

Matrix Solvers
Prof. Somnath Roy
Department of Mechanical Engineering
Indian Institute of Technology, Kharagpur

Lecture – 55
Domain Decomposition Method and Parallel Computing

Welcome. We have discussed about block relaxation scheme in last few lectures. What we have observed that a big matrix equation can be decomposed into smaller matrix equations and they can be solved with some coupling in between them. And we got an idea that probably the small matrix is coming out of the big matrix can be distributed in to different computers and a parallel processing can be accomplished. We will discuss about Domain Decomposition Scheme and Parallel Computing in this particular session.

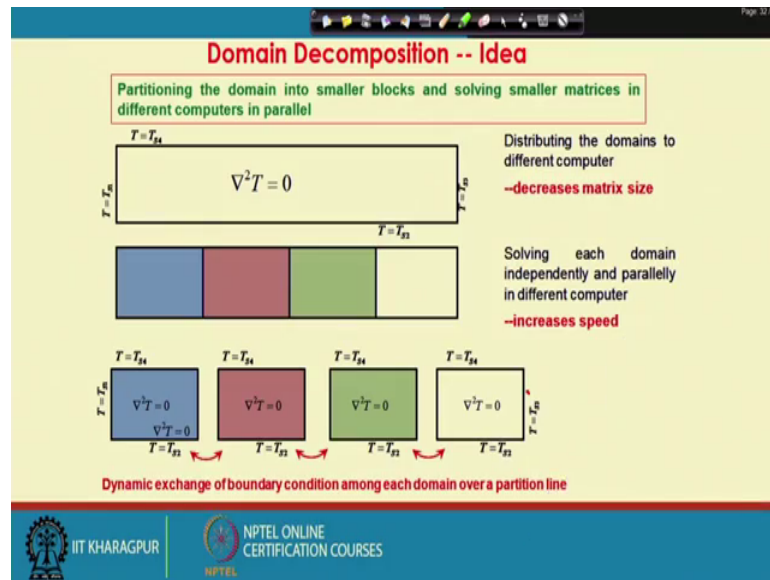
The issues that this is as it is shows parallel computing, it has a computing part, it has a computer science aspect also which is related with efficient design of the hardwares, connecting different computers and using an interconnection t v d switch and data transfer across different computers. That also involve looking into several parallel computing models, which is a programming paradigms, architectures and different message passing interfaces.

However as we are more interested in the matrix solution part in parallel computing exercise, will focus more on matrix computing of course, will use the least amount of computer science aspect that we need to know in this discussion. If somebody is interested in parallel computing, he has to take separately dedicated courses and lecture sessions and parallel computing.

I will give you some information about parallel computing, computer science aspects on that but will more focus on the on how this infrastructures are actually exploited when we do a parallel computing of matrix solvers and what are the issues then the matrix solvers has to take care of when doing this. So, the method we will use for parallel computing because, parallel computing can be done for anything. When you have your credit card and the bank processes the data of your expenditure, it does some parallel computing. Or when your insurance claims are processed like all the insurance claims that went to a hospital are processed by some server it is taking care of parallel computing.

So, parallel computing can have plethora of applications. We are particularly focused on matrix solutions and using a method called domain decomposition method. There are other data decomposition methods, data parallelization method, so matrix solvers which we are not discussing here.

(Refer Slide Time: 03:15)

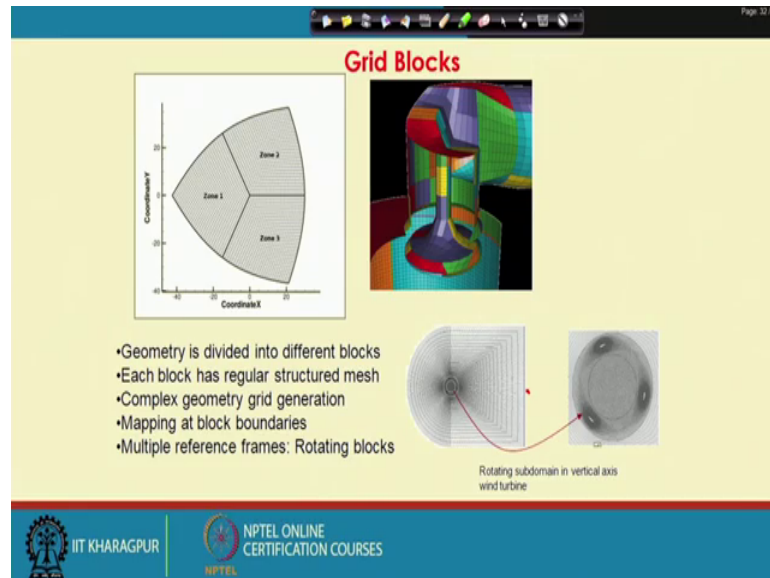


So, what is the basic idea of domain decomposition? Partitioning the domain into smaller blocks and solve smaller matrices in different computers in parallel. I have a geometry I assume it to be a large geometry where, I have to solve nabla square T is equal to 0, Laplacian of T is equal to 0 given all the register boundary condition. I will distribute this domain into smaller subdomains and give each of this subdomain to different computers. And then in each domain I will write the, I will try to solve the equation; however, the solution has to be continuous throughout the domain, so it cannot be, it solution cannot be independent from other. So, I have to dynamically exchange boundary data through the boundaries among different domains, over the partition of the boundaries.

So, the advantage is that when I distribute the domains to different computers, the overall matrix size particular pertinent to 1 particular computer is much smaller than the net matrix size. Also solving each domain independently and parallelly in different computer can increase the speed. So, we can decrease the effective matrix size or also you can increase the speed, which are good in terms of the code performance as well as the

capacity of our computing resource. We can now try a really large matrix to be solved which we cannot probably solve in our on PC with a single processor.

(Refer Slide Time: 04:55)



And this is done by several ways, one is grid blocks. So, if we have a domain over which you have to do some matrix solutions, we divided into several blocks. Now the idea of block partitioning of a matrix will come that each block is solved independently. So, it is a block matrix equation which has to be solved over the entire domain and within each block, we will do some we will do the block relaxation scheme, will solve the equation within each block.

This can be very complex or much complex geometric if we the physical geometry is very complex we can generate multiple blocks within it. If say there is a part which is rotating a turbine blades which are rotating in a domain, we can use different blocks for this parts and the matrix equations will be different for this. This is as this is rotating the momentum equation will come in rotating reference frame with coriolis forces etcetera; however, the momentum equations will be will be fixed cartesian in fixed cartesian frame here.

So, different equations can be coupled in the same system. Geometry is divided into different blocks, each block has regular structured mesh at least what we are looking into in this case, complex geometry regeneration can be obtained. A mapping is required at

block boundaries; that means, zone 1 has boundary with zone 2 and zone 3 maybe this is γ_{12} and this is γ_{13} .

So, that I can map that, this particular boundary map zone connect zone 1 2 zone like that. And we can have multiple reference frames, multiple physics at different blocks etcetera can be handled using grid blocks.

(Refer Slide Time: 06:45)

Classification of domain decomposition

- Non-overlapping domain decomposition
 - The sub-domains intersect only on their interface
- Overlapping domain decomposition methods
 - The sub-domains overlap by more than the interface

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

There are 2 possibilities in domain decomposition; one is non overlapping domain decomposition, the subdomain is intersect domains intersect only at on their interface. So, there is a single interface between the subdomains. And overlapping domain decomposition; that means, one subdomain has certain overlap with the other subdomain, they have 2 interfaces actually, so these 2 are also possible. And in blocking of the solution making blocks of the solution vector, we have seen that overlapping and non overlapping that blocks are also possible and that was done by that particular matrix w we remember.

(Refer Slide Time: 07:30)

Why Parallelization?

1. To reduce computational time by dividing the number of operations into a large number of computers
2. To reduce the matrix size to be stored in a single chunk of memory

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Why is parallelization? we discussed it earlier to reduce computational time by dividing number of operations into large number of computers. Each computer will take care of one subdomain and so, large number of computers can be associated with the entire matrix problem and it can be paralyzed in a sense then, the speed will increase the number of iterations will be same, but each one is solving a small problem.

So, each computer will perform all the row calculations local to it in a much smaller time because, number of rows pertain into each computer is smaller then we can increase the speed, we in less physical time will get the solution. Also to reduce the matrix size to be stored into single chunk of memory, instead of having a large matrix we will break down into smaller matrices. So, each computer in 1 particular RAM location it is storing a small amount of the matrix memory. This both these are helpful for the performance of the code.

(Refer Slide Time: 08:35)

Estimates of computational cost for large-scale problems

Grid spacing to resolve smallest turbulence length scale $\sim (Re)^{-3/4}$

For $Re \sim 10^6$, three-dimensional geometry needs 10^{13} grid points

One time-step (one set of matrix solutions) will need $O(10^{13})$ FLOP (Floating Point Operation)

Typical physical duration of a time step $\sim 10^{-3}$ seconds

Simulation for 10 seconds will need 10^{17} FLOP

Fastest computers give 10 giga FLOP/sec speed

Estimated time for this calculation : 10^{10} seconds = 317 years !!!!

Matrix size = $O(10^{13})$ floats = $10^{13} \times 16$ bytes \gg standard RAM size

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

We will see what can be estimate of computational cost for a large scale problem so that, I can assert that there is certain cases there is need of parallel computing with single processor pc's with standard RAM we cannot solve it. Think of our turbulent flow, the grid spacing required for smallest turbulent scale is Reynolds number, Reynolds number is a number which determines the physics of most of the simple flows we encounter flow of fluid. So, Reynolds number to the power 3 by 4. If Reynolds number is 10 to the power 6 for a 3 dimensional grid it needs 10 to the power 13 grids. For an unit box the grid spacing is a Reynolds number to the power minus 3 by 4, so for an unit box it is needs, in one direction it needs Reynolds number to the power 3 by 4 grids.

So, if the Reynolds number is 10 to the power 6 in considering 3 dimension it needs around 10 to the power 13 grid points, you can do this calculation yourself.

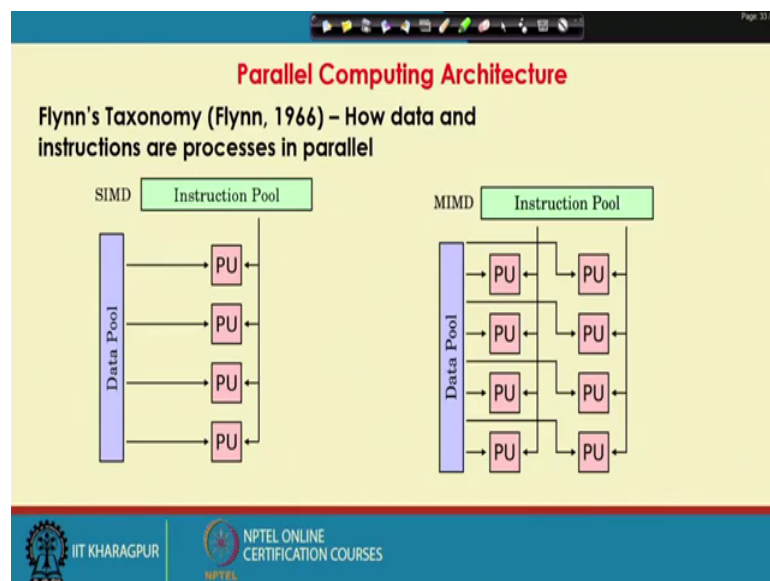
In one time steps, if we have to consider what happened to the flow in our small time delta T. We have to do a matrix solution that will need order of 10 to the power 13 floating point operations because, even with the first state solver the number of steps is of the order of n, number of operation is order of n, so order of 10 to the power 13 floating point operations. Typical physical duration of a time step is 10 to the power minus 3 seconds. So, we can computing from one second, but we have which one time step which is calculating for 10 to the power minus 3 seconds of the flow happening.

So, in order to simulate for 10 seconds of flow what will happen to the flow in 10 seconds at Reynolds number 10 to the power 6 to resolve the small smallest scale of turbulent motion will need 10 to the power 17 operation, 10 to the power 17 floating point operations. The first test computer gives 10 giga floating point operations.

So, estimated time will be 10 to the power 10 seconds or 317 years in order to solve flow for 10 seconds in an unit 1 meter long domain 1 cubic meter of fluid has to what is its turbulence level has to be estimated for 10 seconds. It will take in a computer with 10 giga flop per second speed it will take 317 years. So, what we will leave even nobody's student's student's student will be to look into this computing.

So, we really cannot do this, if we have to simulate a large problem like that with a single computer. So, we have to also the matrix size will be of the order of 10 to the power 13 floats, which is 10 to the power 13 into 16 bytes and it is much more than standard RAM size. So, forget about solving it 317 year say you write your will and ask your next generation to look into the simulation so that, the results are coming whether finally, you are getting some matrix solutions. Forget about doing that you cannot start the calculation because, the matrix you cannot stored in a single computer, in a single RAM is 10 to the power 13 into 16 bytes. So, you need to do something else.

(Refer Slide Time: 12:09)



And we will look into parallel computing, parallel computing has 2 parts that one there is a memory it stored somewhere and there are processing units which take care of the

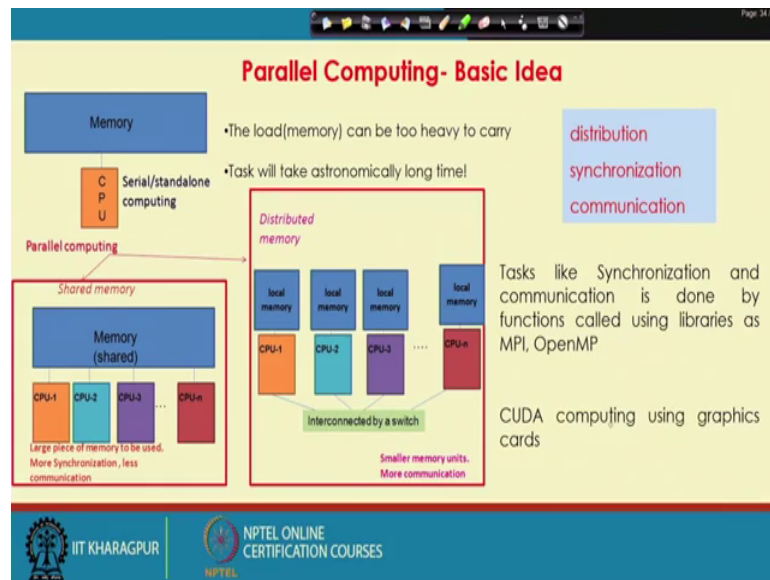
data given to it and follows the instructions. So, instructions are given to a processing unit, it takes reads from the data on the memory stored and works accordingly.

So, as per Flynn in 1966 give how data and instructions can be processed in parallel and will follow 2 different architectures we usually follow in matrix computing; one is Single Instruction Multiple Data or SIMD, another is Multiple Instruction Multiple Data or MIMD. What is in SIMD? There are different computers essential each computer is following, same set of instruction but they have different data. So, a large amount of data can be processed by different computers but in instruction in each computer is same. Then MIMD is that there are different instructions to different computers and there are different data which is going to different computer and they are processing this data.

This is mostly used in may domain decomposition algorithms this model that. There are number of computers, they are using different data and they are using they are doing different set of instructions that are given to them. However, these computers are connected to each other so that, they can communicate in between them. So, we will have lot of computers which will do some activity followed by the instruction that I have given with different part type of data.

So, different type of block matrices can be handled by different computers and they are talking to each other there is a switch by which these computers are connected, and they are sending some is, one computer is sending something to the main instruction pool which is asking another computer to do something like that. All one computer is sharing some data with other computer.

(Refer Slide Time: 14:38)



The basic idea of parallel computing is that the load memory can be too heavy to carry and the task will be, task might take astronomically long time that is what we have seen this cases. So, we have a large piece of memory which has to be taken in a computer by which is a standard serial or standalone computing. That is what we do, we have discussed till now that there is a memory there is a computer and it will do. In parallel computer it can be shared memory; that means, the memory is the same, you have a we need a large memory.

We now, there are number of computers, each computer is accessing the entire memory and it is accessing the memory elements, it is assigned to on that but they can see the entire memory. So, data has not it is not important to transfer data from one computer to other, each computer is working on the same memory this is called shared memory. The difficult thing is that you need a very large piece of memory and memories are you large RAMs are usually very expensive, much like 1 terabyte RAM is much more expensive then I will say 100 of 1 gigabyte RAM 1000's of 1 gigabyte RAM.

So, RAM this is much more expensive to get a large piece of memory. All in distributed memory each one has a small memory, each computer has a small memory and they are interconnected by a switch, which can take care of the memory which can take some of its memory and send it here or some of the instruction to send it there. The smaller a memory units are needed, but more communication because there is some overlap in

domain decomposition we have discussing and the overlapped data has to be exchanged, so more communication is required.

The essential bottleneck parts of parallel computing is that, except doing computation, computing for the matrix solvers for finding out the inverses etcetera doing Jacobi or Krylov subspace operation etcetera, it has to spend some time for distributing doing a domain decomposition and distributing different parts of the job to different processors. We also synchronize in between the processor, so, in while performing the operations then they have different processors as to communicate in between them.

Tasks like synchronization and communication is done by functions called using libraries called MPI OpenMP etcetera. There is another very new development here which is using graphics card for doing matrix parallel computing. A graphics card is a occurred with multiple very small processors with small memory, which can be connected with the CPU of the main computer, it can be added as a card to the in the motherboard. And small amount of calculation, so using small amount of data can be uploaded to the offloaded to the graphics card by the CPU.

And there can the interesting thing is that, 1 graphics card comes with 1000's of small processing units. So, we can have great infrastructure of paralyzing it, provided we break it down to really small pieces of activity the great infrastructure of paralyzing the job. However, we are not discussing about graphics card CUDA implementation, either we are discussing about MPI OpenMP implementation here, trying to this is the mathematics behind the row in decomposition and how can it lead to a parallel computation, parallel matrix solver.

(Refer Slide Time: 18:26)

Domain Decomposition

Distribute the domain into several subdomain

Form the matrix equation for each subdomain using the inter-boundary domain values

Propose an algorithm to solve each domain independently

Transfer inter-domain solutions to obtain continuity across the boundaries!

Converge to a final solution involving subdomains.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, we have a L shaped domain which is distributed into several subdomains. Ω is the main domain that is distributed into Ω_1 , Ω_2 and Ω_3 small subdomains and Γ_{13} and Γ_{12} are the overlaps between them. Distribute the domain into several subdomains form the matrix equation for each subdomain using inter boundary domain values. That is that if I try to write down the difference equation for a point here say, so these are the grid points here. So, for this particular i, j , if I try to write down the matrix equation, I need to use values here, here, here, here and here.

So, I need to use the boundary value; however, this boundary is pertinent to another subdomains. So, the values are not part of this particular subdomain or the processor attached to this particular subdomain is also it is a task the updating this value is a task of the processor associate to here. So, that is a complex issue which we will discuss now actually.

However, when we will form the matrix equation you have to form the matrix equation assuming that these inter boundary domain value is known to you. You have to consider this you cannot talking this. So, get the matrix equation for the inter subdomain including the off diagonal value that is coming due to this particular elements. Now you proposed an algorithm which can solve each domain independently. Solving independently each domain is difficult because of this particular boundary.

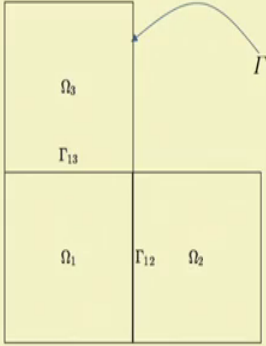
This the value here belongs to this domain as well as it belongs to this domain and when once we try to write an equation for a point here, we need a point from this particular domain as well as we need are coming from that particular domain. So, you have to think of we shall go and a block relaxation scheme where we can write separate some do something special for this.

Transfer the inter domain solutions to obtain continuity across the boundaries. So, once you get some solution here, you get some solution here you have to transfer them across the boundary so that, the solution is continuous across the boundary. It does not look like 150 to 50 degree centigrade temperature here and 100 to and 250 to 300 degree centigrade temperature here, there should be some continuity in the solution. And converge to a final solution involving all the subdomains. So, you have to get some solution, you will get some solution here, some here, some here and then you will iterate with the boundary values etcetera and then finally, get a converge solution

This is this is basically a block relaxation algorithm what we need here, but this block relaxation algorithm has to be distributed over different computers so that some algorithm can be proposed where each domain can be solved independently. At least some part of the solution can be carried out independently for each domain.

(Refer Slide Time: 22:14)

The matrix solution in decomposed domain



Solve: $\Delta u = f$ in $\Omega = \bigcup_{i=1}^3 \Omega_i$, $\Delta = \nabla^2$

With boundary conditions: $u = u_\Gamma$ on $\Gamma = \partial\Omega$

The matrix equation: $Au = b$

Let $x = \Sigma x_i$ be the solution at the domain-internal points and y be the solution in the inter domain boundaries Γ_ψ . Then $Au = b$ can be written as:

$$Au = b \Rightarrow \begin{bmatrix} B_1 & & E_1 \\ & B_2 & E_2 \\ & & B_3 & E_3 \\ F_1 & F_2 & F_3 & C \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ y \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ g \end{bmatrix}$$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, we have the matrix equation. This is Laplacian of u is equal to f, this particular sign is nothing, but nablus square, in the matrix, in the subdomain omega where, omega is

collection of all the small subdomains. With the boundary condition that on the Γ on the boundary well u is defined and once we convert this particular problem into difference equation, this differential equation, this is the differential equation, when once we convert into the difference equation you get a matrix equation. You have seen that several times that, a Laplacian equation can be converted into a matrix equation.

Let x is equal to sum of x_i , be the solution at the domain internal points, so x_1, x_2, x_3 at the domain internal point solutions, x is the solution and domain international point. And y be the solution at inter domain boundary. So, we will consider x_1 sorry x_1, x_2, x_3 at the solutions in inside the domain and y is these 2 net solution vector, x is the solution vector in each domain, so sum of x is the solution in the domain internal point and y are the solutions along the boundaries the inter domain boundaries.

So, now, we can write it as $Au = b$ where, A is the matrix which has block diagonal, which is a block diagonal form and there is something apart from block, what is that? That if I try to find out solution of for this x_1 It has some points, some neighbors which is in this particular boundary similarly a point here we have some neighbor in this particular boundary.

So, when we will write the equations for this x there is something which will be connect multiplied with the boundary inter domain boundary values. Similarly, if I try to write equation for the inter domain boundaries there is something which is with x_1 some neighbor x_1 some neighbor in x_2 some neighbor in x_3 . So, it will have all the off diagonal F_1, F_2, F_3 terms. So, once we write it as a block partition manner or the as a block matrix equation, the blocks are B_1, B_2, B_3 and C for x_1, x_2, x_3 and y and the inter block inter boundary connecting coefficient $E_1, E_2, E_3, F_1, F_2, F_3$ and we get a right hand side whatever will come we get a matrix equation like this.

(Refer Slide Time: 25:20)

The matrix solution in decomposed domain

$$Au = b \Rightarrow \begin{bmatrix} B_1 & E_1 \\ B_2 & E_2 \\ B_3 & E_3 \\ F_1 & F_2 & F_3 & C \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ y \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ g \end{bmatrix}$$



or,

$$\begin{cases} Bx + Ey = f \\ Fx + Cy = g \end{cases} \Rightarrow \begin{cases} x = B^{-1}(f - Ey) \\ (C - FB^{-1}E)y = g - FB^{-1}f \end{cases}$$

So, $FB^{-1}(f - Ey) + Cy = g$
 $(C - FB^{-1}E)y = g - FB^{-1}f \Rightarrow Sy = g - FB^{-1}f$
 $\Rightarrow y = S^{-1}(g - FB^{-1}f)$
 S (Schur Component)

So, finally the internal point solutions can be obtained as

$$\begin{aligned} x &= B^{-1}(f - Ey) \\ \text{with} \\ y &= S^{-1}(g - FB^{-1}f) \end{aligned}$$

So, $Au = b$ can be converted as a matrix equation $B_1, B_2, B_3, E_1, E_2, E_3, F_1, F_2, F_3, C, x_1, x_2, x_3, y, f_1, f_2, f_3, g$. Now my idea/question is that, how I can solve it independently in each computer. If y is some way known to me this can be distributed into this is the decoupled problem, B_1, x_1 plus $E_1 y$ is equal to f_1 , B_2, x_2 plus $E_2 y$ is equal to f_2 something. So, if y is known this is a solved problem, but y is not known y depends on x_1, x_2, x_3 . So, unless I solved for x in a coupled manner I cannot get a solution, let us see what will happen.

$Ax + y = f, g$ which is this is B , this block we take it this B , this blocks we take as A, F, E , this is as f and this is A, C, B, E, F, C, g, f or $Bx + Ey = f, Fx + Cy = g$. So, you get $x = B^{-1}(f - Ey)$ if just from this equation. And now, we will substitute this x , so from this equation we get $x = B^{-1}(f - Ey)$. And now we will substitute this Fx here, so you will get an equation for x that $FB^{-1}(f - Ey) + Cy = g$, $(C - FB^{-1}E)y = g - FB^{-1}f$.

Now, if we can solve this equation, see this equation which we are used solving for y this equation does not have any x component right that I can see here. This is a decoupled equation for y only. If I can solve this equation, I will get y and once I get y , I can solve all the x 's independently all the x_1, x_2, x_3 independently, it can be really a parallelizable equation. So, what is $C - FB^{-1}E$, how can it be solved?

This is called Schur component yes, this is an very important parameter in domain decomposition specially considering parallelization of a domain decomposition algorithm. Sy , so you have Sy is equal to $g - FB^{-1}f$ or y is equal to $S^{-1}(g - FB^{-1}f)$. This S^{-1} has to exist and has to have nice properties so that, even if we think of iterative schemes you can easily get S^{-1} .

So, finally, the internal point solutions can be obtained as x is equal to $B^{-1}(f - Ey)$ and with y obtained as $S^{-1}(g - FB^{-1}f)$.

(Refer Slide Time: 28:27)

Schur Component

In a domain decomposition problem, Schur component is defined as

$$S = (C - FB^{-1}E)$$

Solution at the internal points of different subdomains are found as:

$$x = B^{-1}(f - Ey), y = S^{-1}(g - FB^{-1}f)$$

If S^{-1} exists, y can be found and hence x can also be found

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, in a domain decomposition problems Schur component is defined as S is equal to $C - FB^{-1}E$. Solution at the internal problems of different subdomains are found as x is equal to $B^{-1}(f - Ey)$ and y is equal to $S^{-1}(g - FB^{-1}f)$. Once you can write it like that, this part decoupled problems; that means, finding y does not need any of the x interestingly what we can say. So, x has been eliminated here. So, if we can solve it we can find y and then we can give y to different processors and try to solve x . If S^{-1} exists, y can be found and hence x can also be found. So, the domain decomposition method will stand if S^{-1} exists. That is that, if this is done then domain decomposition stands for any decomposition will stand if we can find an S which is which is invertible

(Refer Slide Time: 29:45)

Domain decomposition parallelization

To solve: $x = B^{-1}(f - Ey), y = S^{-1}(g - FB^{-1}f)$

B is a block diagonal matrix,

$$Au = b \Rightarrow \begin{bmatrix} B_1 & & E_1 \\ & B_2 & E_2 \\ & & B_3 & E_3 \\ F_1 & F_2 & F_3 & C \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ y \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ g \end{bmatrix}$$

Hence, inverse of B can be found in a decoupled sense as disjoint processes.

Hence the sets of equation given can be solved, provided y is made available to the particular process.

To solve x is equal to B inverse f minus $E y$ and y is equal to S inverse g minus $F B$ inverse f there is a problem in the matrix form which you have to get now and we see that whether this is parallelizable term. B is a block diagonal matrix, this part is B of A u is equal to B .

Inverse of B can be found in a decoupled sense as a disjoint point because, this is the block diagonal matrix B inverse can be easily found out. Hence the sets of the equation can a given can be solved provided y is made available to the particular process. This can be solved independently in different processors provided y is also already known to the processors.

(Refer Slide Time: 30:31)

Page 41/42

Schwarz Alternating procedure

Alternate between the domains for solution. Solve Dirichlet problem on one domain in each iteration and consider boundary conditions based on the most recent solution of the other domains.

Algorithm

1. Choose an initial guess u to the solution
2. Until convergence Do:
3. For $i = 1, \dots, s$ Do:
4. Solve $\Delta u = f$ in Ω_i with $u = u_{ij}$ in Γ_{ij}
5. Update u values on Γ_{ji} , $\forall j$
6. EndDo
7. EndDo

$\Delta u = \nabla^2 u$

- The algorithm sweeps through the s subdomains and solves the original equation in each domain based on the boundary conditions that are updated from the most recent values of u .
- We can start with a global initial guess and update it in each domain during the iterations

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

And one method for that it Schwarz alternating procedure alternate between the domains for solution, solve Dirichlet problem on one domain in each iteration, solve Dirichlet problem on one domain in each iteration and consider boundary conditions based on most recent solution of other domains. So, essentially you start with some guess value of u , consider that the problem in one domain is completely Dirichlet all other boundaries are known and that inter interconnected boundaries guess value as is the Dirichlet value, so solve it. And then you solve it for the other domains and again based on the solution update u , by solving something like Schur component equation and repeat it.

So, the algorithm is choose an initial guess u until convergence in each domain solve $\Delta u = f$ with the boundary conditions in need in the inter block boundary, is inter domain boundaries and this will come from the initially from the guess solution. First step it will come from the guess solution and later update the u values of the boundaries and find repeat it till convergence.

But, we have to see whether this process converges actually because, this is not like our direct solution method, not like standard block Jacobi iteration; you have to see whether this convergence. And this is Laplacian of u Δu that Δu is Laplacian of u , the algorithm sweeps through a subdomains and solves our original equation in each domain based on the boundary condition that is updated from most recent values of u .

So, when we are solving for each domain we are using a boundary condition which is coming from our other domains and that is the most recent value of u available to the processes. We can start with a global initial guess and update it in each domain during the iterations and then we have to check inter change that we have to send the boundary condition from one domain to other to maintain continuity and updates.

(Refer Slide Time: 32:54)

The slide is titled "Schwarz Multiplicative Procedure for Overlapping Domains". It features a diagram on the left showing two overlapping rectangular domains, labeled Ω_1 and Ω_2 . The boundaries of these domains are marked with $\Gamma_{1,2}$ and $\Gamma_{2,1}$. The intersection of the two domains is also indicated. To the right of the diagram is the "Algorithm" section, which lists five steps:

1. For $i = 1, \dots, s$ Do:
2. Solve $A_i \delta_i = r_i$
3. Compute $x_i := x_i + \delta_{x,i}$, $y_i := y_i + \delta_{y,i}$, and set $r_i := 0$
4. For each $j \in N_i$ Compute $r_{y,j} := r_{y,j} - E_{ji} \delta_{y,i}$
5. EndDo

At the bottom of the slide, there are logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES.

For overlapping domains, so this is an example of overlapping domain 1 has an overlap up to so, this is still domain 1 is there, domain 1 has overlap in domain 3, some part in domain 3. Similarly, domain 3 has some part in domain 1, this is an overlap, this is the example of an overlapping domain. For overlapping domain, we can use something Schwarz multiplicative procedure that solve in each domain solve $A \delta$ is equal to r r is the residual compute x such that, x is updated with x plus δ x i in that particular domain and also update y . Now the inter domain boundary, say the boundary of domain 1 is a member of domain 3, the boundary of domain one is a member of domain 3.

So, how will it be updated, it will be updated by the solution, whatever we are getting as the internal point solution of domain 3, that will go and update the boundary for domain 1. So, computed the solve the inter domain problems independently and update the boundary values for the other domains. And then calculate the residual for each domain use using the already known residual, residuals and the E vector, E is coming from the boundaries again. So, update the residual in each domain.

(Refer Slide Time: 34:25)

Page 43/47

Schwarz Multiplicative Procedure - Steps

1. Chose an initial guess u to the solutions
2. Iterate until convergence
3. For $i=1,\dots,s$
4. Solve $\Delta u=f$ in Ω_i with $u=u_{ij}$ in Γ_{ij}
5. Update u values in Γ_{ij}
6. Till convergence in all $\Omega=\Omega_1,\Omega_2,\dots,\Omega_s$

} Convergence?

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

The idea is if I ready to explain the steps that choose an initial guess u to the solutions. Iterate till you get convergence for in each domain solve the Poisson Laplacian of u is equal to f with the boundary condition which is coming from the other domains. Update the values at the boundaries for the other domains.

And till convergence you do it in all the subdomains till you reach a global convergence. And when and this step can be done parallel in different computers because, u_{ij} is the gauss value which is going into different computers. Now, we have to see that whether, this step actually converges what is the convergence of this particular step.

(Refer Slide Time: 35:26)

Theorem for Convergence of Schwarz Procedure

If the guess $\begin{pmatrix} x^{(0)} \\ y^{(0)} \end{pmatrix} = u^{(0)}$ is chosen as $x^{(0)} = B_1^{-1}[f_1 - E_1 y^{(0)}]$ then the iterations are identical to Gauss-Seidel sweep of the Schur component and they converge!

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

And this the theorem of convergence for Schwarz procedure is given that, if the initial guess u_0 is chosen such that the internal vector x_0 is as obtained from the solution of the block per of the block relaxation of the matrix. Then the iterations are identical to Gauss Seidel sweep of Schwarz Schur component and if Schur component exist they must converge.

(Refer Slide Time: 35:56)

Domain Decomposition based parallel Matrix Solver

1. Divide the domain into a number of subdomains. Domain overlaps are allowed such that the full row equation for each internal point of the subdomain is available
2. Start with a global guess
3. Update the solution at every subdomain locally. Consider the inter-domain boundaries as Dirichlet with the last updated solution value – **parallel step**
4. Update the boundary values in one sub-domain as obtained by local solution of neighbouring domains. – **Data transfer step**
5. Iterate over the domains for a global convergence – **synchronization step**

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, domain decomposition should converge if we can choose the initial guess according to the calculation of x based on the values y . So, if the initial guess is consistent with the

equation system, the domain decomposition is the Gauss Seidel sweep of Schur component and they must converge.

The idea is divide the domain into number of subdomains, domain overlaps are allowed such that, full row equation for each internal point of the subdomain is available. Start with a global guess. Update solution at each subdomain locally, considered the inter domain boundaries as Dirichlet with the last updated solution value. And this is a parallel step that updating each subdomain locally. Update the boundary values in 1 subdomain as obtained as by the local solution of neighboring domains, this is the data transfer step, so from one domain to other you have to transfer data. Iterate over the domain, still you get a global convergence if to synchronize the residuals have to obtain residuals from each domain and send it to one particular computer and it has to check over different values of the residuals, what is the global value of residual whether, it has converse this is the synchronization step.

(Refer Slide Time: 37:09)

Elements of a parallel program

- ✓ Initialization of Parallel environment
- ✓ Allocation of decomposed domain to the processors
 - load balancing, idle time minimization
- ✓ Calculation in each domain
- ✓ Synchronization
 - latency
- ✓ Communication
 - avoiding bottleneck or deadlock in data exchange!
- ✓ Assembly of results
- ✓ Termination

overheads due to initialization, synchronization and communication

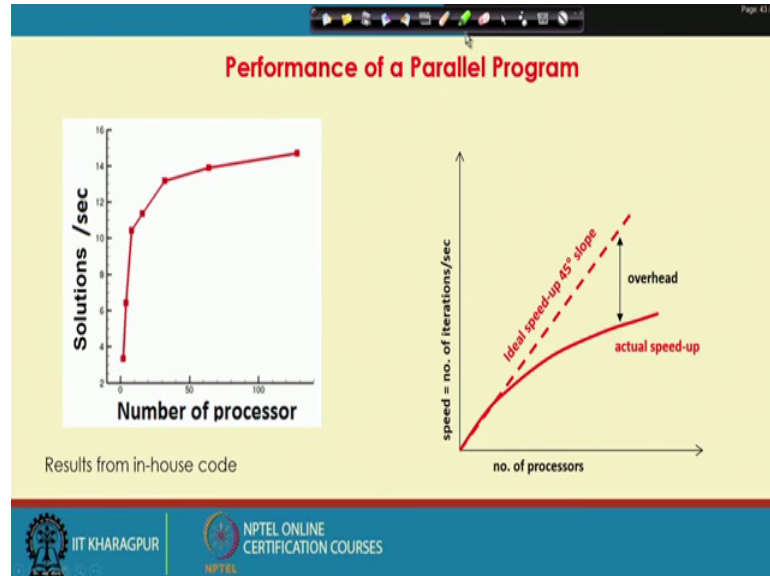
The slide features a 3D diagram of a rectangular domain divided into a grid of smaller, colored subdomains. The subdomains are arranged in a 4x4x4 grid, with colors alternating between blue, purple, green, and yellow. This visualizes the concept of domain decomposition for parallel processing.

The slide footer includes the IIT Kharagpur logo and the NPTEL Online Certification Courses logo.

So, the elements of parallel program, when we have distributed the geometry in to different sub domains, initialization of parallel environment, allocation of decomposed domain to the processors, which needs load balancing, idle time minimization, calculation in each domain synchronization to reduce latency. Now we will see latent all the computers will do almost similar amount of job, now we will say it idle. Communication avoiding, there can be bottleneck in communication, one is trying to

send another is also trying to send there can be a bottleneck, so avoid that, assembly of results and termination.

(Refer Slide Time: 37:51)



Overheads come due to initialization synchronization and communication. And this is our in house code where number of solutions obtained per in per second. So, you got in 1 second how many matrix can be matrix solutions can be obtained for 1 particular large matrix.

And we have seen that as the number of processors are increasing up to 50, this is in the number of solutions in one second is increasing speed of the computation is increasing and that then it is going kind of flat because, communication where it has been increased so much that, there is no increase in the performance or computational speed which is number of iteration per second ideally as you increase the number of processor it should follow up 45 degree slope, it will also increase. It increases, but it does not follow the 45 degree slope due to the overheads which is due to communication synchronization and latency. So, try to explain some aspects of parallel computing using domain decomposition in this particular lecture.

Thank you.