Parallel Algorithms
Prof. Phalguni Gupta
Department of Computer Science and Technology
Indian Institute of Technology, Kanpur

Lecture - 22

(Refer Slide Time: 00:17)

CS646
Case Study: Presentations

(Refer Slide Time: 00:19)

Case Study: Parallel Algorithms

- Group [A]
  Parallel Clustering Algorithms
  (Hierarchical Clustering)
- Group [B]
  Parallelization FFT
- Group [C]
  Sorting Network

(Refer Slide Time: 00:22)

Case Study: Parallel Algorithms

- Group [D]
  Parallel Approach to compute DNA sequence Alignment.
- Group [E]
  Parallel Data Structure
- Group [F]
  Parallel Algorithm for computing Voronai Diagram.

(Refer Slide Time: 00:25)

Case Study: Parallel Algorithms

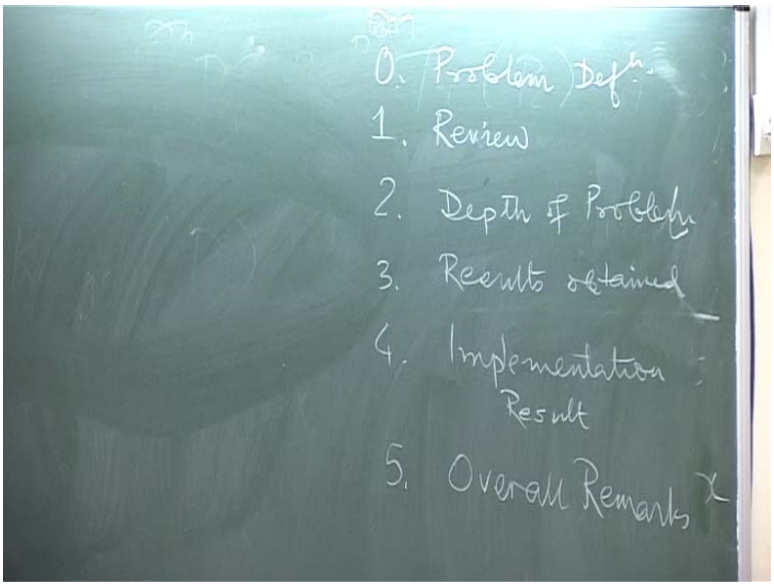
- Group [G]
  Importance of Parallel algorithms using PVM.
- Group [H]
  Parallel algorithm for K-Sorting
- Group [I]
  Parallel Complexity of evaluating game tree.

(Refer Slide Time: 00:28)

Case Study: Parallel Algorithms

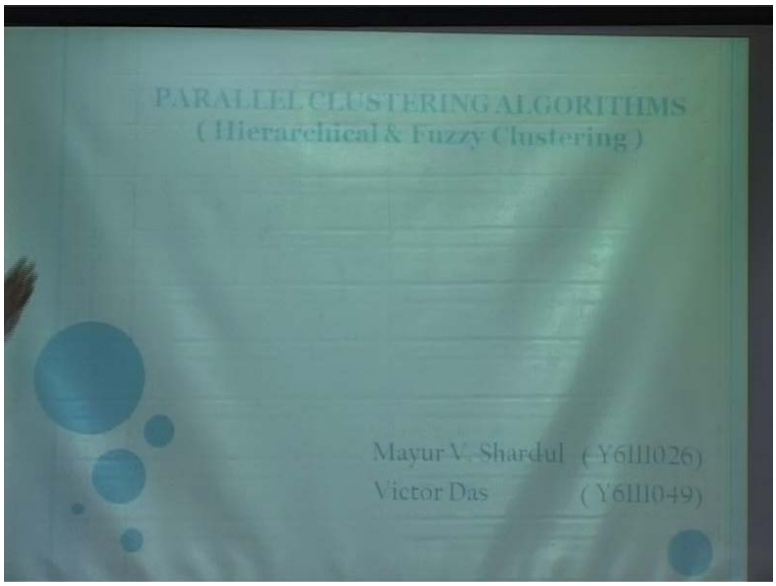
- Group [J]
  K Sorting
- Group [K]
  Parallelization of sparse BLAS.

(Refer Slide Time: 00:30)



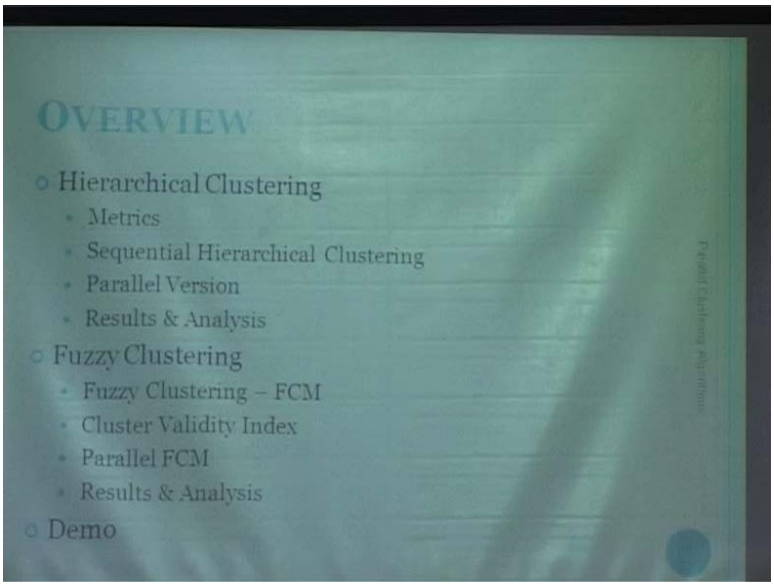
This to be seen on a review Depth of problem, Problem definition, Results obtained, Implementation result all for the own problem or problem been represented. Now, note what you order for you problems others will give comment overall remarks for now if implementation is not there right and note down somewhere this points otherwise tomorrow again.

(Refer Slide Time: 02:02)



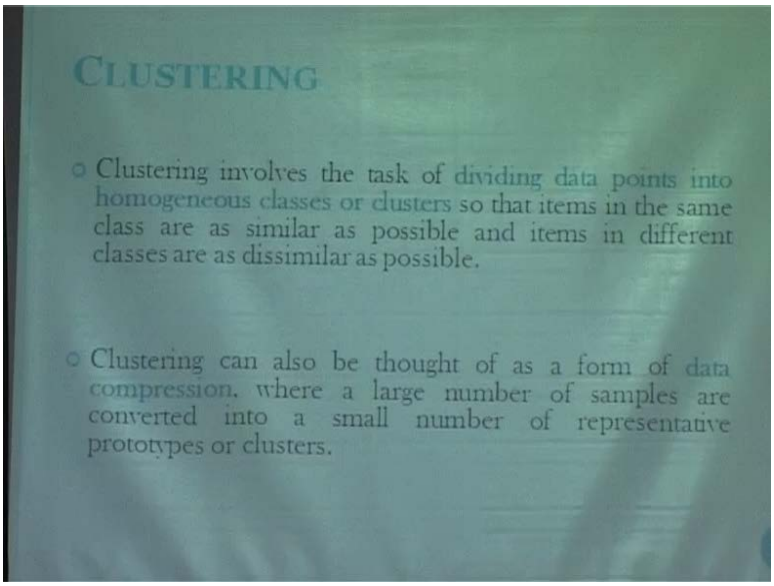
Today will be presenting Parallel Clustering algorithms, it will Hierarchical and Fuzzy clustering both its implementation.

(Refer Slide Time: 02:11)



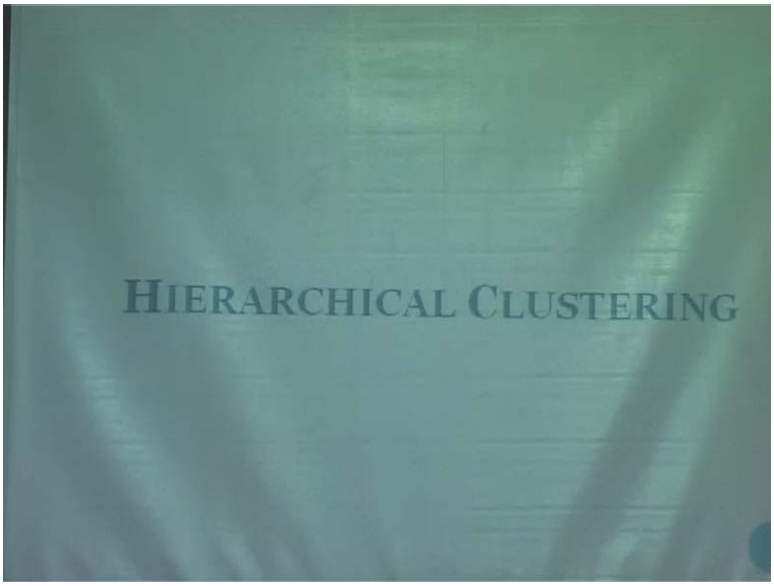
So, before going into details let me give you an overview what the presentation will be. First we will deal with Hierarchical clustering, the define matrix, then the sequential version, we will start up with the main sequential version, then the efficient sequential one, then the parallel version and result and analysis and similar for the Fuzzy clustering and we will end up with the demo.

(Refer Slide Time: 02:34)



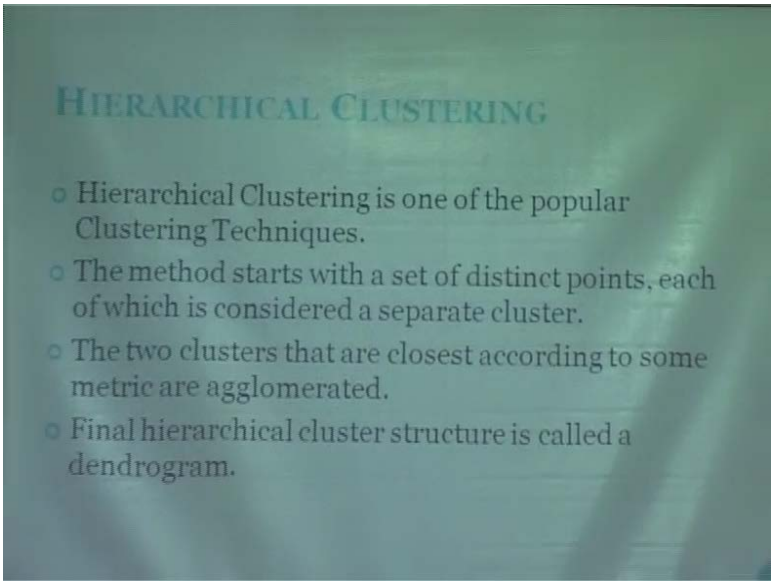
So, before going into Hierarchical clustering and Fuzzy clustering, first we must have a idea of what clustering is. So, clustering mainly divides a set of data points into groups and it divides in such a way, such that every group have similar types of elements. Whenever you compare the elements from different group there class is dissimilar. It can also be viewed has a data compression because now every cluster can be viewed has a single data point because the other point is similar.

(Refer Slide Time: 03:11)



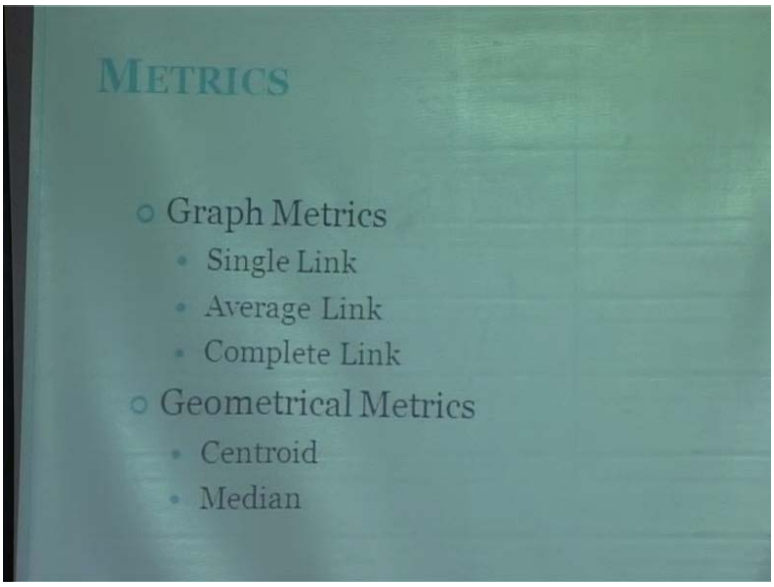
So, we saw one of the important clustering technique is the Hierarchical clustering.

(Refer Slide Time: 03:17)



To begin with we start taking all the end points cluster of single elements and we actively merge two closest clusters to each other. So, evintially we combine all points of crystals that is called has dendogram.

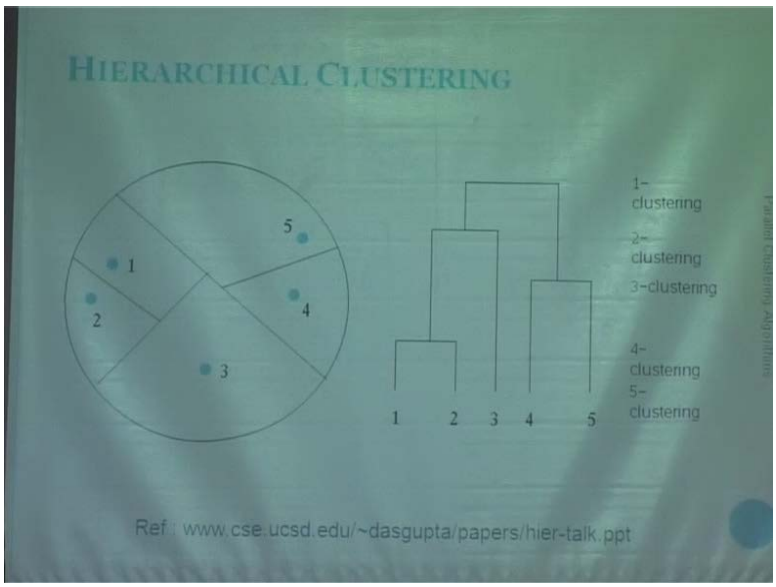
(Refer Slide Time: 03:42)



So, when considering this distance between this two clusters so they are many ways to define it. In general the classification is Graph metrics and Geometrical metrics. Graph metrics is technique has single link, for example single link defines the distance between

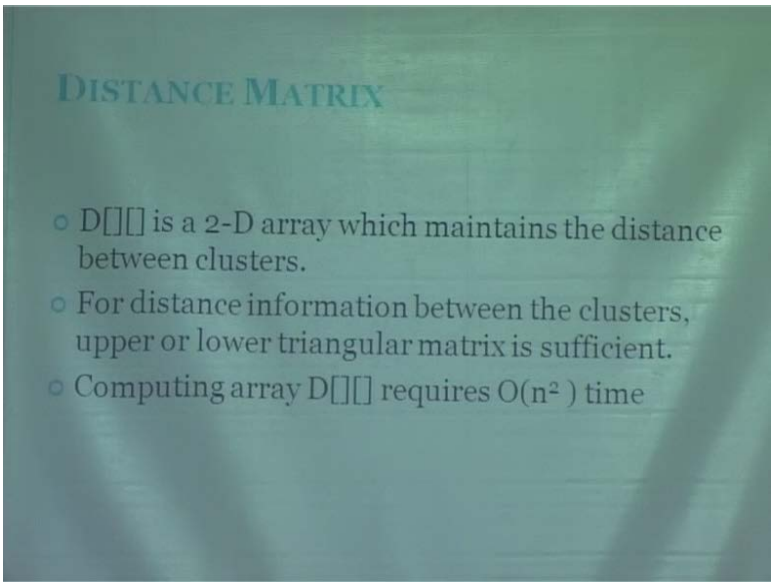
pair of points each point belonging to one cluster and the distance is minimum. Similarly, average of distance between all the pairs completely with the maximum (( )) that the one point is in one cluster and other is in another. Geometrical metrics centroid is set off all the points in the cluster, so you may measure distance from centroid. And median is when merging two clusters we take median of two previous cluster as the center of the new cluster

(Refer Slide Time: 04:40)



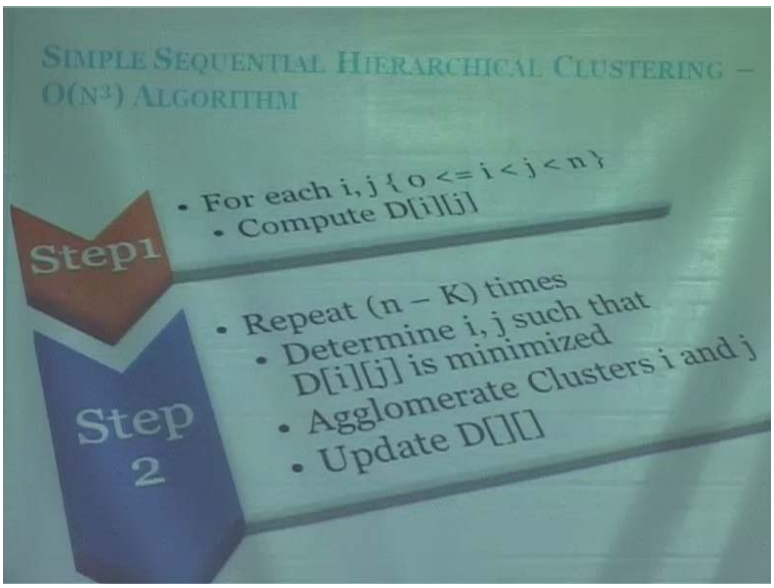
So, just an example is there we have five data points and initially we start up with every data point in a single cluster and then we also shown you the dendogram here. So, it initially at the first level it was seen that the cluster 1 and cluster 2 both contains only one points they are closest to each other. So, at the first level we are combining this 2 and at the second level after combining 4 and 5 are the next closest and at this level second level we are combining this 4 and 5 and then iteratively we do this. Now, after doing this we need k number of clusters so we have to cut this tree into at a certain level. Now, if we cut this at this level we get 5 clusters, if we cut at this level we get 4 clusters with only 2 and similarly, if we do it at the top level we have only 1 cluster.

(Refer Slide Time: 05:41)



So, we will have to understand what the delta structure is, so first only thing we made here is distance between the every data points. So, for doing this we need two dimensional array which maintains the distance between all the data points. For doing this view we need the upper or lower triangular matrix because the other half will be the copy of the same thing and the computing this array requires order n square times.

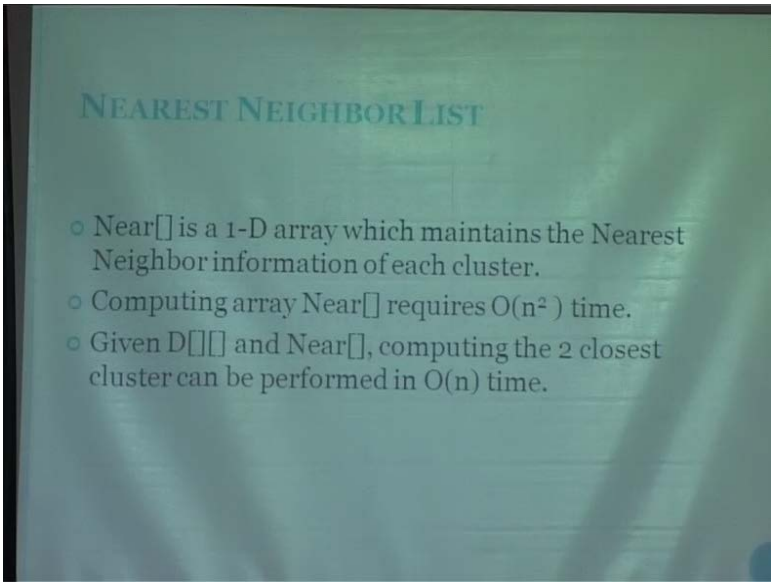
(Refer Slide Time: 06:14)



So, this is the basic algorithm and we compute the first step with the matrix D, i, j, this will take n square time and we fix number of cluster primarily k and do iteration n minus

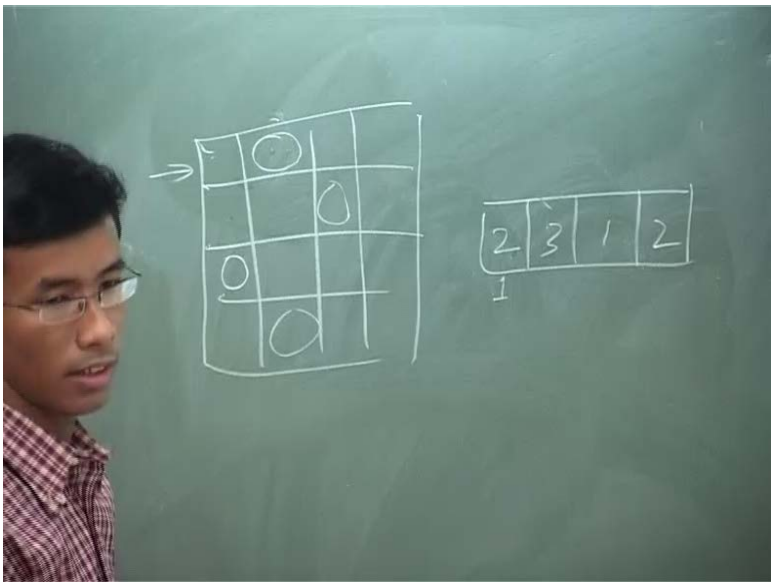
k times to obtain k clusters. So, in each iteration we obtain two closes clusters we merge them and update the distance matrix. This part takes n square time and again finding a minimum we have to find all the scan all the elements of D. So, this also this also requires a lot of n square time and again updating requires other n square times, so overall this step takes other n cube times.

(Refer Slide Time: 07:00)



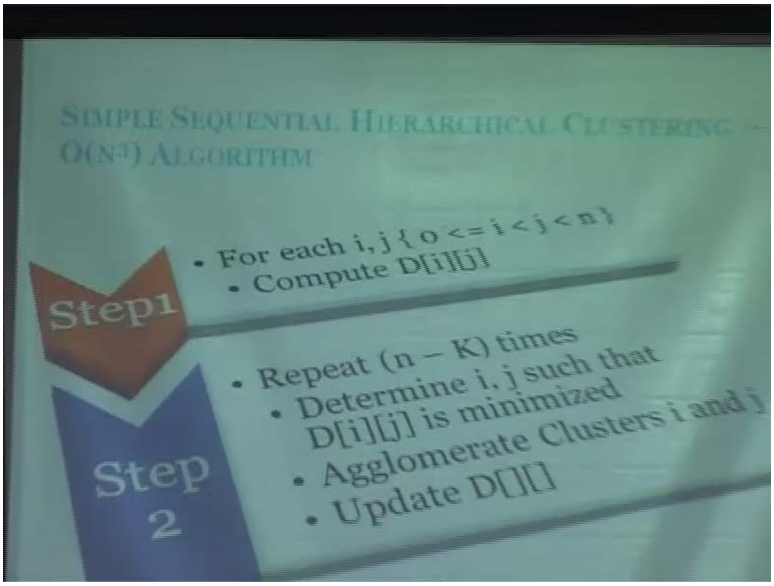
This can be the additional data cluster we maintain a list of nearest element, so for each element we maintain a element or a cluster closely to that. So, to begin with is the nearest end and near here is the index of point to (( )). So, computing near will take n square time and using near we can obtain the two nearest clusters in order n time, because we have to find the pair closest to each other. So, obviously this will be one of this because this is for each point the closest point is there in the near array. So, the closest can be found in near array in other random.

(Refer Slide Time: 08:00)



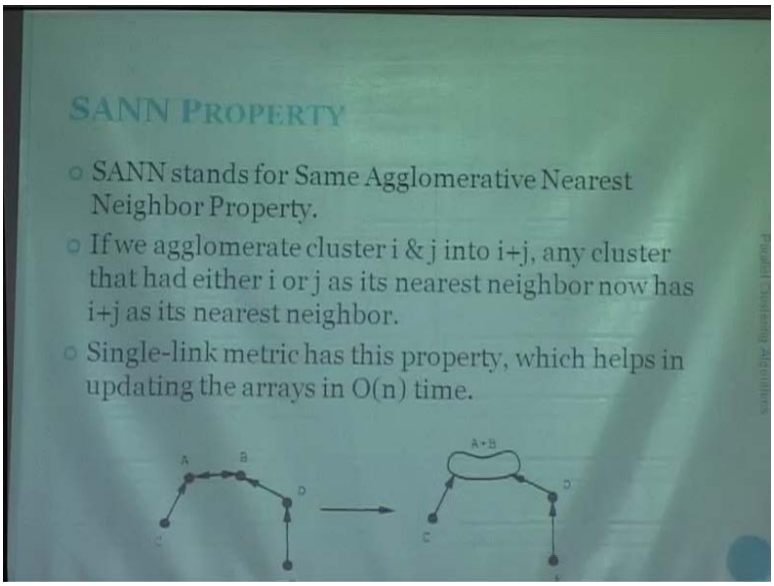
So, if this the D array and we have the near array now so after having the near array, initially we are doing all this scan to find the minimum. To find the minimum we were looking all the value of D, now since we know that the 1 cluster near is second. So, we will only look into 2 so in the entire row this is the minimum, in every row we will look at the minimum value, this is 2 3 say 1 and 2, so we will look only into this values. So, this comes out to be order n now finding the minimum.

(Refer Slide Time: 08:51)



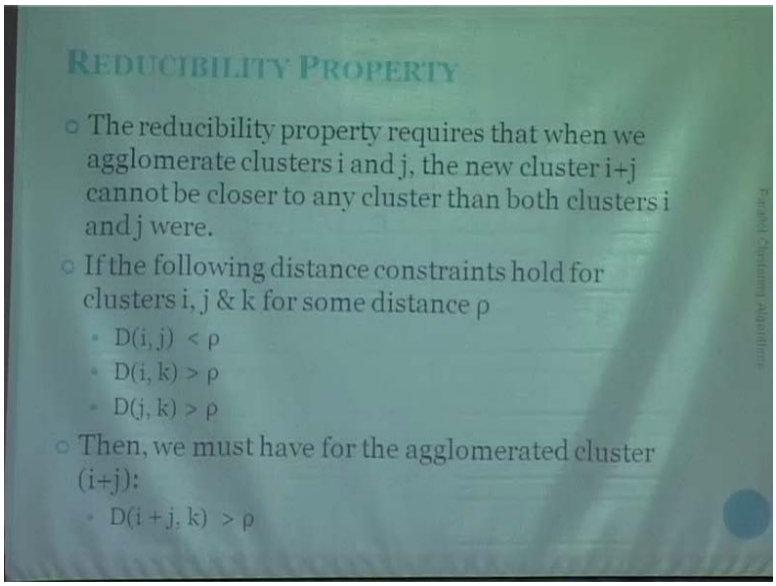
In this part this will be done in order n time.

(Refer Slide Time: 08:58)



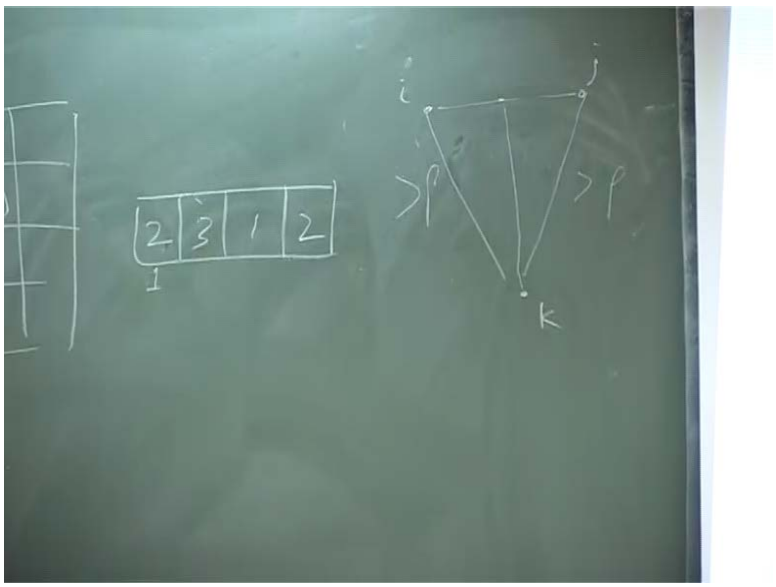
Now, this is the SANN property, so SANN stands for the Same Agglomerative Nearest Neighbour. It means that whenever we looking into clusters A and B. So, say we are marging A and B now the nearest neighbour of C and D change to the new. If say for C the nearest neighbour was A, now it will be the combined one, for d also it will be the same combined one and for others it remains the same. So, updating the near (( )) whatever clusters we are merging we will just look into it is equal to A or B if it is equal to any of the combined one.

(Refer Slide Time: 09:59)



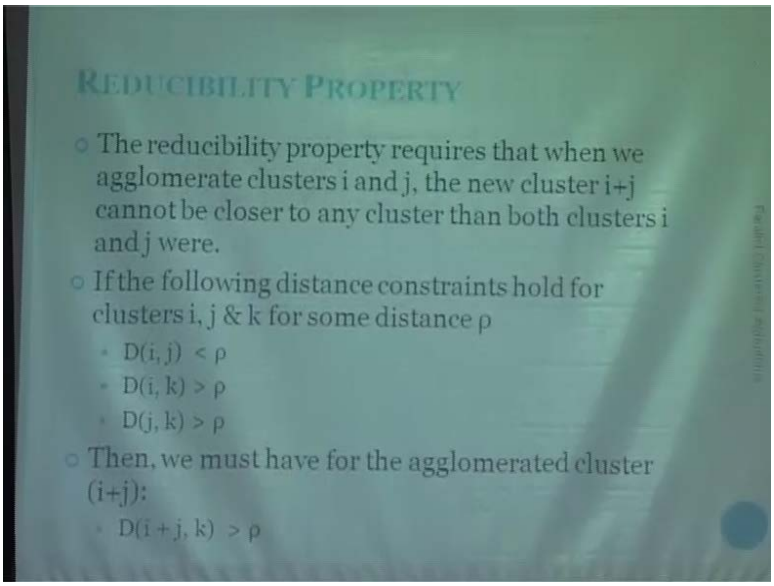
Then we have the reducibility property, the reducibility property is that if this are the distances and with some distance say row and with merging i and j. So, the distance between the combine one and the other cluster k will be greater than row.

(Refer Slide Time: 10:20)



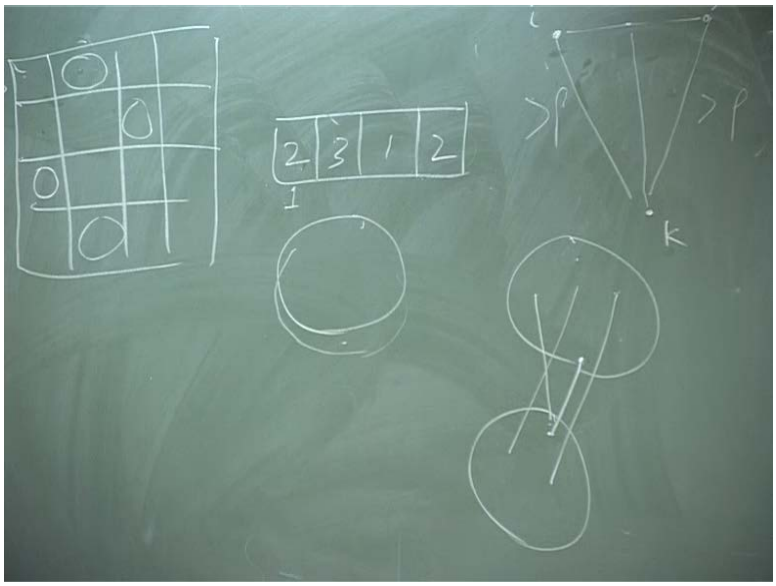
So, if we are taking centroid has the minimum matrix say this are the point merging say this is the k point this is i, this is j. So, after merging this, the centroid will come up somewhere here, if distance is greater than row this is greater than row. So, this may be less than row.

(Refer Slide Time: 10:52)



But since we are using this single link matrix so this format is also there, because we are consider only between points.

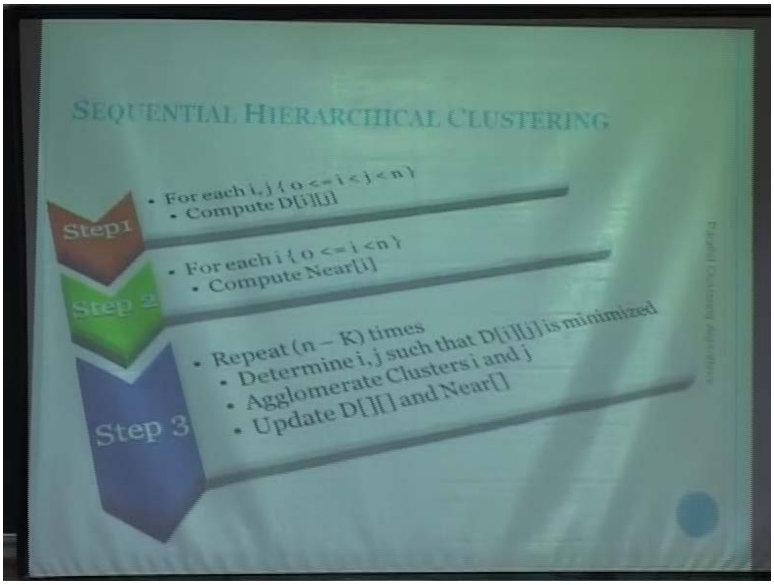
(Refer Slide Time: 11:02)



These are the clusters and we have another cluster this, so the distance between we are taking all the points, distance between pair of points, this points is this distance every possible combinations and we are taking the minimum. So, it will come out something like this, if you considering this matrix this property holds the single link. Nobody knows what happens the distance between this point and that point you verified. Yes sir using

single link we are considering, this is the point and this is the whole cluster, then in the next level we will be merging this, even if the other point is far away from it. In this case we are not making any mistake I do not know let us see.

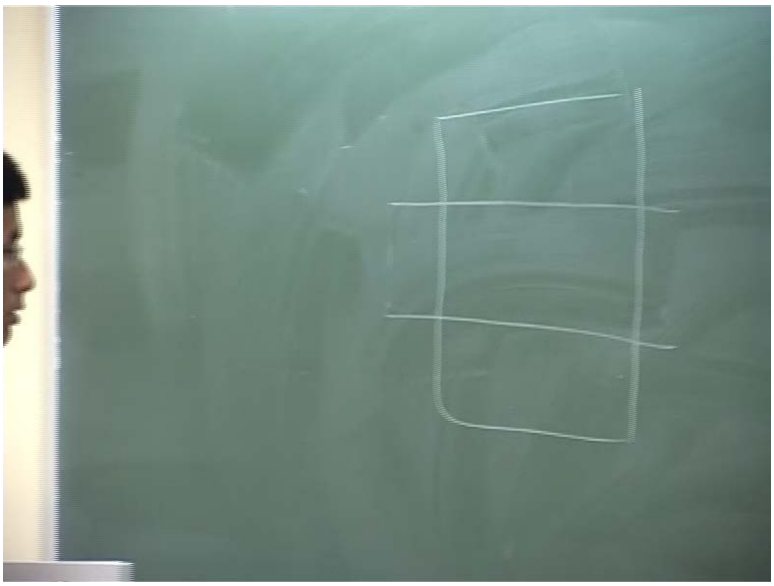
(Refer Slide Time: 11:56)



So, this is an improved sequence algorithm so we use those properties to update near and distance matrix, so previously this two steps were n square now they are n, so overall complexity n square. So, first step is computing D, i, j, now computation of D, i, j, is independent operation, so can be divided at a processor. So, we divide n elements into P groups, where P is the number of processor and each processor will calculate its set of matrix.

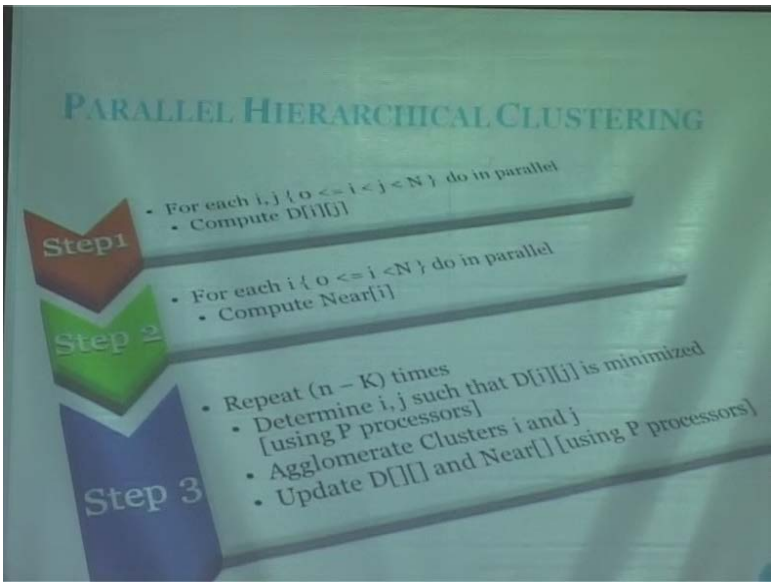
So, this will take n square by P time, then we will compute near, again here we can divide the data at the processor and again this will reduce the factor of P. So, again it is in the sequential order n and again it is an independent operation we divide across the P processors, this gets reduced with P and same with the case with updating D and P after merging and near after merging so the complexity is now n square by P overall.

(Refer Slide Time: 13:19)



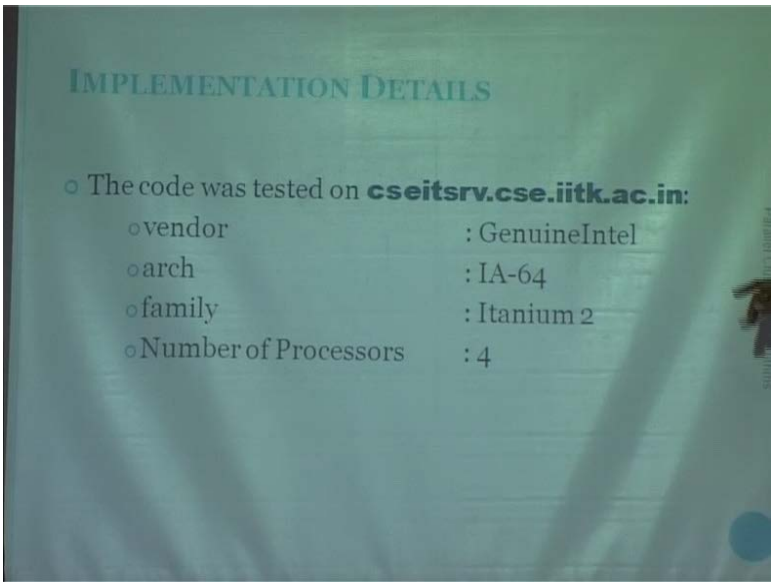
So, actually the every matrix is divided into this row ones, block row and every processor computes its own part.

(Refer Slide Time: 13:40)



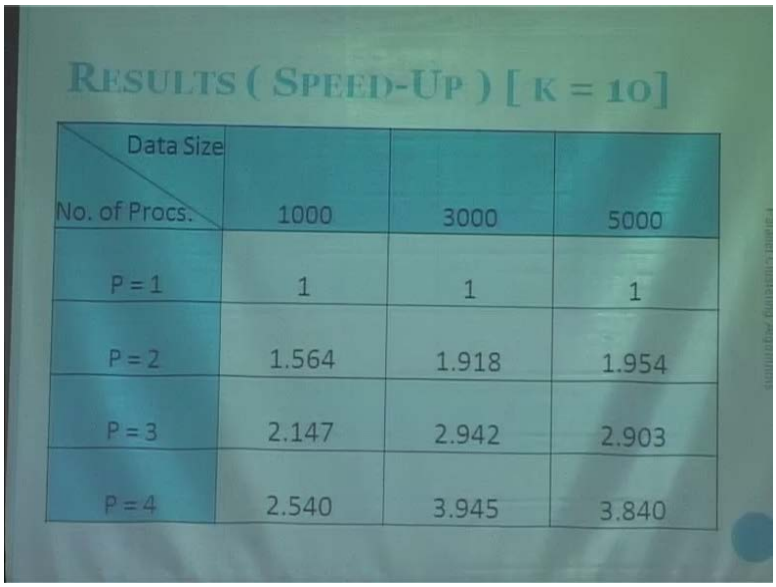
So, in that case you are assuming that the implement yes sir and all the things are available in the program memory right and it is there any possibility, how concurring read is there right, concurring read is there.

(Refer Slide Time: 13:55)



So, we have implemented on a code processor machine with itanium 64 processor.

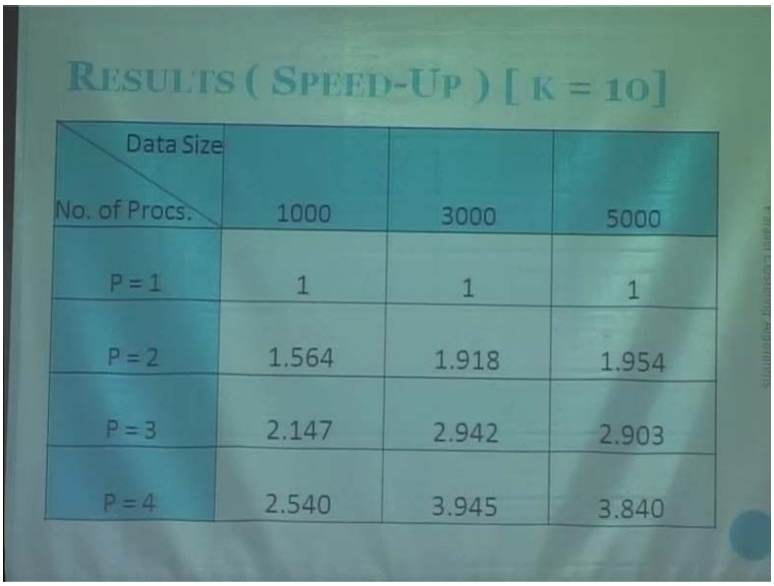
(Refer Slide Time: 14:03)



Data Size / No. of Procs.	1000	3000	5000
P = 1	1	1	1
P = 2	1.564	1.918	1.954
P = 3	2.147	2.942	2.903
P = 4	2.540	3.945	3.840

So, this was the result means you see 1000 data sheet size we are getting this and whenever the data size is increased the data is getting around the range of P with 5000 we were getting around 3.840 processor. So, for the larger data size it is better to paralyze, because the results here are not so good because this number, is an integer, no sir we are using floating point and it is the random numbers and another thing is that the core processor will gain by a factor of 4 so speed up is what?

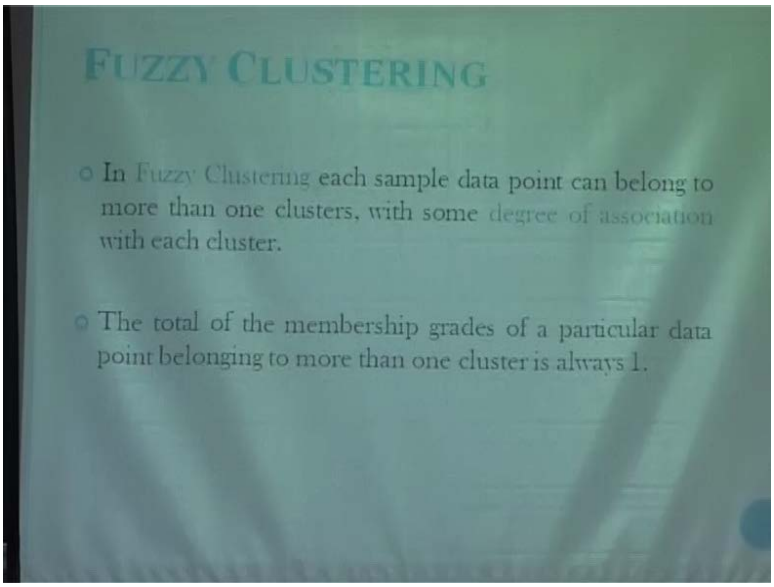
(Refer Slide Time: 14:48)



Data Size / No. of Procs.	1000	3000	5000
P = 1	1	1	1
P = 2	1.564	1.918	1.954
P = 3	2.147	2.942	2.903
P = 4	2.540	3.945	3.840

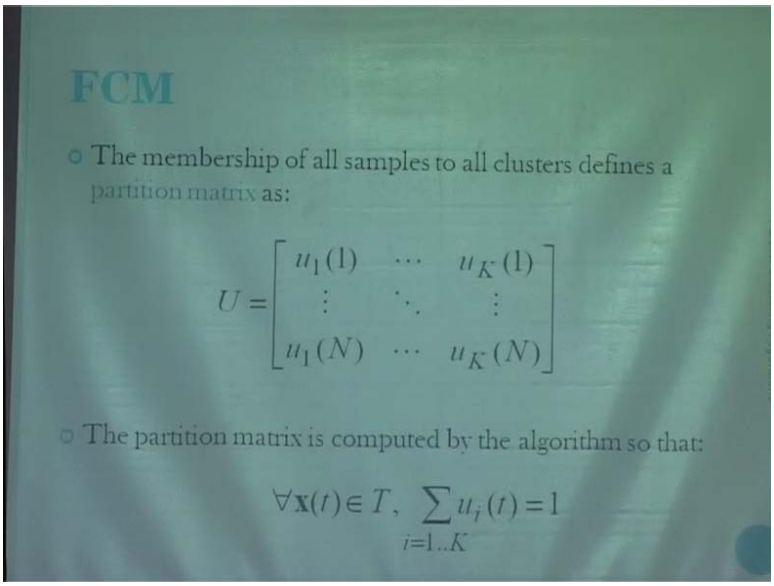
So, this is the speed up sequential time.

(Refer Slide Time: 14:55)



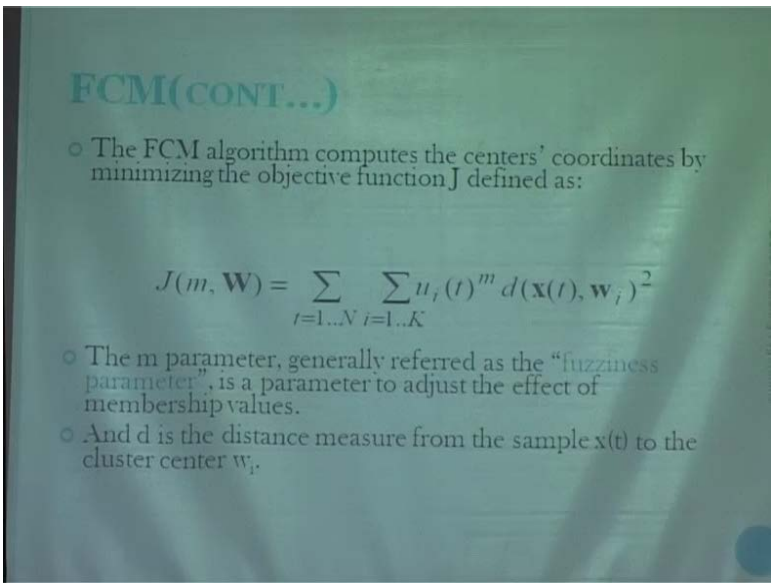
The next clustering technique is the Fuzzy clustering, in the Hierarchical clustering either a members belongs to a given cluster or not. So, here there is a notion of membership to each cluster and the membership varies in degree. So, it belongs to all the clustering that we want, but it is fractionally belong to the cluster such that some of the membership of every cluster is 1 that is which can be normalized way.

(Refer Slide Time: 15:32)



So, we have n points so we maintain n by k matrix. So, for each point i j is the value for the membership for a particular point to that particular cluster, since sum of the membership will be 1.

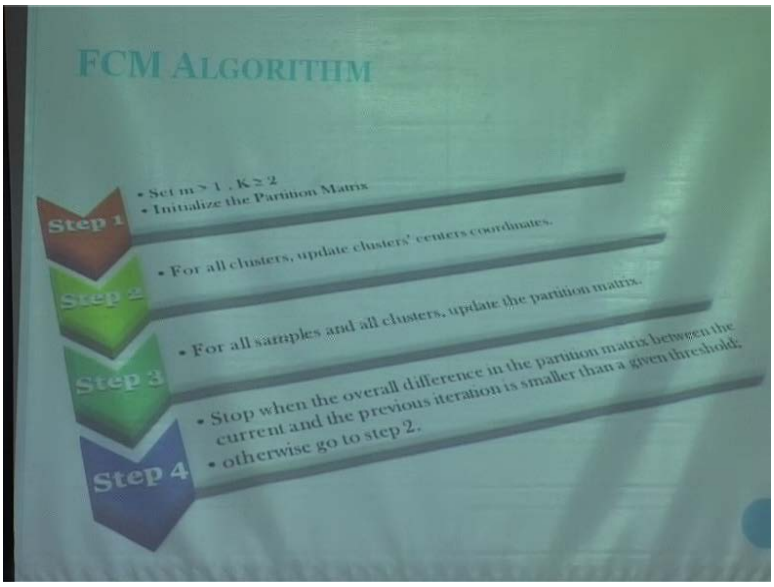
(Refer Slide Time: 15:54)



This is the objective function mainly we are trying to minimize this, some of the distances when we are considering the distance with the corresponding cluster to which it belongs. So, this is the membership degree and with the membership degree, changing the membership degree actually we can actually change the fuzziness to the crys cluster.

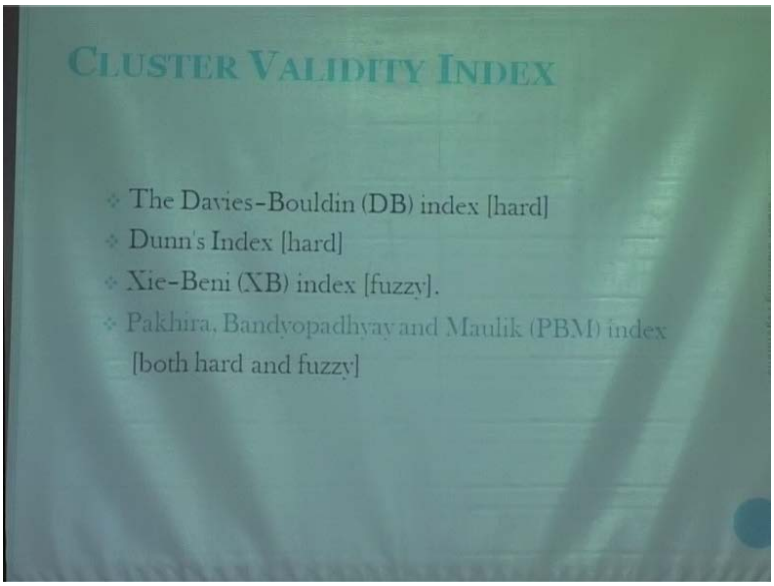
This is the iterative formula from which the cluster centre can be calculated from the value of partition matrix and vice versa, from the partition matrix to w i.

(Refer Slide Time: 16:30)



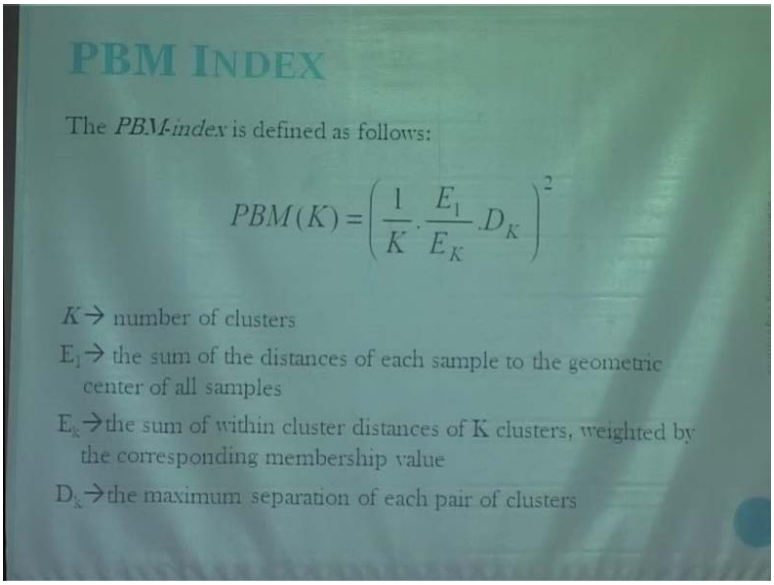
So, this is the basic sequential algorithm for FCM , we initialize that partition matrix, then we update centre of the coordinate this three steps are iterative. We update the centre coordinates according to that again update partition matrix. And for each alteration previous partition matrix and current partition matrix is compared and if the difference is below threshold will stop iterating, so if it is again threshold we again go iterate.

(Refer Slide Time: 17:06)



Now, for this as I said before besides the number of clusters, it may be that the number is not appropriate data. So, we need some matrix to evaluate the quality of clustering we have got, there are different algorithms that gives us index value telling what is the quality of clustering, out of that we are using p b m index.

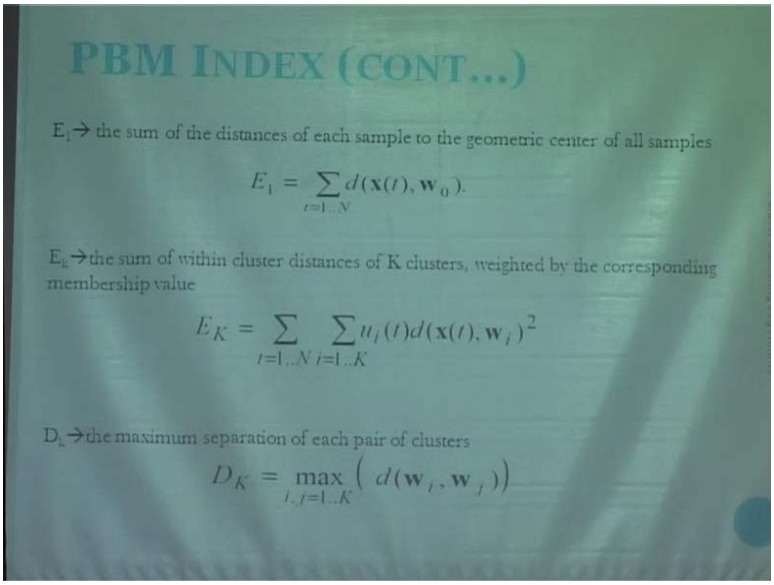
(Refer Slide Time: 17:35)



So, the index value is calculated has follows this is the formula where k is the number of clusters that we want to have, e 1 is the sum of distances from the geometric centres of all samples. So, this can be considered has expression for considering the entire set up

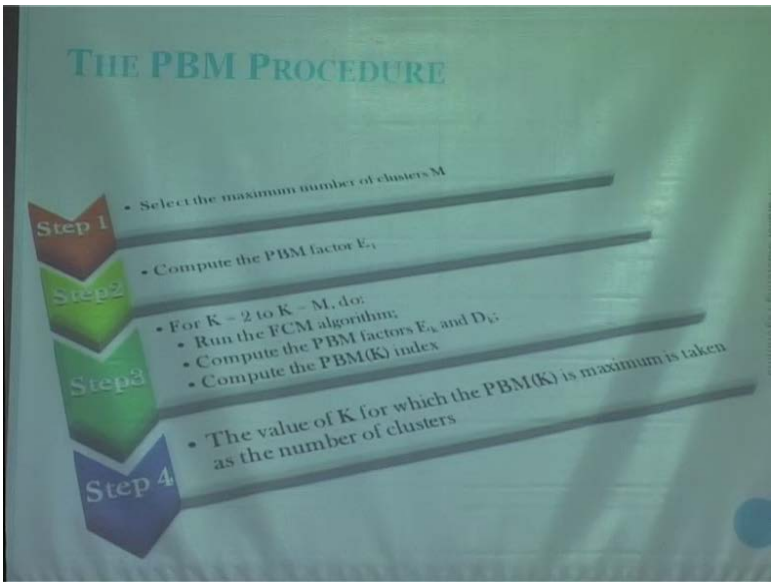
point has a single cluster. And this is for each individual cluster for that the same distance for the set up points and d k is the maximum separation of each pair of clusters. So, after partitioning we calculate the value of this, so we start with different values of k and for each value of k we calculate the value of the index. For the value we get maximum value of index we assign that number value is most appreciated for clustering of data.

(Refer Slide Time: 18:29)



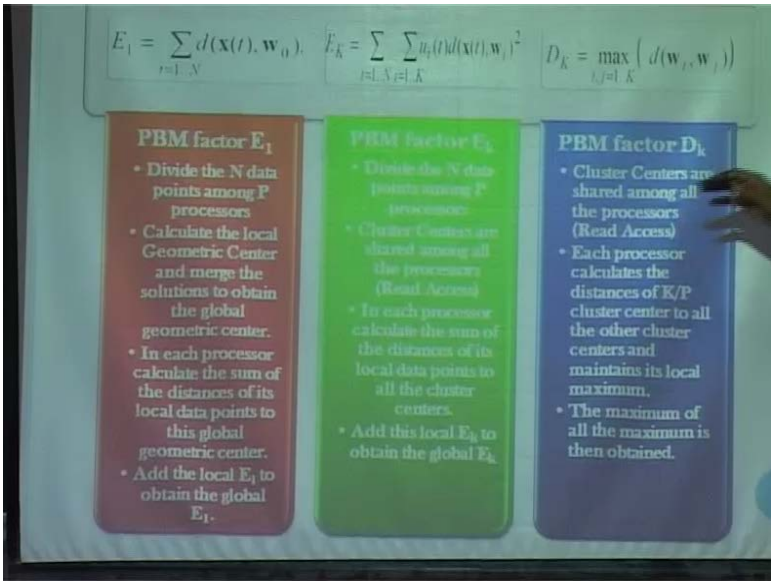
This are the formulas used for calculating E 1, E k and D k this is as said taken has the entire data is taken has 1 cluster and the distance are taken here, it is taken into k clusters here it is taken into k clusters and d k is the maximum separation between clusters.

(Refer Slide Time: 18:45)



Then the entire procedure can be summarized has this first selecting the maximum number of clusters and then because even when we can compute the number of clusters it is independent of the number of clusters, so we can compute in advance and when we run this for various values of k, first the FCM algorithm. Then we compute E k, D k, PBM k after that after doing this we find what is the maximum value of this P B M k. And after that whatever the maximum value is we consider that to be the activate number of clusters.

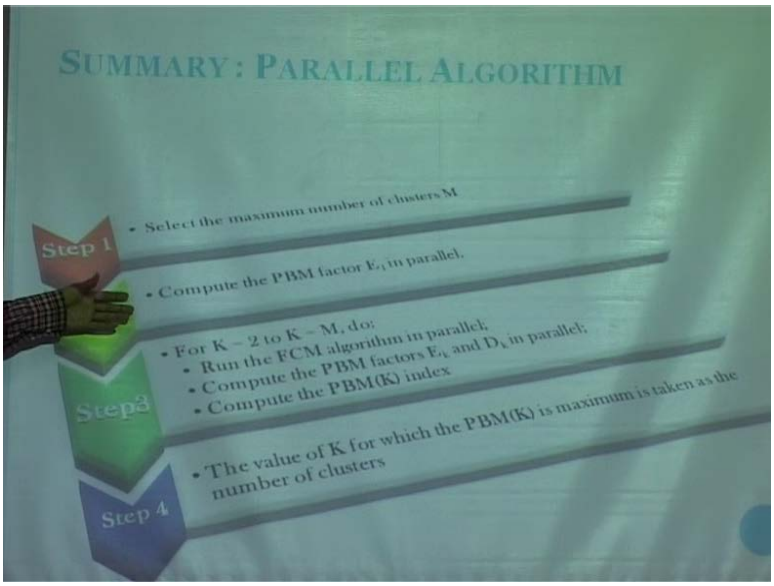
(Refer Slide Time: 19:16)



So, here is the parallel version of FCM, again the operations are much independent so we divide to calculate the partition matrix, again we divide into p groups each processor we work on the partition matrix. Then to calculate central coordinates again we divide into p groups, so this is alternative procedure to update partition matrix again we divide a first into p groups. Now, to calculate difference between the old partition matrix and the new partition matrix, similar principle is applicable. We again divide data into p groups and calculate the difference, so to calculate the index values we need this three expressions, this is independent for each point and again division of p sets, E k is for each point and each cluster.

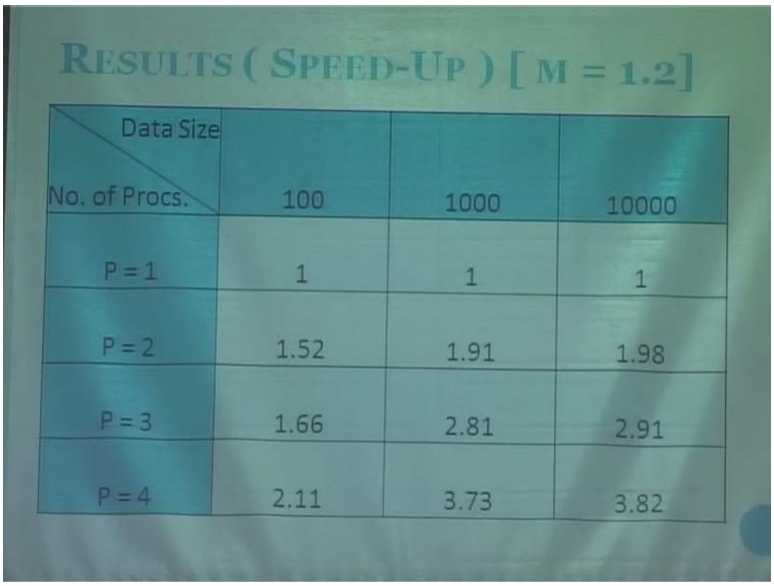
So, this loop we can divide into n set, during this auto summation from the first processor from 1 to n by p is from second processor to n by p and similarly in E k also the outer loop is divided. So, to calculate the maximum distance among the clusters we have around k square pairs, so we divided into k square by p and each will calculate its local value of maximum. And after that all the local values of maximum so sequentially we have to get global maximum.

(Refer Slide Time: 21:02)



We can do this part in parallel computing E 1 and then has said earlier FCM algorithm can be done in parallel and this we can calculate E k and D k in parallel. And then this is done in sequentially and stay constant just a multiplication and then we find the maximum of PBM k the cluster will be the and in this case we got this result.

(Refer Slide Time: 21:32)

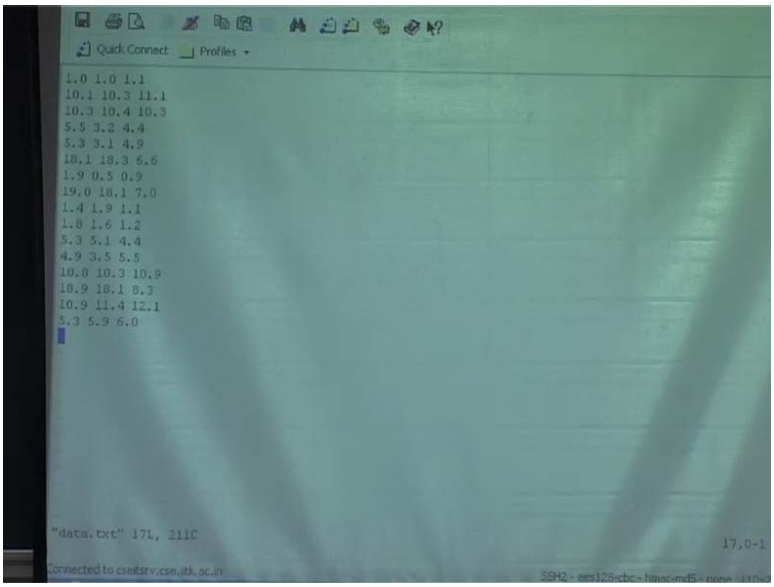


RESULTS (SPEED-UP) [M = 1.2]

Data Size / No. of Procs.	100	1000	10000
P = 1	1	1	1
P = 2	1.52	1.91	1.98
P = 3	1.66	2.81	2.91
P = 4	2.11	3.73	3.82

For 100 we are getting, but for 10000 we are getting step size around same thing.

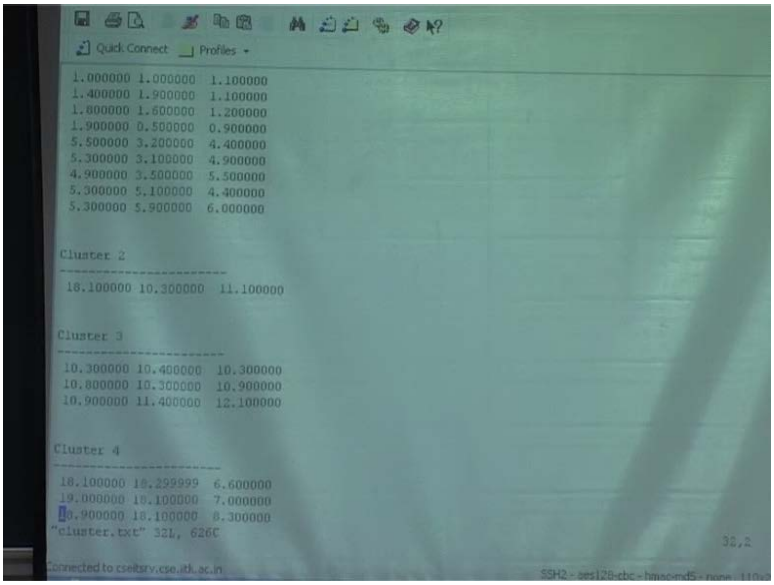
(Refer Slide Time: 21:45)



```
1.0 1.0 1.1
10.1 10.3 11.1
10.3 10.4 10.3
5.5 3.2 4.4
5.3 3.1 4.9
18.1 18.3 6.6
1.9 0.5 0.9
19.0 18.1 7.0
1.4 1.9 1.1
1.8 1.6 1.2
5.3 5.1 4.4
4.9 3.5 5.5
10.8 10.3 10.9
18.9 18.1 8.3
10.9 11.4 12.1
5.3 5.9 6.0
"data.txt" 17L, 211C
17,0-1
```

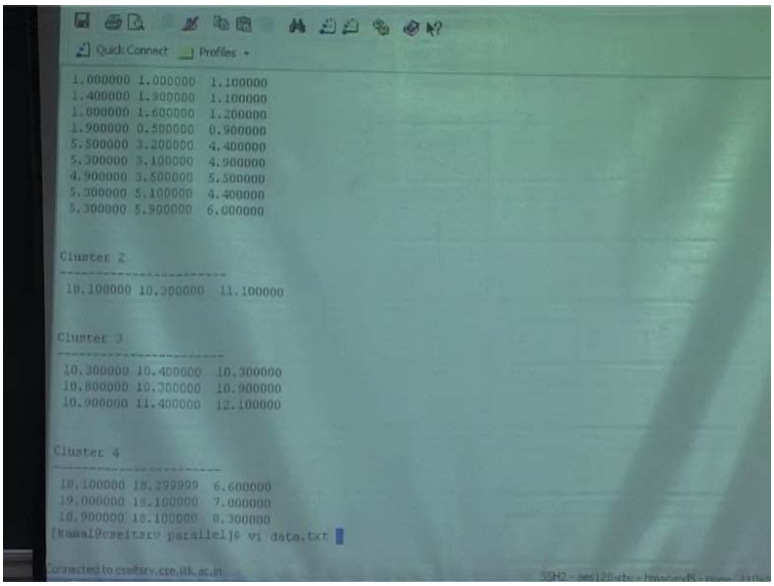
So, for simplicity we are taking this 16 points around 1 and 1 other 10, 11, 5, 3 and 4 and 18, 6 for that this is the output we are getting the four clusters from here all the data points here. Change the data obtaining the same data we will or you made 10.1 is there rite you make it 18.1, so 18.1 will be increased right? Yeah.

(Refer Slide Time: 23:45)



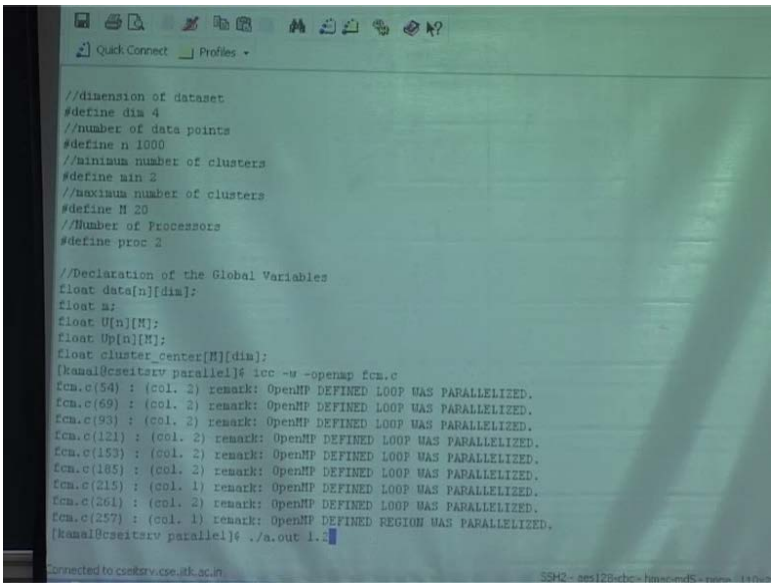
At the end of the semester also your time table is not matching, so this is what we have got, actually the second and the third coordinate are not similar. Actually it is 18, 19, 18 we are getting a cluster 18, 18, 6 we are getting on one cluster and here we are getting a separate cluster and these are taken as one cluster, why cluster two and cluster four is there? Sir second dimensional and third dimensional are much different. It is 10.3, 11.1 and here it is 18.3 and 6.6, this two clusters were much closer 1 1 1 and 5 3 4 2 are much closer than that.

(Refer Slide Time: 24:44)



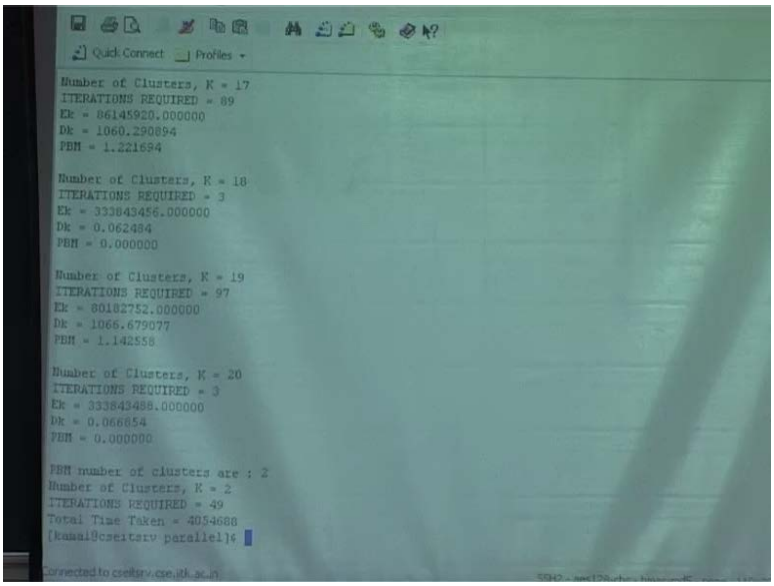
Sir if we change that.

(Refer Slide Time: 25:25)



Sir do it fully, sir this 18.1, 16.29 anyway show the other one, sir for 1000 data sets I sent.

(Refer Slide Time: 27:00)



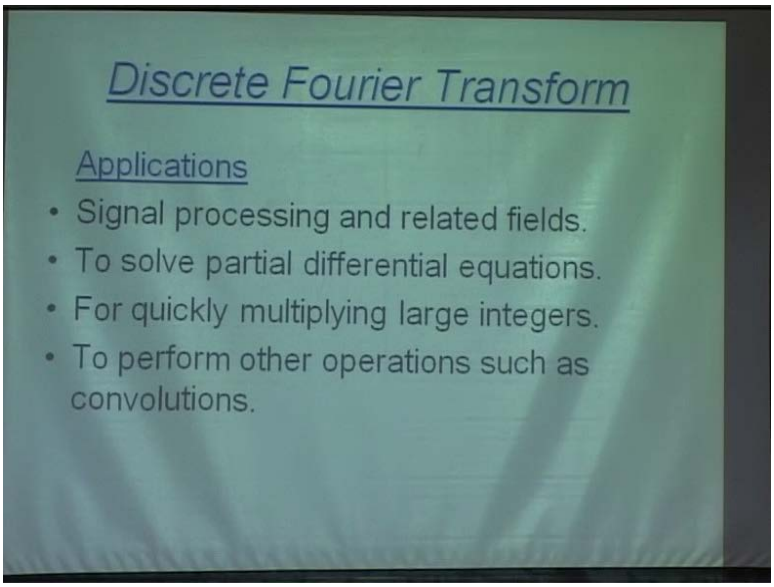
Sir this time taken for this in micro seconds, for taking as 1 processor and now if we take it as 2. So, this was the time taken almost it is displaying all before set 4, this was the time taken, this is for data set size equal to 10000, but in case e k was the factor in PVM, E k and D k, E 1 was constant for all.

So, initially it was before going into the FCM loop, initially E 1 was calculated at the initial part E 1 is equal to 15, 2 6, 6 by 1.5 and then for each value of k this E k, D k and P V M are calculated. And finally with the highest p v m it was 1.934 for 50 equal to 2, then for 6 it was 1.8, 9 it was 1.5 so maximum was for 2, so it was giving this or k equal to 2 then this was the time during 4.0.

(Refer Slide Time: 29:03)

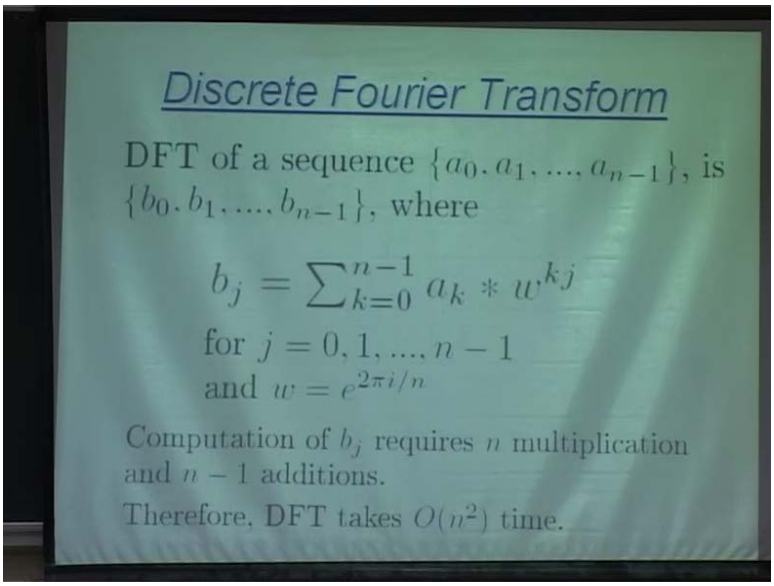
Sir our presentation is about parallelizing FFT.

(Refer Slide Time: 29:07)



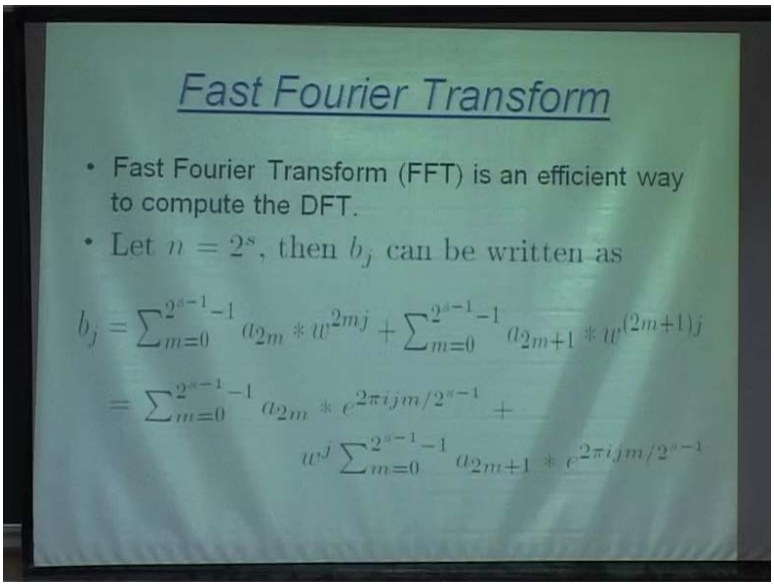
The basic DFT Discrete Fourier Transform has lot of applications, like in signal processing and its related fields to solve partial differential equations or for multiplying large integers or to perform various operations and convolutions. Here the main thing is with DFT is that each of the DFT is get called a lot so we should try to optimize it as much as possible.

(Refer Slide Time: 29:31)



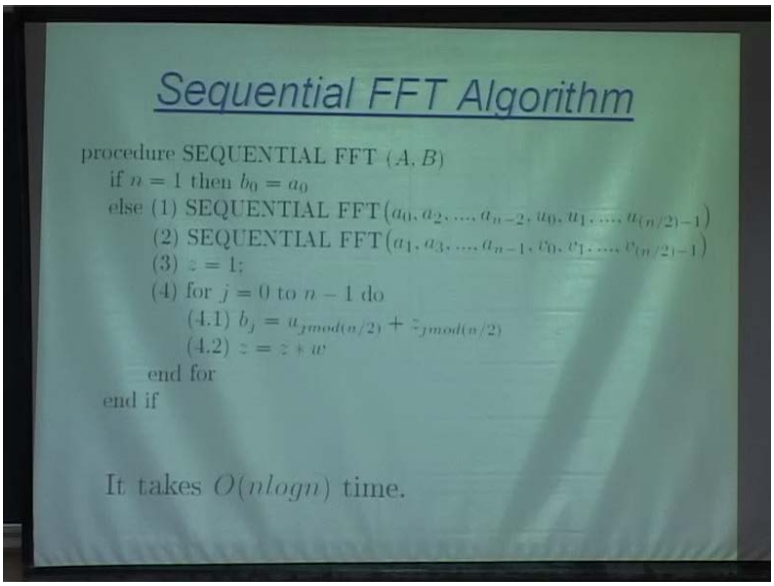
This is the basic DFT formula where you have b j is equal to this is your data set and that is the complex variables and then you multiply it to get b j.

(Refer Slide Time: 29:44)



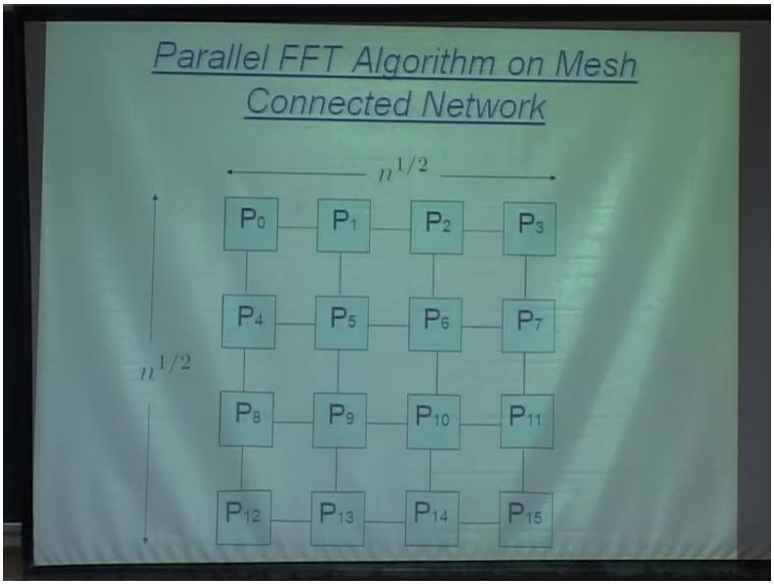
And here you have FFT, FFT is actually very simple basically instead of using the whole data set you divide it into odd and even. And then you use even terms to multiply to the even terms in a data set and the odd complex ones to odd a terms.

(Refer Slide Time: 30:03)



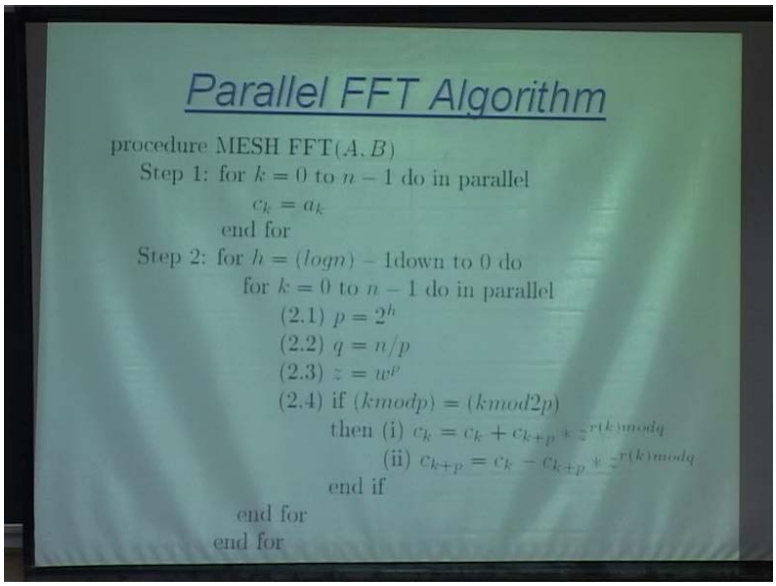
And then here we have the sequential FFT algorithm, this is basically very simple here there it just happens to have sort of recursion here, where you send all your even terms here and odd terms here. This will take up o of n log n time so we have been trying to parallelize this.

(Refer Slide Time: 30:23)



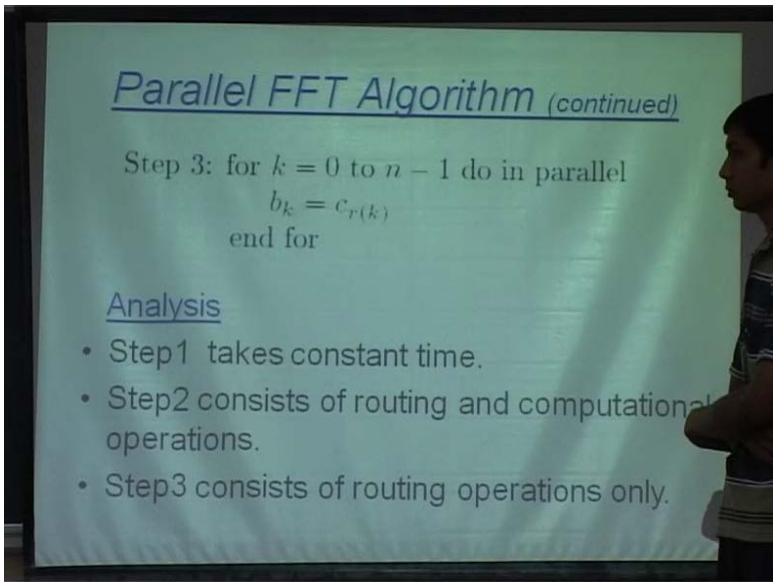
This is the basic paralyzing algorithm for the mesh connected network, here if you have like 16 processors and you have a data set of size 16. Then you have 4 on the rows and 4 on the columns and this things will be connected in a mesh.

(Refer Slide Time: 30:43)



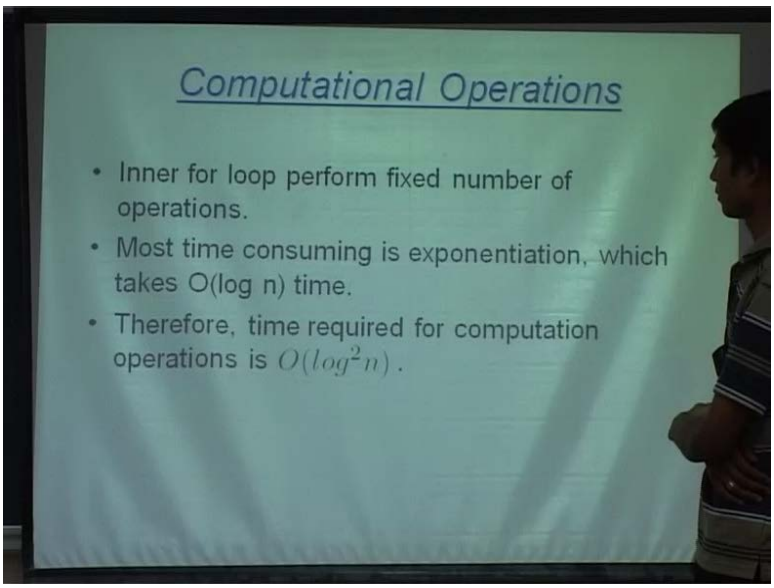
This is the basic algorithm in the parallel FFT, basically here you will be bringing the values to each of these processors, each processor will have its own value. And according to the log n represents the row or the column basically it is like you have 16 data sets so this becomes 4 and when you take it here and you multiply it here. The main paralysation comes here when you say C k is equal to C k plus C k p, it means that C k will request the value of C k plus p and C k p will send the value here. It will do the necessary computation create a new value of C k plus p and it will send and it will send it back. So, basically there is a send from C k plus p, receive from C k and again there is a send from C k and C k plus p will receive its new value here.

(Refer Slide Time: 31:33)



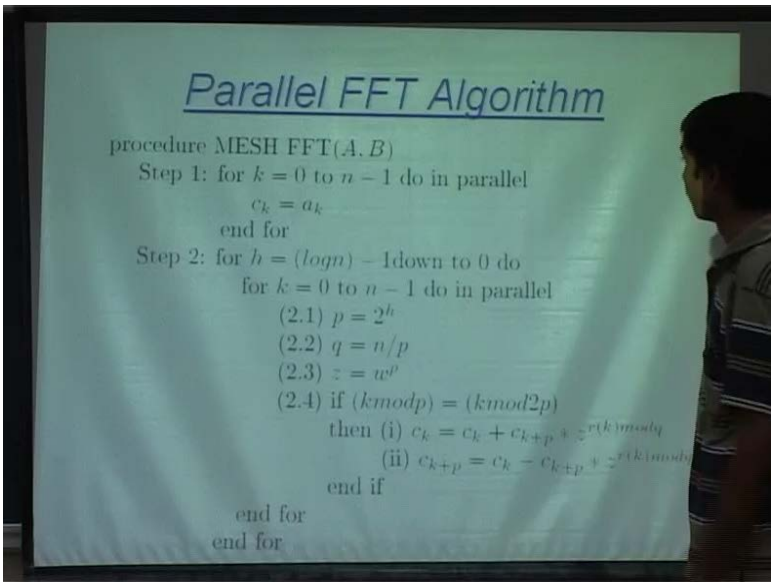
Then you have the last step where there is small computation of r of k, which is basically reversing the bits you have, like four becomes twelve. So, basically just reversing the bits, step one takes constant time basically it is just like assigning values so that takes constant time. And step two will take you lot of routing and computational steps which we will come to it and this step three contains only routing.

(Refer Slide Time: 32:01)



Back...

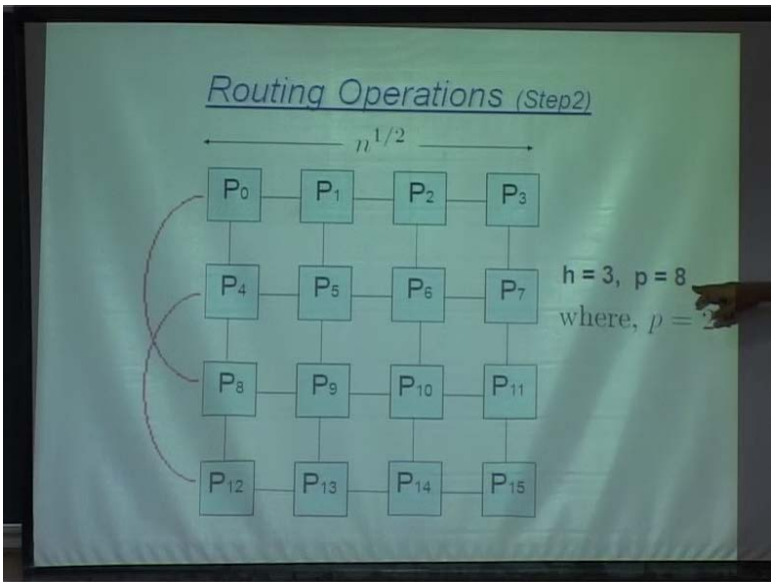
(Refer Slide Time: 32:05)



Step one will take only constant time, step two consists of computational as well as routing operations. So, we can see that there are six number of computational operation, out of which this exponentiation is the most time consuming. And it will take order of log n time and as the loop iterates for log n times the computational will take, what is p? p is just we are calculating it here 2 power h, h you get from the loop and p is equal to 2 power h that is all. All this foundation is pretty straight forward here only this part we have to understand that is all, computation will take order of log as current term.

Now, come to the routing part here as Harsha told that every processor with index k will send value to other processor having index k plus p and it will also receive value from it. Now, we will take an example of sixteen processors where value h will initialized to three and p equal to p will be there for h.

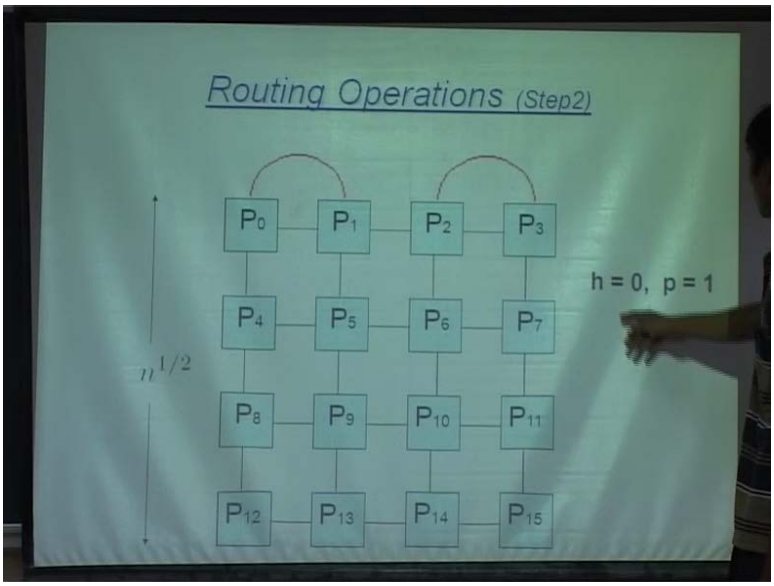
(Refer Slide Time: 33:14)



Now, routing here the value of h equal to 3, therefore p is equal to 8, now routing will takes place between processors with the net. Basically the routing will takes place between the first row and the third row that is P o will talk to P 8, P 1 will talk to P 9, P 2 will talk to P 10 and P 4 will talk to P 12. Basically what the first perception of anybody would be that you know this routing should be very simple we can do it, but when you are doing in p v m this becomes very complicated I will show you later. The importance of this thing is we are tried to calculate like what how many steps take place in each of these iterations. So, this is the first iteration where you will take from p 0 to p 8 and P 4 to P 12 and in the second iteration value of H is 2 and P equal to 4, so routing will takes place with processor index 4 and in the 3 iteration you have this.

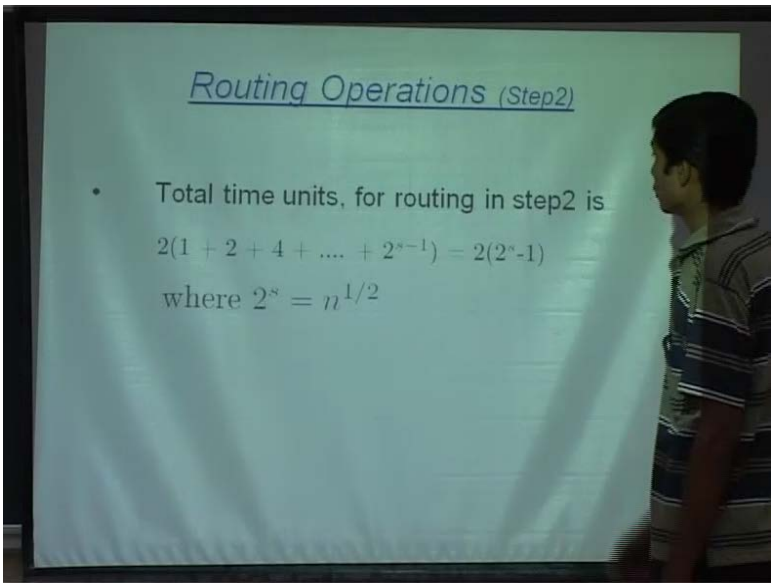
Now, this will takes place in the same row, so routing becomes a bit more easier actually it will go from here to here and P 1 to P 3 at the same way P 4 to P 6 and P 5 to P 7. Actually I will tell you when p v m comes the routing from one row to another it becomes a bit easier. For example, if you want to bring from P 3 to like P 8 you have to come from here, no we are routing in this form this to this, no that is I am just giving as example, P knot to P 8 is just simple you just come from here to here, no you told that (( )) row wise here also it will come from P 1 to P 2.

(Refer Slide Time: 35:00)



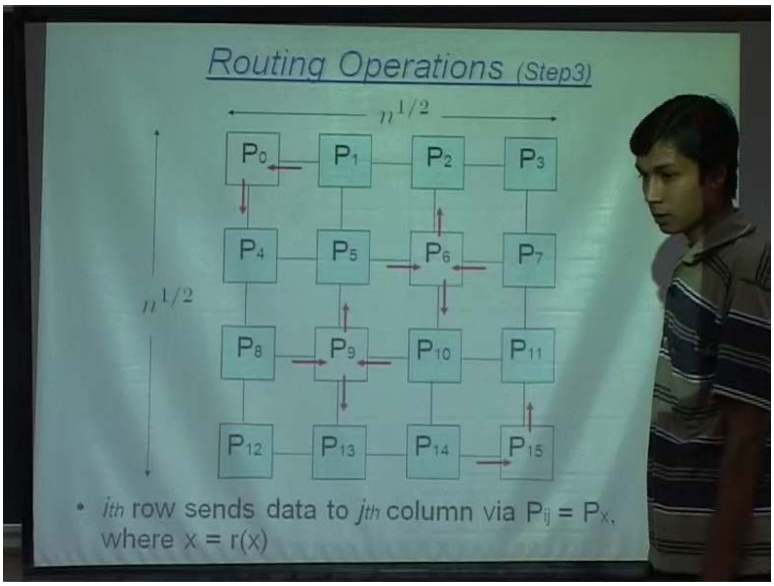
In last iteration value of h is 0 so routing will takes place between these processors.

(Refer Slide Time: 35:05)



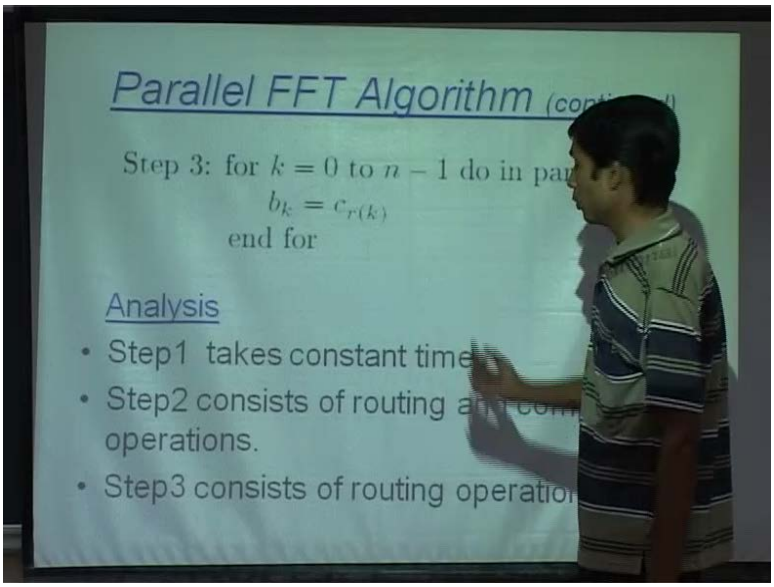
So, this routing process can be represented by this expression here 2 to the power s equal to root n and multiplied by 2 because each processor will sending as well as receiving values. Therefore time taken by this routing process will be N minus 2.

(Refer Slide Time: 35:22)



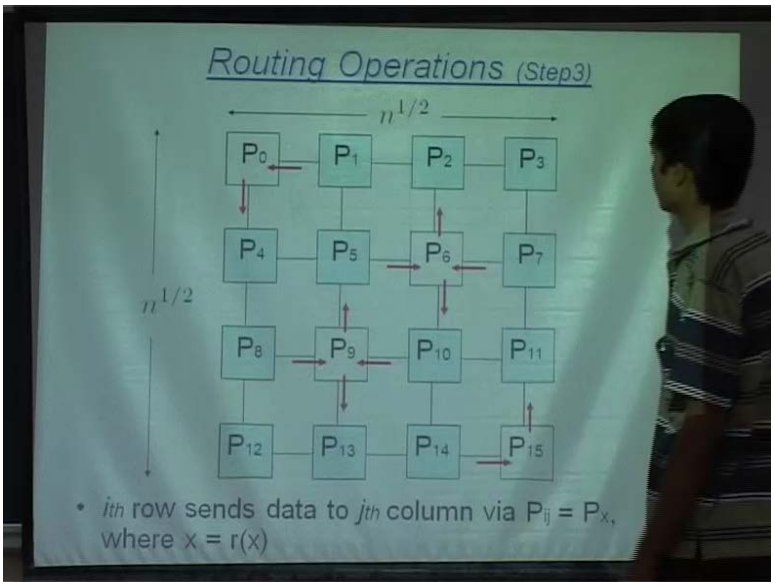
n minus 2...

(Refer Slide Time: 35:26)



In step three we are sending as well as receiving value to the processor whose index is reverse of itself. Now, we will see this by simple example.

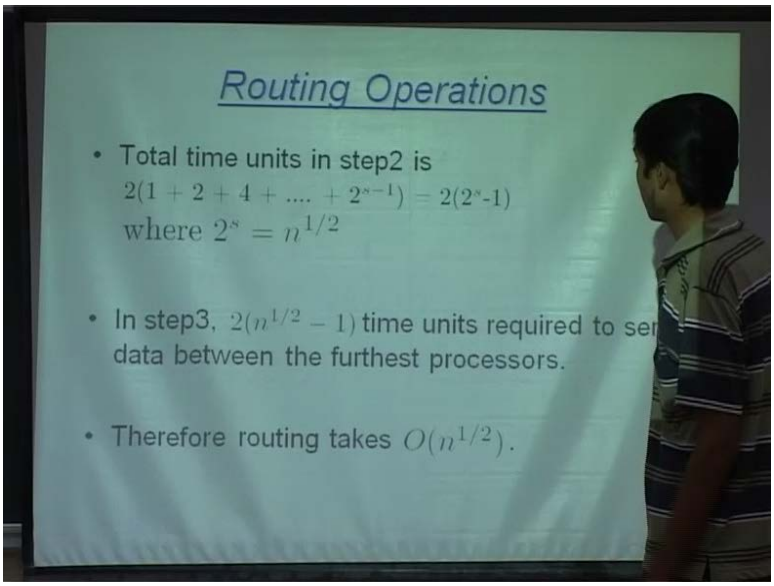
(Refer Slide Time: 35:41)



Here one thing we can notice that the processor in row one are sending value to processor in column one, second row will send value to this column, third row will send to this column, so we will send this element by this processor P 0, P 6, P 15 and P 9.Here what happens is the reverse of P 0 is P 0 itself, so it does not have to send and the reverse of P 6 is P 6 itself. So, when you are sending it we created a route it will first go from all the values in row one will go here and only then they will go here. And the same way here what happens is the values from P 4 to P 5 will go here and then they will make a decision whether to go down or upwards P 7 as to go to 14 so P 7 will go down like this.

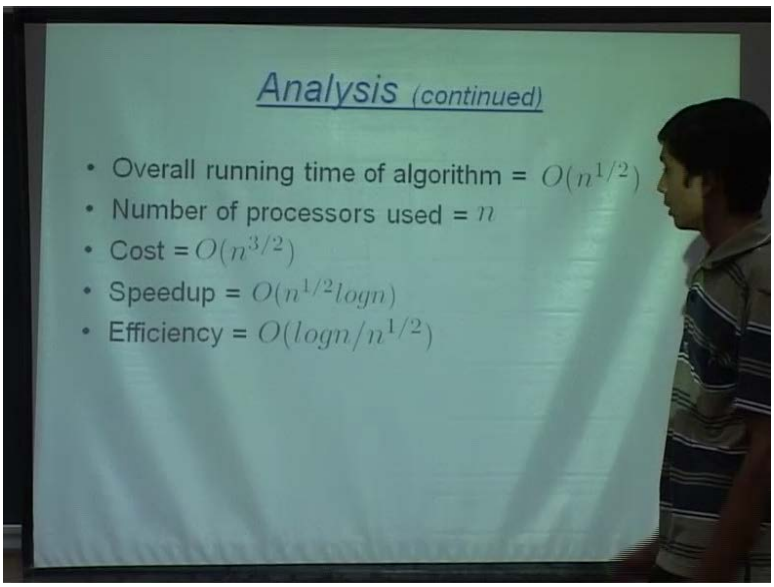
This is like much better approach of doing it when you are doing it with an algorithm even if you take it to a 64 mesh this thing this will become very easy for you and this will be the second iteration in that. Second time we will receive those values because each processor is sending as well as receiving values. So, we can see that the maximum distance between any two processors here is 2 into root n minus 1 doing this end.

(Refer Slide Time: 37:07)



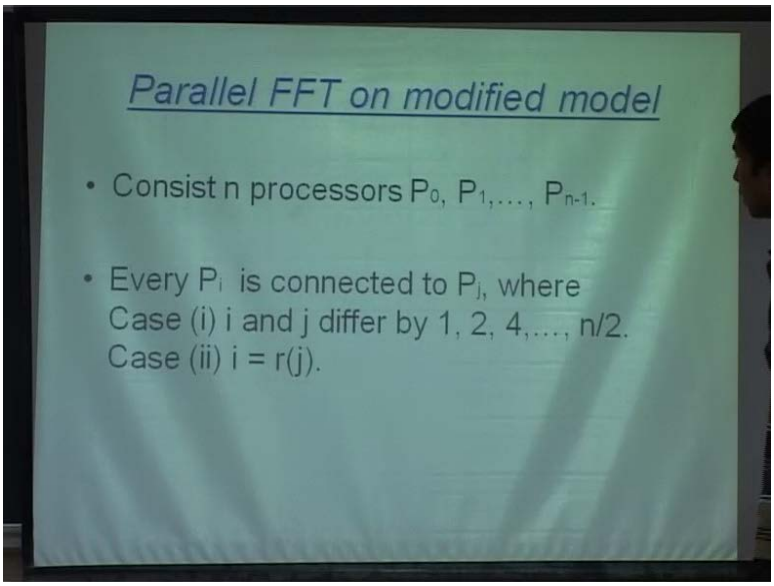
So, order of this routing type is 2 into root n minus 1. Therefore here we saw that routing dominate the computational time, therefore we want mainly optimize the routing in this in the mesh of 50.

(Refer Slide Time: 37:25)



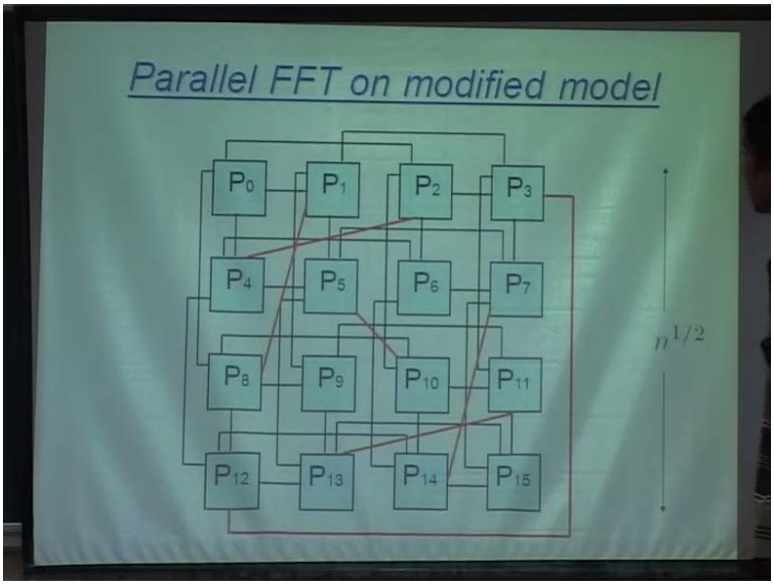
So, the whole routing will take the order of root n time and therefore complexity of algorithm is order of root n, number of processors used here is n. Cost is order of n to the power 3 by 2 speed up we are achieving is of order of root n log n and efficiency is log n by root n.

(Refer Slide Time: 37:44)



Now, come to the modified model here we are using n processors P 0 to P n minus 1, here every processor P i is connected to this is the model we created sir completely. Here P i is connected to P j where i and j differ by 1 2 4 up to n by 2 means index differ by power of 2. Case two is where index is reverse of profit sir.

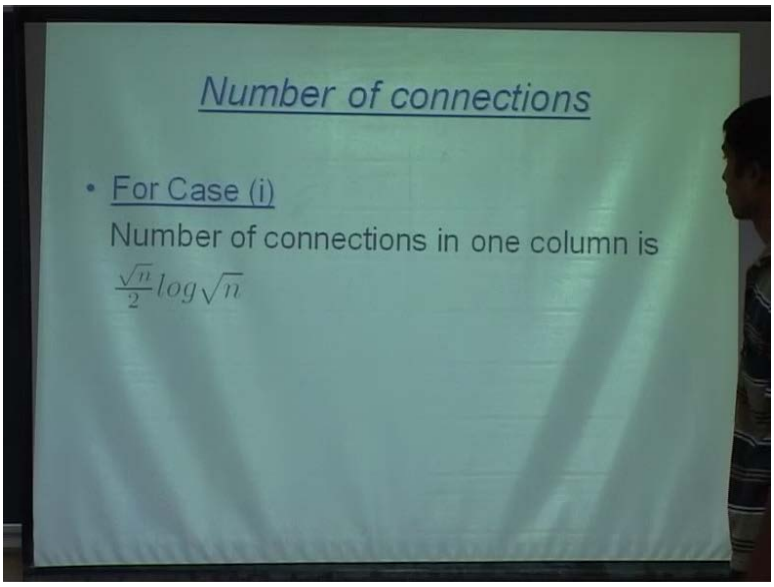
(Refer Slide Time: 38:18)



So, the model will look like this here the index of P 2 is reverse of P 4, so this red edges are basically for step three and this black edges are for step two. Basically you are really maximizing the number of connections you will have and you are looking at which

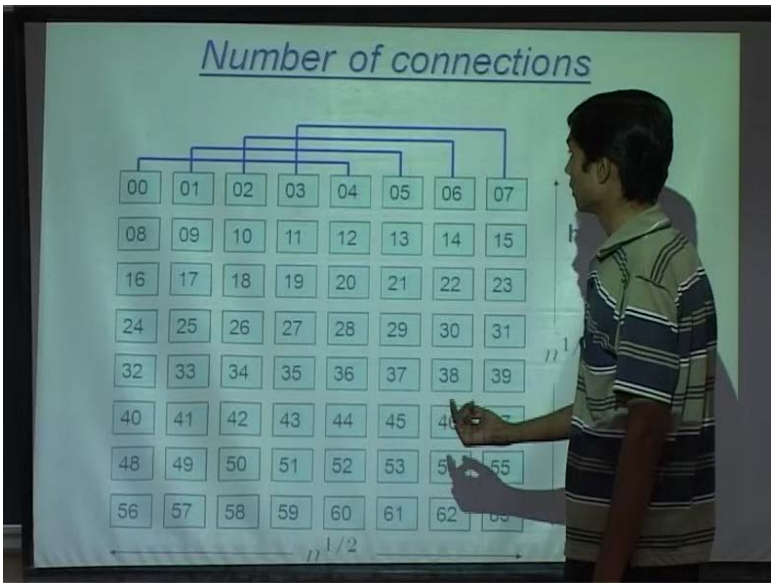
possible connection will be used and you are adding that connection, so instead of having a whole mesh this will be much easier to use for. And one point we can note that we have removed this connections from this and this also and also here this there are no connections here.

(Refer Slide Time: 38:55)



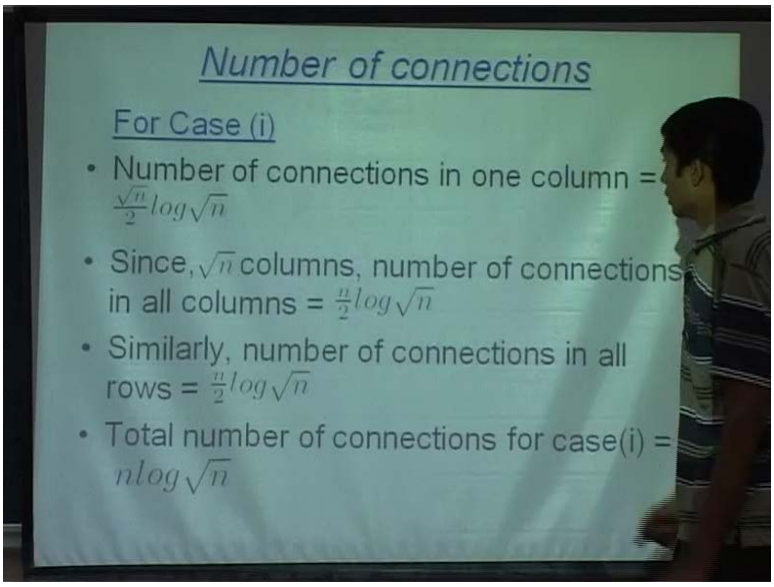
So, now we count the number of connections we use here, number of connection in one column is root n by 2 log n.

(Refer Slide Time: 39:05)



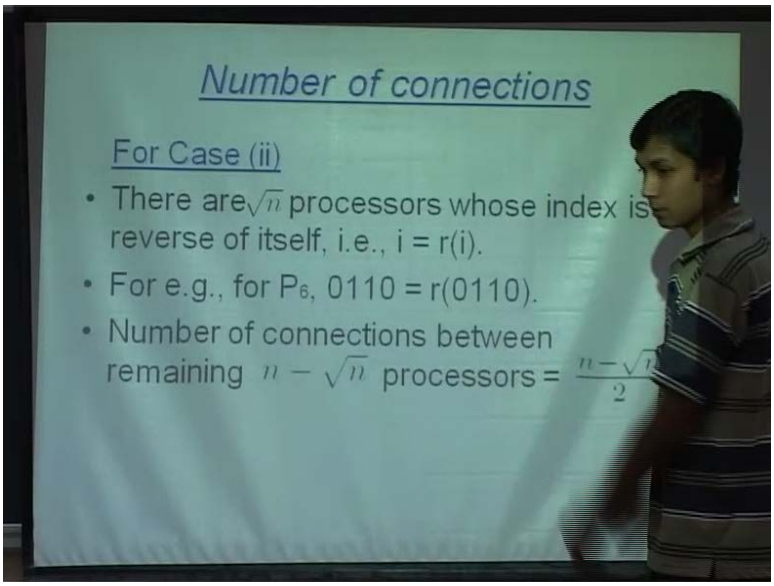
For an example in one column there are root n by 2 connections like this, next root n by 2 connection like this and like this root n by 2. Sir this is basically when for each of the h value this is for the first iteration of h, second iteration of h, third iteration of h. Log root n times root n by 2 and there are root n columns therefore total n by 2.

(Refer Slide Time: 39:38)



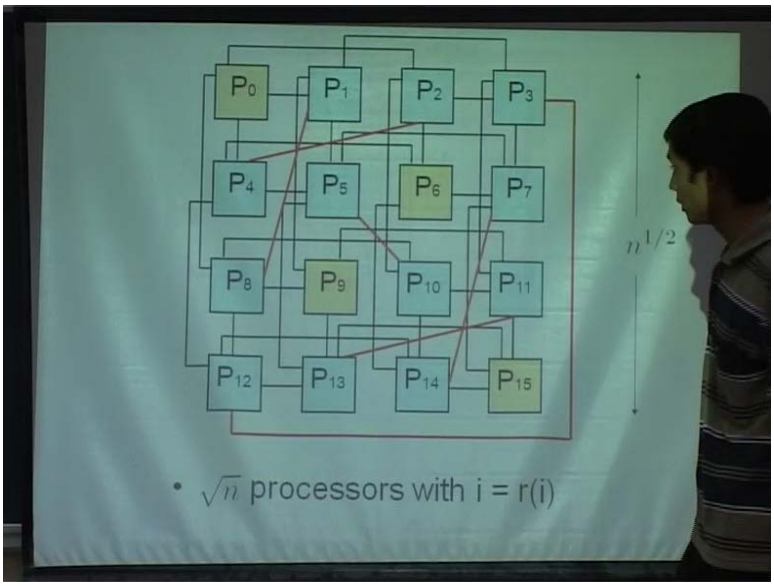
Similarly, number of connections in all columns will be this, similarly for each row will do connection and therefore total number of connection for case one is n log case n. Case one is the step two the main computation step that we are using from C k plus P 2 C k and again the reverse direction, that will take you n log of root n.

(Refer Slide Time: 40:04)



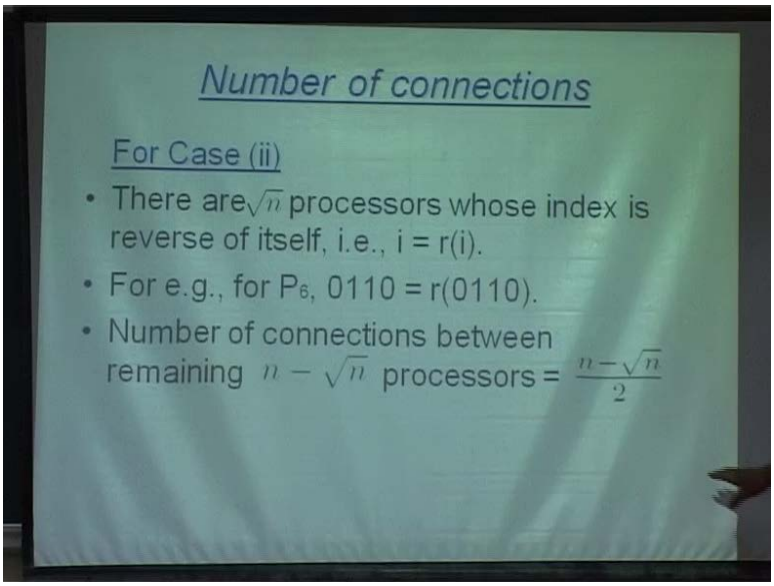
And every processor is having fixed number of connection, yes sir fixed number of connections what is that number value? Sir we are counting in case two. next (( ))

(Refer Slide Time: 40:20)



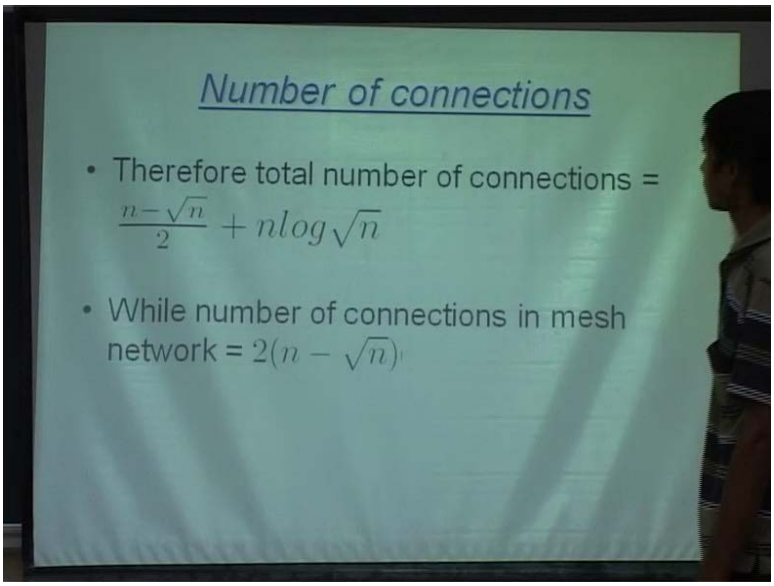
We can see that root n processors are there who do not need connections for step three because the index are 6, 9, 15 are reverse of itself back.

(Refer Slide Time: 40:30)



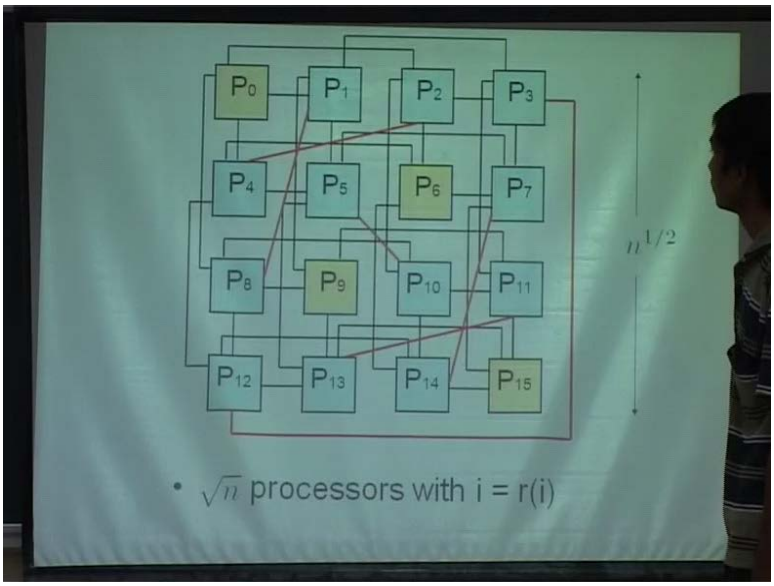
So, there for root n connection processors do not need require any connection. So, the remaining n minus root n processors require only n minus root n by 2 connections.

(Refer Slide Time: 40:46)



For step three, therefore total number of connections will be n minus root n by 2 plus n log root n, while number of connections in mesh network is 2 into n minus root n. No for your case some processor is having less number of connection, some processor is having more number of connections agree? No.

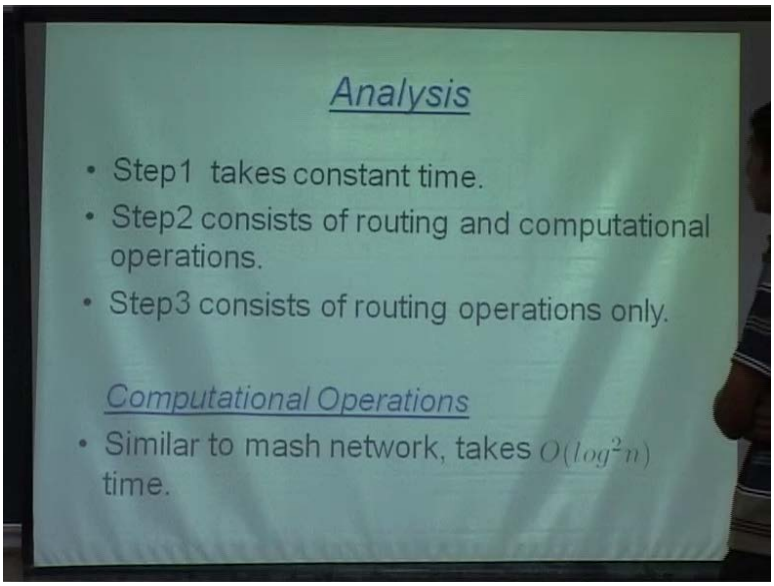
(Refer Slide Time: 41:10)



Say P 10 is having how many connections? P 10 will have like 1, 2, 3 and 4, 5, the same P 0 will also have five connections, P 8 will also have five connections, P 2 will have five connections, P 0 there is nothing this side. So it will have four connections, but there is a pattern actually we can design from that, P 9 is having four P 9 it will have four because P 9 is a reverse of itself I am not justified with that, what is that P 9 is having four, P 10 is having five right.

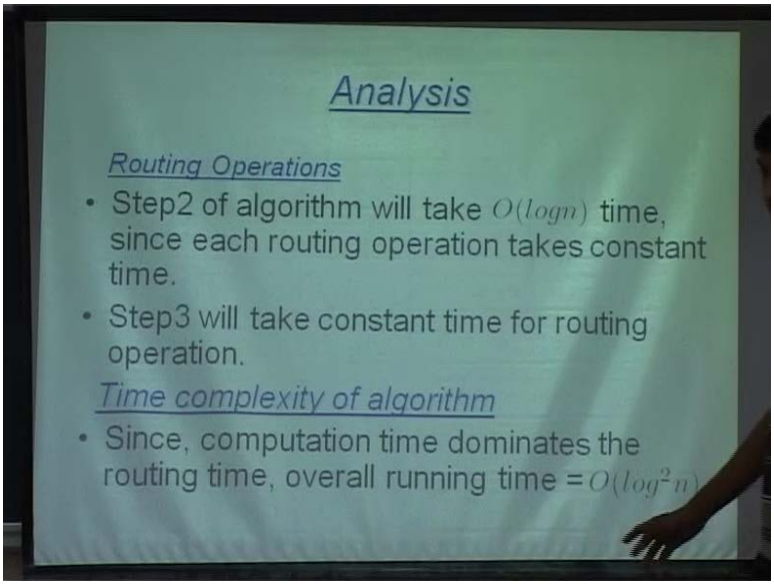
So, can we have a formula for that? yes we do have a formula because when we write an algorithm we need a formula like this to actually use (( )). Sir here we need one less connection because it is reverse of itself, see that is you need less connection, but the point is that if you want to fabricate it, then you need I got it sir so go for formula yes sir. If we pass the index we should know the number of connections we will have and which connections it will have.

(Refer Slide Time: 42:48)



Now, you can do the analysis of this we are using the same algorithm here, so step one will take constant time, step two consists of routing and computational operations and step three consists of routing operations only. So, computational operations similar to mesh network take the order of log raised to n time.

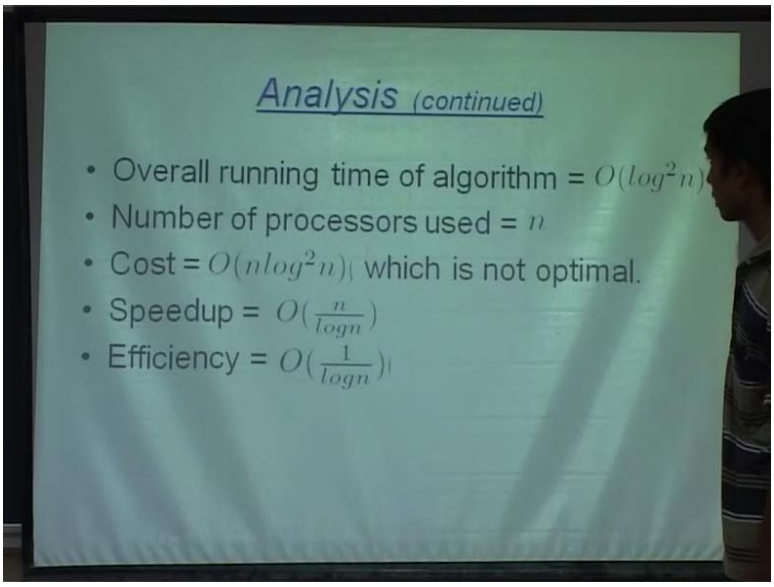
(Refer Slide Time: 43:08)



Step two routing will take constant time (( )) and the order is log n time therefore, order of log n time for step two and for step three constant time is for routing operation. So,

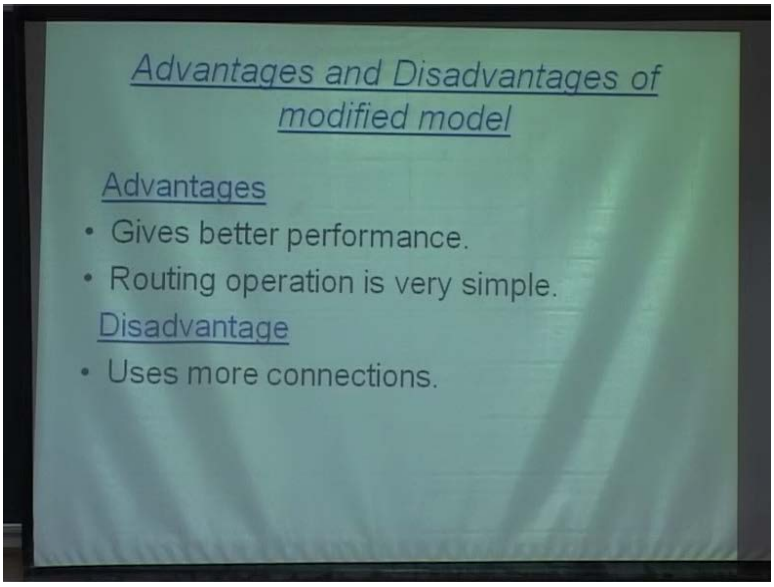
time complexity of algorithm is order of log raised to n power times here the computation is dominating the routing.

(Refer Slide Time: 43:25)



So, overall running time of algorithm log square n, number of processors is n, cost is order of n log square n which is not optimal and we are actually speed up in order of n by log n and efficiency is order of 1 by log n.

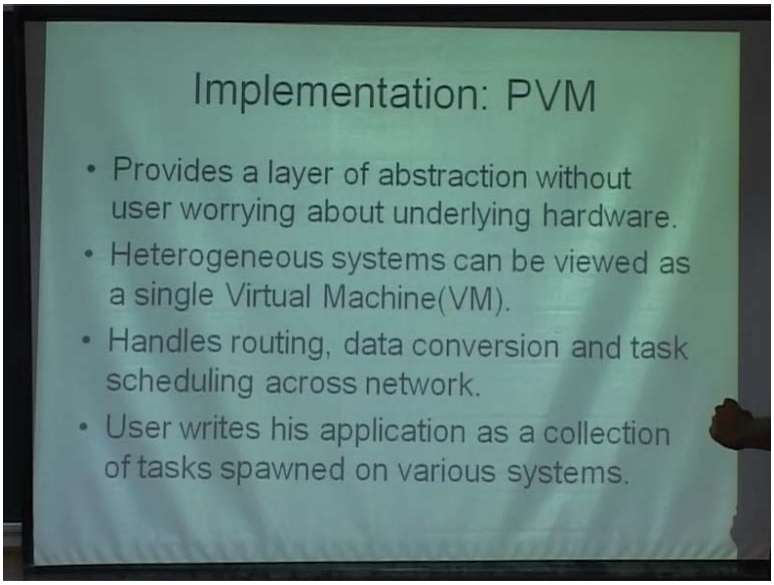
(Refer Slide Time: 43:44)



So, advantage here it gives better performance, routing is very simple. We already know which processor you will be sending through, so the routing happens directly there are no

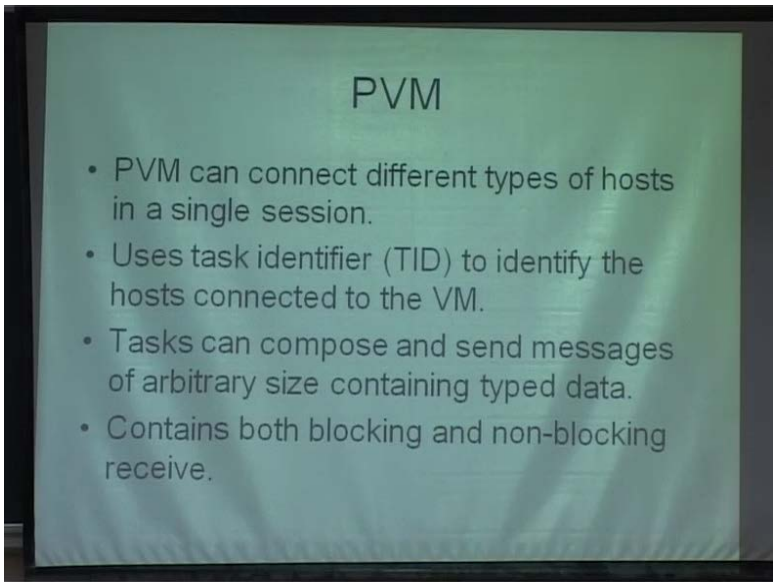
hops actually it is just one connection. We have more number of connections compared to the mesh network and also it is far more complicated, in mesh network we knew exactly it is going.

(Refer Slide Time: 44:10)



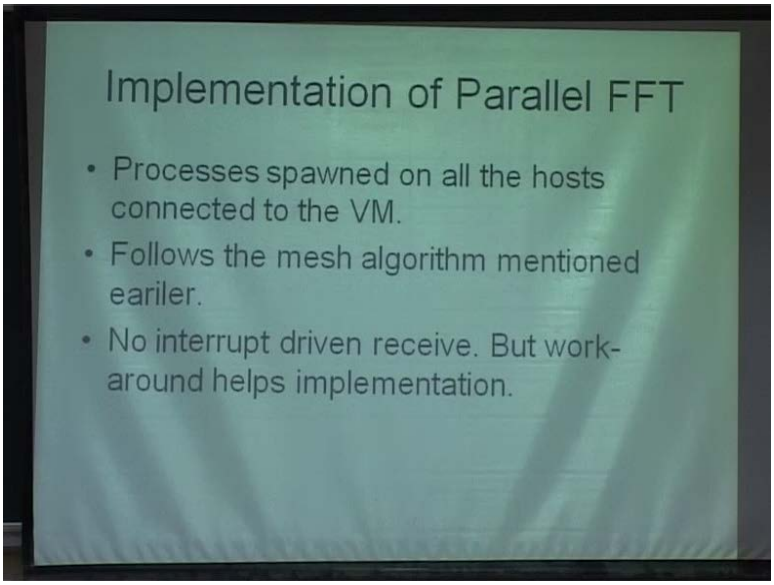
Implementation we did it in PVM, so PVM basically it provides a layer of abstraction without the user worrying about the underlying hardware. And heterogeneous systems can be connected to a single PVM and PVM will give a certain abstraction that all these machines will be using the same data formats and so. It also handles the routing data conversion, task scheduling across the network. The user just writes his application as a collection of task and spawns it on various systems irrespective of the hardware. Even if they are heterogeneous systems the PVM will actually give you some sort of an abstraction layer where you will say all the systems have some common.

(Refer Slide Time: 44:55)



It can connect to different types of hosts in the single session, it uses certain variable called the task id, which is used to identify the host that is connected to the VM. Tasks can compose and send messages of arbitrary size containing typed data, that is you can define initially that I will be sending a int at this instant or a float at this instant.

(Refer Slide Time: 45:20)



The implementation of parallel FFT the process spawned on all the hosts will be connected to the virtual machine basically. And first algorithm be implemented follow the mesh this thing and there is no interrupt driven receive, so this is one of the main

drawbacks of PVM like for example, in a processor you have spawned a thread and it is running. And if you do not know exactly when on which processor what value will be coming and where it has to be forwarded. The problem with blocking receive is that it will give you an infinite loop and the problem with non blocking receive is that you have to be constantly checking.

Instead if you have something like an interrupt driven receive, where whenever I receive a value the program will just say that ok I have received an interrupt, I will go and handle it and then I can come back to my computation that would have been much better, but PVM does not provide that. And that is a reason why we have to go through all the computation of the routing like how many we had to go and all that, we did it in c c sir using sixteen machine. What is the performance you have seen? Sir performance the problem is measuring on PVM what I have to do is do is I have to create a master thread which spawns this thing and from that I have to take, because each of this machines have their own clocks.

So, that actually what happen is first algorithm came out at an average of 32 micro seconds, second one came out 21 micro seconds, data set is 60 (( )). Sir that we have not done that becomes a completely different algorithm that does not become this, because generally FFT matrix size is very large yes we know that (( )), but this algorithm actually we do not use that part, where you have certain array in each of the processors, so any questions (( )) do not want to disturb him.