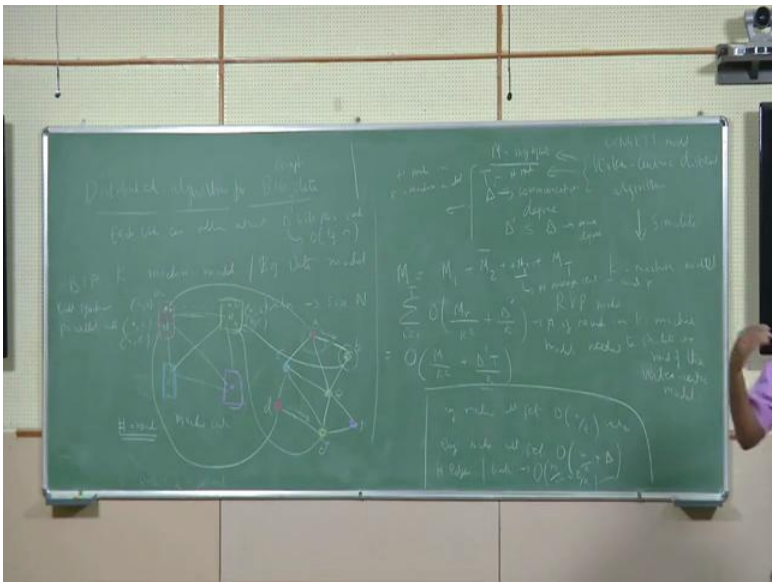
understanding that it is so to simulate it so let us see what is being the complexity so I just simulate in a in this standard way. So, I am going to understand how many rounds it is need so to just know that I have to see how many rounds it takes to simulate one round in the vertex centric model.

So, let us look at one arbitrary round in the vertex centric model. So, in one arbitrary round some messages are going like that it is congest so that means, two messages cannot go in the same as or may be not more than, may be one can go like this; so not more than constant number of messages can go through one edge in one round in the vertex centric algorithm. How many such simulate that how many rounds I need in the k machine model. So, now, think about this now pretend that there is some communication pattern going on in one some arbitrary round r.

Student: Data frame n by k.

Professor: So let us look at.

(Refer Slide Time: 28:17)



So, m is the message complex. Let us write m as let us split m as following. So, let me erase this; let keep this, because it is important. So, assume this n and m is clear right. So

m is the total message complexity right, so let me split this as following, what is let me split this as what is this, this is the messages sent by the algorithm in round one, there are only t rounds this because message complexity is a total messages right over all the rounds.

So, let us focus on some round, so let say some round m r, some round that is the total number of images sent in that round. I want to see what is the time what is the number of rounds needed to simulate that in the k machine model; it might be more than one because there might be this congestion, because here because two things although there is no congestion here, there is congestion here because they are map to the same link.

So, now to analyze this, let us now pretend that the mapping is done now. So you have this random vertex partition model right, so this r v p model, so this is done in the so what is the random vertex partition model that gives you this property every machine has this many number of vertices, and importantly this is a thing every number of edges per link is this. So, you can just pretend that the mapping is done per round although it is not like that, but you can think of it in the unconditional way because I am going to and take a union ( ) or for expectation does not given matter so for expectation what you just look at it and see the mapping is done now.

So, since the mapping so every edges per link is number of edges per link is this; that means, what if there are m edges every link will get m by k squared plus delta by k. But now I am only focusing on these m r messages that can be a sub set of m, it can be m itself, because these are the total number of messages going across the network, it can be m itself.

That means, every edge is used that is means m, but in those in general it can be sub set of that. So, think of these edges being they are all mapped randomly right in particular the edges that are sending the messages they are also mapped randomly. So, how many edges what will be the number of edges mapped number of edges that have the active messages going through them in this round, what will be the mapping through each link, it will be exactly this by instead of instead of m it will be.

Student: M r.

Professor: M r, because those will be the number of message mapped. So, I am going to expectation, when I take expectation it does not matter because I just add them does not matter, because it is the each one of the can, because that these are think of random variables, we can just add them for expectation. So, I am going to take only the average stored. So these many rounds are needed, because so many edges are mapped to one link so I have to take so many rounds in the k machine model because each I can only send one message per time because only law.

Student: (Refer Time: 32:09).

Professor: For just one round this M r is the number of message and round r. So, m r is the number of messages sent in round r.

Student: Was writes whether a bound using delta dash instead of.

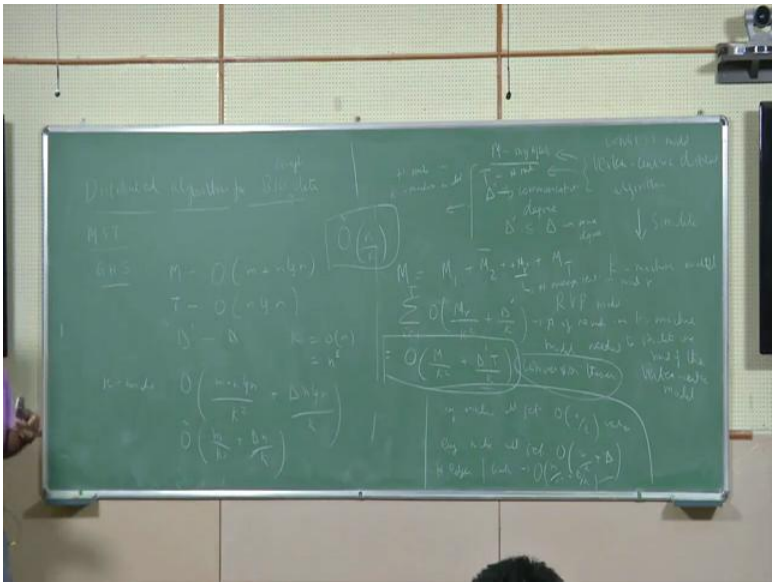
Professor: You are right, you are absolutely right; it should be delta dash perfect, it is delta dash not delta. Because delta that is what I said of course, the worst case can be delta, but it exactly delta dash correct, that is why introduce that. Otherwise I have to just say delta. Yes, so the total this is what this is the number of rounds needed to simulate one round number of rounds in k machine model needed to simulate one round of the vertex centric model. So, what is the total number sorry over all the rounds so this will be what?

Student: (Refer Time: 33:36).

Professor: So this will be I take the sigma inside so it to be sigma 1 by k squared comes out sigma I is equal to one t m r that is m, and then delta dash. So those cut the complexity. So, this gives you a vanilla way of converting an algorithm in the vertex centric model that is what we have studied till now in distributed algorithms to execute in the k machine model. So, now, let us look at a particular problem so and then see what

did give you and can we do slightly better. So, let us look at again our running canonical example the law can take any of your favorite problems graph problems. So, I would say some of the most MST shortest path page rank connectivity things like that.

(Refer Slide Time: 34:53)



But anyway let us strict with the MST, because we started with the MST. So, let say I want to see what is I want an MST algorithm in the k machine model I want to get a complexity for that one algorithm. So, start with, so you can start with any algorithm with these parameters and then convert it.

For example, you can start with GHS algorithm which is a vertex centric algorithm or the pipeline b different things will give you different complexities, because based on their complexities here right whatever their complexities will get translated like using this. So, you can think you can call this as the conversion theorem so this sort of converts that algorithm in a standard way. So, let us look at MST and let us particular look at GHS, we can look at other algorithms as well, but let us look at GHS algorithm first because it is the class the first algorithm we studied.

So, I want to implement this algorithm in the k machine model the big data model to compute MST. What will be its complexity, so this is all I first algorithm implement in

Pregel by Google people. So, what will be the algorithm, so what will be the complexity? So, I have to look at the corresponding complexity in the g h s. So, what is the GHS complexity what will be m for that m will be remember what is the message complexity of GHS. So, it will be m plus n log n, m is the number of edges right. So, time complexity is what n log n and delta dash is what, can be delta, can be delta right. Because if you look at GHS, because it in the very first thing, it looks at all neighbors right mean it can be delta, because at any some near leader might send it broad cast all the nodes, so it can be delta.

So, let us convert this if I use this theorem what will be the conversion. So, this is the in the vertex centric, so in the k machine I get what so I get o of m plus n log n t is, so this is yeah so I am going to just write it like. So, I am going omit log factors, so just make it cleaner. So, o tilde I means going to omit log factors, so then it is clean in a stating it, so it will be m by k square because these log factors basically it go away. M is always at least m is like n it is connected right m is like yeah so all most n log n right. So, you can interpret this based on the parameters if it is parse so definitely it is this thing is at least n by k, but it can be much bigger than n by k right it can be much bigger.

Because for example, if m is dense this will be n squared by k square so k always think of it is think of k as very so k is definitely little o of n; k is much smaller in practice like k is like thousands or machines may be and n is billions. But so in practice you can think of a let us say k is may be even logarithmic n, but at but typically let us say like you can think of it is n to that to the epsilon some small epsilon. You can think of it like that so it is much smaller than the size of the input. So, it is something like so that means if m is n squared that completely dominates this so if m is dense this is n squared by k squared.

Student: That also kind of becomes n square.

Professor: If that is n squared by k. In fact, that is even worst actually right so this actually if we it is dense; that means, if there are some nodes with degree delta is n in this n squared, it is very bad. It is reasonable I mean it gives you a bound, but it can be bad. So, now, let us see whether you can get a better algorithm by doing a slightly different more somewhat more efficient simulation or by sometimes this is the best you

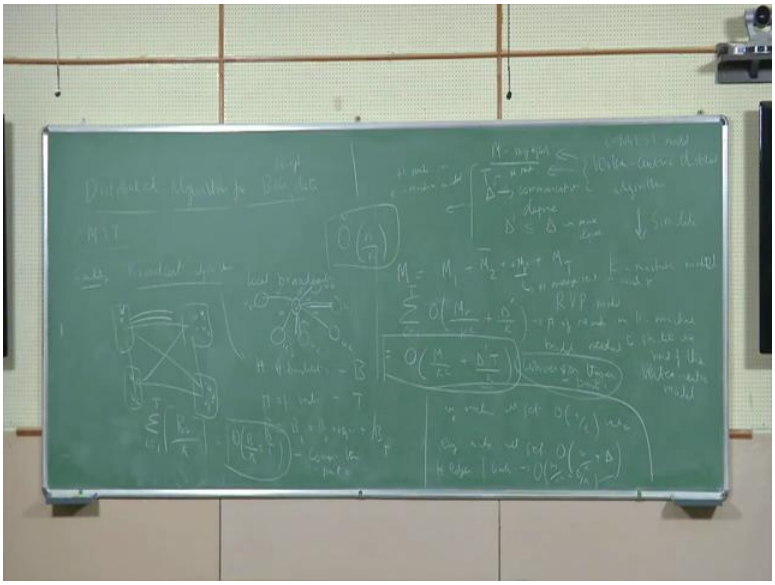
can you can do possible. For example, this is already if let say if delta is small let say delta is bounded and m is not very large, it is parse graphs, then this is already very good if I get n by k squared and delta so n by k. So, this dominates so n by k dominates so if basically you cannot get better than n by k, but it can be much worse than n by k ok.

So, let us see whether you can get somewhat better than this. So, ideally I want to give you an algorithm now that that is very clean that is essentially n by k in all graphs. So, this is actually a linear speed up algorithm. So, how will I do that so you can also try other algorithms other algorithms will be much worse. For example, the GHS the pipeline algorithm can because its complex it can be n square right if you look at message complexity message complexity comes here so it can be very bad. So the g k the d plus square root of algorithm that is also bad here, because although it is d plus square root of n in the vertex centric model does less number of rounds presumably it has lot of messages so again that plays a role.

So, in some sense when it is this means that I should look at algorithms in the congest model that is not only good in one parameter, but in sort of both parameters. So, even if I have very fast algorithm in congest model, but takes lot of messages that is going to take it here in the in the k machine model. So, it is very interesting basically. So, now, let see how I can get so I am claiming that can do something like this you see that it is already not given by this by this vanilla simulation it may not give this.

So, I want to do this regardless of whether the graph is whatever dense or parse or degree or whatever how will I do that so I am going to a slightly different kind of simulation and that already sort of addresses a sort of a drawback very all most the trivial drawback that was there in the previous simulation. So, now, let us look at algorithms in the vertex centric model that where nodes broadcast there are lot of distribute algorithms their nodes broadcast right.

(Refer Slide Time: 42:24)



So, what we mean by a broadcast, here it is I should not say broadcast, so I should rather say local broadcast to contrast with the broadcast problem. Broadcast problem is sending one message to all the nodes that is not what I am trying about here. What is local broadcast; that means, let say I have a node in a round I a node it has local broadcast if it sends the same message, let say alpha to all it is neighbors. Of course, our model are more power, it is actually unique graphs model, it can node can send different messages across different edges in the same thing that is allowed.

But sometimes you it might just two broadcast. If an algorithm during all the rounds does only broadcast then it is a broadcast algorithm; that means, all rounds all nodes are only broadcast that means, it is always sending the same message to all of neighbors. In fact, all wireless algorithms are like that because there is no edge links there. So you can think a wireless algorithms are doing broadcast because they only broadcast everybody who in the nearby can hear that that is only one message sent it, it not send multiple messages, because it will be lost the interference it is only one message.

So, wireless algorithms are naturally broadcast algorithms, but you can here also you can think of broadcast in the sense because we have links I am sending these same message. So, now let us focus on broadcast algorithms alone. It is not when I say broadcast it is

not in that is sense that is means every node sense the same message to all it is neighbors in all rounds; that means, every node sends only one message. Of course, GHS algorithm is it a broadcast algorithm, not in the way I described, because it might be we will see that you can interpret GHS algorithm in a broadcast way and that is what gives you this power basically. So, you can interpret GHS algorithm. I will come to that in a second, but that is what I am going to do.

But now let us kind let us think of simulating broadcast algorithms. I claim that I can simulate broadcast algorithms better than this the simulation that I gave here. Why because so let say what happens the broadcast algorithm so that means, the algorithm in the vertex centric model is only doing broadcast in all rounds all nodes have only doing this broadcast that is means they are sending the same message through all the neighbors I claim that I can simulate it efficiently.

So what happens? Let say this is v, it sending messages to all its neighbors let say u 1, say it has many neighbors u 1, u 2, u 3, u 4 may be u 5, u 6, it is only sending alpha to everybody. So, what happens look at the k machine model u is assigned to some node some machine let say v is assigned to some other machine of course, if they are both assigned to a same machine that is easy.

Student: Only send you can only send n by k messages on that.

Professor: To simulate one round of broadcast for one vertex, how many rounds going it?

Student: Just one.

Professor: One, but the in the conversion theorem that I did I did not exploit that. What I mean is this write. So let say so not u v I should say u one so these are the neighbors right so u is assigned sorry v is assigned to some machine this is v. The neighbors are assigned randomly. So they are just assigned randomly so may this machine got two neighbors, this got may be two neighbors, this got two neighbors. So, what happens in the original simulation, the original simulation it simulates you send two messages, but there is same messages. You will send two messages here because v sends two messages

u 1 and u 2; there both the messages are sent to this link. So, I will be counting them as two messages, but you can do better simulation. You can just send only one message and then ask that machine to locally deliver it to u 1 and u 2.

So, it basically simulates n by k messages in only one round. So you have to assume is little bit more for this right because when you deliver some message, it cannot send deliver it to u 1, u 2 that is that is lot of information that is so like that is bad because I am actually cannot list the vertices. When I send v to I cannot say send and v send message to u 1, u 2, I cannot list them because that is like sending two different messages I will just send v and I will just send message alpha.

Student: Would I want to take it.

Professor: Correct this machine can know the reverse index right it knows that u 1 v is the neighbor of u 1 u 2; it should be delivered to that because we can easily have the reverse index. We can already calculate it and keep it there. So, when it messages this is the message from v it knows that u 1 and u 2 are the targets, because it knows that this is they are the neighbors of. So, I send only one message. So I say so one broadcast can be simulated not in n by k rounds, but in one round. So, now, let us look at broadcast algorithms. So, what is the parameter in broadcast algorithms there is only two parameters.

Student: (Refer Time: 48:13)

Professor: Number of broadcast. Let us look at it. So, you can also look at that, but yeah the time. The two parameters one is the number of broadcast like the three parameters there is the number of broadcast. What is number of broadcast is the total number of broadcast done by all the nodes over all the rounds, because every operation is only broadcast.

So, is total number of broadcast in one round by all the nodes sum up over all the rounds that is the total number of broadcast, let us call it b, that is the parameter and number of rounds of course let us call it t here. There are only two parameters are broadcast

algorithms. The total number of broadcast over the course of the algorithm, and the number of rounds in the vertex this is I am looking at the vertex centric model. Suppose, I want to simulate it in this fashion slightly more cleaver fashion so what will be the complexity in the k machine model.

Student: (Refer Time: 49:18)

Professor: No, way well yeah b is clear, but you have what so again look at in the in the random vertex partition model. So, you have n nodes they are randomly partitioned. So, if let us again look at it look at one round. So, let us again look at b, b you can partition into b 1 plus b 2 plus b t that means, b 1 or where b r generically b r is the number of broadcast in round r. So, this is total number of broadcast in round r by all the nodes, but the random partition again you can argue by expectation that they are randomly distributed; that means, each machine will get how many will be the doing how many simulating how many broadcast.

Student: 1 k.

Professor: b r by k, so how many rounds it will take b r over k. Because to simulate one broadcast I need one round because it has to send to all the links right because send it to all the machines. So, b r by k is the average, but you can also show that is come maximum because of the load balancing one average is b r by k, but it can also may be it is 2 b r by k this is.

Student: Di-probability.

Professor: Di-probability so, but b r by k is the number of rounds to simulate 1, 1 round that took b r broadcast. Now I have to sum over so I have to be little care, so I have to say like this right, I does not write. So, this is what so this b r by k plus that is should be at least one I mean this cannot be smaller right, so this cannot be a fraction so there are no fractional rounds. So that is why in general it can be b by k plus t because it has to be at least one broadcast in one round.

So if I have one round broadcast I cannot get one by k that is not possible I mean I have to simulate at least one I have to spend that time so b by k plus t. So, this is some an alternate version of the conversion theorem that only applies to this is you can say conversion theorem part one that applies to any algorithm that it means applies to even uni-cast algorithms this only applies to broadcast algorithms.

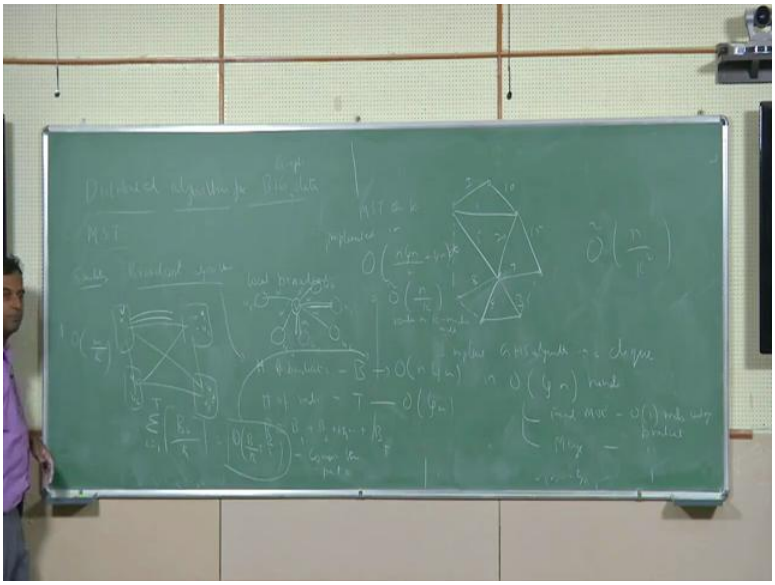
Student: (Refer Time: 52:08)

Professor: This is the number of rounds b by k plus t rounds, t is a number of rounds, yeah so t is a number of rounds in the in the original algorithm, b is a number of broadcast that translates to b by k plus t rounds in the still t is there, still t is there. So, this generally assumes that b is not very large; when b is very large I can do that. So, now, let me try to solve MST by using a broadcast algorithm.

How will you do that again I am going to simulate GHS algorithm, but GHS algorithm is little bit tricky they are not it is not really a broadcast algorithm I did not say because it is not that broadcast every round every node should be the same message. So, the way we described if we think about it is not, it is not a really a broadcast algorithm, but I can think of it I can implement it as a broadcast algorithm in the following fashion.

So let see this, this one of the question this is sort of later one of the final exam questions. The key is thinking about GHS not in the way we used to think like before like GHS is the operating in a network, but you can think of the GHS algorithm operating on a clique I means I have this graph. So what is the graph?

(Refer Slide Time: 53:28)



So, I still this is my input graph right may be this is some input graph, so this is my input graph I am find the MST, they are some weights I did not put the weights, but there are weights so let me put the weights I think let us put the weights. There are some weights. I am going to simulate the GHS algorithm, but it is what I am claim what I am going to view the GHS algorithm is a broadcast algorithm; I am going to first pretend that the GHS algorithm is actually happening on a clique.

So, I can think of these edges this node this graph is clique for the purpose of implementing the algorithm. Why because when I am trying to do the simulation here this algorithm is mapped to the network. So, every node every node can communicate with the every other node. Even they are so even this node can communicate with this node that does not happen in the original GHS algorithm if we think of it is a vertex because there is no edge between them, but here I can right.

So, I can put an edge here within so I can completely I can put the complete graph clique, but of course, I have to put a weight here, what weight should put? Infinity, so it is I can use it for communication it is not a MSTH, it cannot be a MSTH. So, I can pretend that my graph is actually a clique where non edges are infinity weight, but they can use it for communication why because I can use it for communication in the k

machine model.

But if you purely think about it is a vertex centric algorithm it is not right, we did not that is why, we did not put that when we have a network I did not put in I did not links across all edges because vertex centric thing it is not the case, but here I can. Now I claim that and this is actually think I can imply now I am going to study how long it will take to implement the GHS algorithm. What is the GHS algorithm if you remember it takes in phases so there is finding m o e and merge, finding m o e merge how many such phases will be there.

Student: log n.

Professor: Log n phases, because we showed that in every phase the number of components is going to be reduced by factor of two so that is one of the problems in the final. So, I ask you to show in the final that you can finish if it is a clique I can finish the GHS algorithm in the log n rounds. So, you can implement why because a diameter is one the diameter is one, so what is broadcast, so this is a thing. So, you can implement GHS algorithm in a clique, it is a clique here because I made it a clique. And more importantly the algorithm is a broadcast algorithm I can always that this implementation will be broadcast it means in every round I will always only broadcast.

Why? Because I can show that each phase of the GHS algorithm which is two sub phase algorithm first phase finding (Refer Time: 56:51) that I can do that by broadcast. Finding merging them I can also do by broadcasting, you have to think a little bit about that that is basically your final exam question, how to implement the both of them. So, finding (Refer Time: 57:07) order n broadcast finding merge merging the components broadcast that is means all the nodes will send the same message. It is might be possible that some node may not send any messages silence, but that is, but that is like sending a dummy messages send a dummy message. Some nodes may be sending nothing so we just send a dummy message. So, how many broadcast will be there so there are order log n phases.

Student: (Refer Time: 57:42)

Professor: No, number of broadcast b, yeah so time is log n, because there are only log n rounds, and that is also one thing. So, I am going to use so that is what I will do broadcast so using broadcast I am going to implement each phase in actually constant number of rounds like two rounds or three rounds constant number of rounds. So the log n rounds that is what this means that each phase that means, find m o e constant number of rounds using broadcast find merge also same.

So, both the steps of each phase I can implement in constant number of rounds that is why I get totally overall log n rounds, because there are only log n phases this is what you have to think not difficult, you have to think. How many broadcast will be there since I am doing it is so fast that means, each phase constant number of rounds, how many broadcast would be there of course, there cannot be more than.

Student: (Refer Time: 58:59)

Professor: n because only n nodes, so it there can only be n so total number of broadcast will be yeah n log n, because n broadcast per phase order n broadcast and log n. So, the total number of broadcast would be order n log n. So, if I use this here what if I use this part of the conversion, what will be the complexity of implementing the k machine module. We will be if I use this with this then I will get I just showed you that I can do it in a n by k regardless of regardless of the graph parameters. But this one I did not use that is basically that is what I am asking you how to do if we if we have a clique so graph is a clique, you can solve the MST problem in log n rounds because the diameter are so small. So, in generic it is a n log n rounds that is because the diameter so large, but here is the diameter one. So, clique, but still you are doing it carefully because you have to do broadcast.

So, it is much more there is most sophisticated algorithm that gives you a n by k square that I am not going do should come to my lecture seminar tomorrow I will see that so that is much more sophisticated algorithm. That so actually that is randomized this is not really randomized GHS is the deterministic algorithm so you can do. So, the best thing is you can do is n by k square, and that is the best also. In general, you cannot do better than this because you can show lower bound, why because you can show that any MST

problem, you need to exchange basic essentially order n messages.

This is you can show by communication complexity, but it is also intuitively clear I mean here n nodes you need to every node should send at least one message. So, n messages is like sort from must and since there are how many links are there the sort of the maximum amount of parallelism is given by the links, because every link you can send messages in parallel there are only about k squared links. So essentially you cannot get better than n by k squared. And you can get that, but that is much better than this that is like a quadratic speed up so that is the best possible algorithm we get so that is much most sophisticated, so that requires very careful load balancing in it.

So, any questions, so any questions on the exam or whatever, I mean you have questions on that? So basically algorithms in this model I mean they clearly outperform map reduce algorithms. And the reason it for that if I am not moving too many data, but if the graph that is there is all is all there note that in this model what is the naive algorithm, so there is a naive base line you can solve any problem let us say for example, a graph problem in how many rounds. So, the bounds that we see are nontrivial.

So, what is the naive algorithm in this model? I said before, I can collect all the information in one machine, and so all that local because locally I am not charging for it right so there are basically m information m edges are there. So, I can send all the m edges to one machine and then solve so m by k is trivial lower bound, trivial upper bound. You can see that that is why I want to that is why here this is still nontrivial add m by k squared so that is still nontrivial.