High Performance Computing for Scientists and Engineers

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

Module - 01
Fundamentals of Parallel Computing
Lecture - 07
Parallel Algorithms

Hello, we are discussing different topics in the class of High Performance Computing for Scientists and Engineers and this is the 1st module which is Fundamentals of Parallel Computing. This is the 1st lecture of the 2nd week or 7th lecture of this course.This lecture is on Parallel Algorithms.

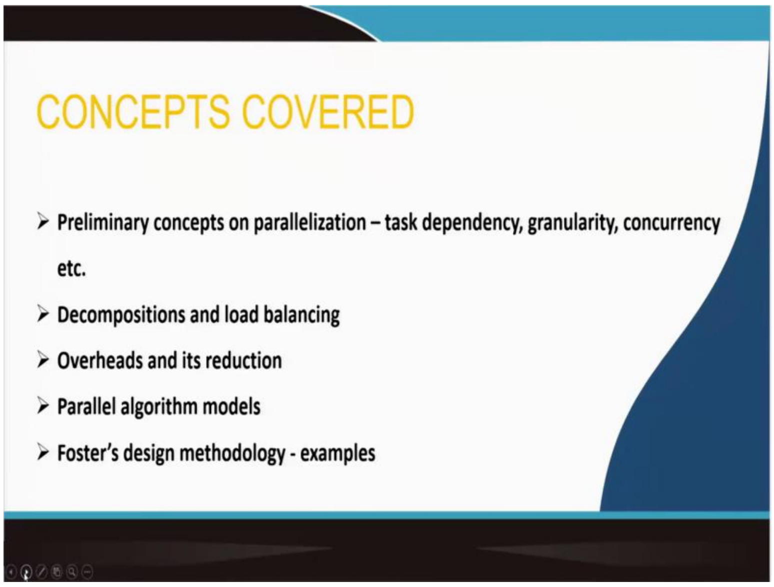
Now, in the previous week, we have discussed about architectures for parallel computing, we read about different hardware supports, which are required to establish a parallel computing environment .We have also discussed about shared memory and distributed memory programming using parallel computing.

Now, we can understand that in order to utilize all the benefits of the hardware and the system level support which is given to us for doing parallel computing, we also need to do something as a programmer. If an algorithm is given and we just write a simple C program and run it in a very high-end high-performance computing infrastructure, we will not get any benefit. We need to parallelize it so that, we can get benefit of running parts of the program or parts of the algorithm concurrently in multiple processors.

And therefore, if we look into the algorithm itself, certain modifications are to be done over the algorithm. We need to find out what are the parts of the algorithm which can be parallelized and how this should map to different processors and how do these processors communicate it between themselves, how will they synchronize. And we need to look from that perspective into that particular program, when we are talking about parallel programs or when we are talking about paralyzing a particular scientific computing algorithm.

So, this particular lecture and the subsequent couple of lectures, we will discuss about this parallel algorithms how can we develop parallel algorithms and see some of the examples by which things will be more clear to us that given a particular problem, how can we parallelize it and how can you ask different processors to work on it.

(Refer Slide Time: 02:53)



CONCEPTS COVERED

- Preliminary concepts on parallelization – task dependency, granularity, concurrency etc.
- Decompositions and load balancing
- Overheads and its reduction
- Parallel algorithm models
- Foster's design methodology - examples

The topics that we will cover in this particular lecture and subsequent couple of lectures are preliminary concepts of parallelization, we will see about task dependency, granularity, concurrency etcetera. We will look about decompositions and load balancing and we will see that how we can reduce the overheads when we parallelize a particular algorithm.

We will we have already discussed about some things like communication time, synchronization overheads etcetera. So, when we parallelize a particular algorithm, the computers as an aggregated manner perform something more than the simple sequential algorithm. There are jobs of distributing the parts of the program into different processors, mapping it as different processors, synchronizing in between them, then doing communication etcetera and all these things add certain overheads on that.

So, how can we reduce the overheads and I am telling you that as a programmer or as an user of HPC systems also, when you develop a program or when you execute a program, one of the primary goal for you is that how can you do it more efficiently, how can you optimize all the overheads and latency in the program and can get an efficient program so that, your computation is done in the first test way.

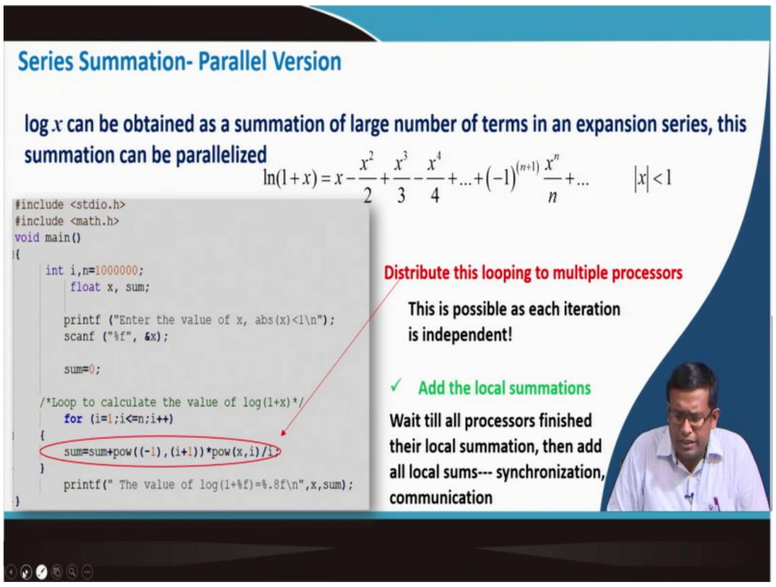
So, therefore, this particular part which is understanding the overheads and its reduction will be very important and we will discuss about these things over and over when we will see some of the examples, when we will go through some of the problems and later, when we will learn about message passing interface or OpenMP or CUDA programming API’s. We will also see for different infrastructures like distributed memory systems or shared memory systems what are overheads and what are the best ways to contain the overheads.

So, this is a very important step and this week after finishing the particular discussion on parallel algorithms, we will also look into performance of the parallel systems. And there we will also see that ,this is that particular thing which you are looking in that what is the overhead when we have parallelized that algorithm and that is reducing the efficiency of the program if you what it is high, then the efficiency is restricted or reduced. So, we need to reduce overhead that we will put certain emphasis on understanding that part.

Then we will see parallel algorithm models, what are the standard models of parallelizing different scientific computing algorithm and we will see a very generalized design methodology called fosters design methodology by which given a problem if you can utilized fosters design methodology and you can get a parallel implementation out of that and we will see some of the examples again on the field of scientific computing only.

So, these are the focus points of this discussion and subsequent discussions and we will start with some preliminary concepts of parallelization.We will see what are the need of making a program parallelized and what are the ways to get that done.

(Refer Slide Time: 06:25)



Series Summation- Parallel Version

log x can be obtained as a summation of large number of terms in an expansion series, this summation can be parallelized

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots + (-1)^{(n+1)}\frac{x^n}{n} + \ldots \qquad |x| < 1$$

```
#include <stdio.h>
#include <math.h>
void main()
{
    int i,n=1000000;
      float x, sum;

      printf ("Enter the value of x, abs(x)<1\n");
      scanf ("%f", &x);

      sum=0;

   /*Loop to calculate the value of log(1+x)*/
      for (i=1;i<=n;i++)
   {
      sum=sum+pow((-1),(i+1))*pow(x,i)/i;
   }
      printf(" The value of log(1+%f)=%.8f\n",x,sum);
}
```

Distribute this looping to multiple processors

This is possible as each iteration is independent!

✓ Add the local summations

Wait till all processors finished their local summation, then add all local sums--- synchronization, communication

So, we consider program in which log(1+x) is found using series summation of

$$\log(1+x) = \frac{x}{1} - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots, \qquad |x| < 1$$

So, it is an infinite series, if we can sum up this infinite series, we find log(1+ x). Now, in when doing numerical computation, you really cannot sum up an infinite series so, you what you can do is you can calculate up to large number of values of terms in that series and sum their values.

So, we thought of writing a program where we will consider 1 million terms of this series and it will be summed up well. So, this is the program and now what we can see that as we are summing up over 1 million terms of this particular expression.

So, every for each of each values of i we are calculating a value $(-1)^{n+1}\frac{x^n}{n}$ and so, given the value of x, for each value of i we are finding out $(-1)^{i+1}\frac{x^i}{i}$ and adding it and we are doing this addition for million times.

So, we can see that these are large number of steps which a computer is performing. If we have to parallelize it this particular program that we have to find out log(1+x) as sum of this series and we have to parallelize it, what we will do we is will take this for loop and as different processes to calculate up to say will as 1000 processes each of them will calculate up to 1000 terms.

So, there will be concurrently 1000 processes going on and therefore, each is calculating 1000 so, the computing time is taking 1000 by each of the processor compared to 1 million calculation and it will be fast, that is the basic idea..

So, if we distribute this particular loop into multiple processors, then we can parallelize this part and then what we have to do? Each processor will come up with its own sum value. So, each one will work ,say first one will probably work 1 to 1000, second one will work 1001 to 2000 and so on, the 1002 will work 99000 to 1 million.

So, each processor will come up with its local sum, each processor will do this sum for small number of values assigned to it and we come up with its local sum and then, we have to add all the local sums to get the global sum. The global sum is the sum of the series for 1 million elements.

So, there are 2 steps, first step will be distributing this into multiple processors this particular loop and then add the local summations. Now, this is possible that I can distribute this loop over multiple processors and each loop can independently calculate a local sum.This is possible only due to the fact that the calculations in each of the processor is independent.

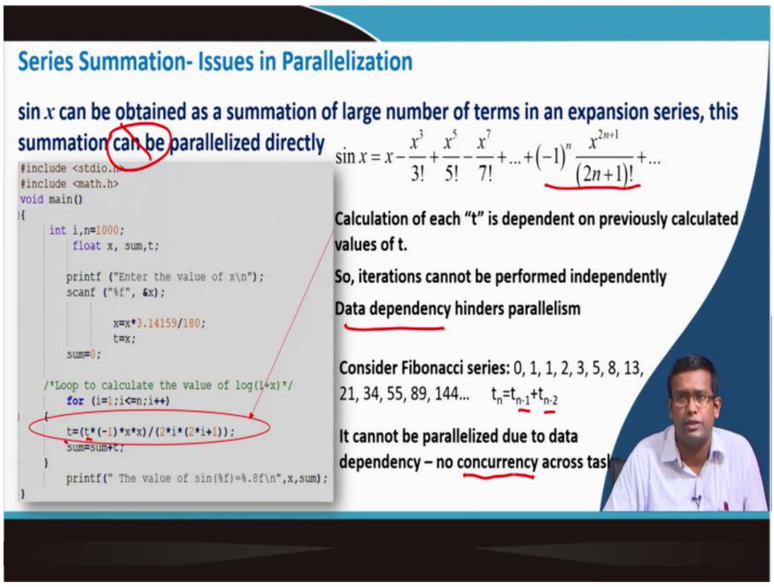
So, if I calculate the first term of the say for i is equal to 1, if I calculate the first term of this expression, this is independent from 1001 term. If I even do not know the first term of this expression, I can just put n is equal to 1000 here and can calculate the 1000 term of this expression.

So, calculation is really independent for each value of i. In case, the iterations are dependent so, it is a reduction operation you will see later, this would not have been such simple and then, we need to add the local sums therefore. We have to see that each processor is finishing its own job and then, we take values from all the processors and add these values.

So, these are the 2 important steps, one is distributing the jobs into multiple processors and that is possible because the calculations in each processor is independent, there is no dependency among the calculations in each processor.

And then, wait until all the processors finish their summation and then add up the local summations .And this part require synchronization so, one processor needs to know that other processors are finished and then only it can send its value to some other processor and that processor will gather values from all the remaining processors and will add it up.

(Refer Slide Time: 11:48)



Series Summation- Issues in Parallelization

sin x can be obtained as a summation of large number of terms in an expansion series, this summation can be parallelized directly

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots + (-1)^n \frac{x^{2n+1}}{(2n+1)!} + \ldots$$

```
#include <stdio.h>
#include <math.h>
void main()
{
    int i,n=1000;
        float x, sum,t;

        printf ("Enter the value of x\n");
        scanf ("%f", &x);

                x=x*3.14159/180;
                t=x;
        sum=0;

    /*Loop to calculate the value of log(1+x)*/
        for (i=1;i<=n;i++)
    {
        t=(t*(-1)*x*x)/(2*i*(2*i+1));
        sum=sum+t;
    }
        printf(" The value of sin(%f)=%.8f\n",x,sum);
}
```

Calculation of each "t" is dependent on previously calculated values of t.

So, iterations cannot be performed independently

Data dependency hinders parallelism

Consider Fibonacci series: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144... $t_n=t_{n-1}+t_{n-2}$

It cannot be parallelized due to data dependency – no concurrency across tas

Now, we take another example, that is sin(x). Sin(x) can be obtained also a summation of large number of terms in that given expression series . And the problem is this cannot be parallelized directly why? can it be parallelized directly. So, this is this remains a question, this actually we can see that it cannot be parallelized directly why is it so?

We can understand that if I look into the sin series, it involves a calculation of a factorial. How do we calculate factorial? We calculate factorial through a reduction operation. So, if we look into the program, that the you go to the for loop for calculating some of the series, each term of the for loop requires the last step which is t last updated value of t.

Therefore, if we have to calculate the value for i = 2, value of i = 1 should already be known to us. If we have to calculate value for i =10, value of i = 1 to 9 must be available to us. So, we cannot ask different processors to do different parts of the calculation independently.

If you use this particular algorithm, you cannot do it in an indirect way there are methodology of paralyzing this, but if we see direct parallelization which I have written here a direct parallelization is not possible.

Calculation of each t is dependent on previously calculated values of t and therefore, the iterations cannot be performed independently. If I need to know the value of t =1050, I need to know values of t for all 1049 terms, then based on 1049 values of t, I will find out the 1050. So, this is like a recursion algorithm and recursive algorithms cannot be parallelized because they are not independent.

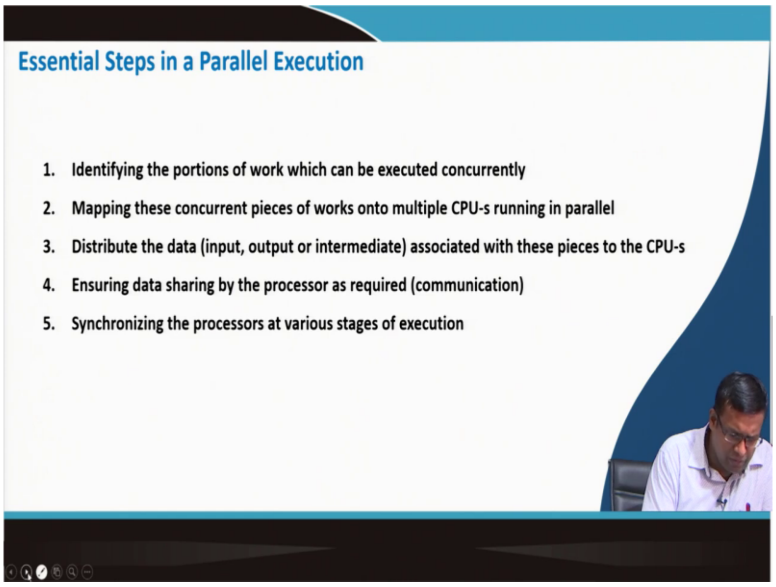
So, while parallelizing a particular algorithm, it is important to find out that if we can get independent tasks in that particular algorithm.These independent task can be given to different processors. At least in some sense, they have to be independent or quasi independent so, we can allocate different task to the different processors , which will not interact in between them. We will look into this things also later.

Therefore, what we can say that data dependency hinders parallelism. And now, you can consider Fibonacci series which starts from 0 goes up to infinite term .The nth term of Fibonacci series is summation of n-1 and n-2 th term. Therefore, none of this term we can calculate independently. So, this cannot be parallelized.

What is in parallelization? Parallelization means different processors are acting at same point of time and taking care of different part of this job. Now, if one particular job is completely dependent on the previous calculations, it cannot be done in a separate processor, when the previous calculations are done in another processor .If it is done, then this processors need to communicate too much in between them which is not advisable.

Therefore, parallelization of data dependent problems is difficult and specially something like Fibonacci series or something like a recursion algorithm, when the particular iteration depends on the values of the previous iterations is actually very difficult to do and it is not possible to parallelize it. It cannot be parallelized due to data dependency and no concurrency across the tasks. Calculation of t =100 and calculation of t= 1500 cannot be done concurrently; there is no concurrency among the tasks.

(Refer Slide Time: 16:40)



Essential Steps in a Parallel Execution

1. Identifying the portions of work which can be executed concurrently
2. Mapping these concurrent pieces of works onto multiple CPU-s running in parallel
3. Distribute the data (input, output or intermediate) associated with these pieces to the CPU-s
4. Ensuring data sharing by the processor as required (communication)
5. Synchronizing the processors at various stages of execution

So, we try to identify what are the essential steps in a parallel execution. First and foremost, we need to identify the portions of work which can be executed concurrently. For logarithmic series, the iterations can be executed concurrently and each of these iteration can ideally go to a different process.

But for sin series, we saw that they cannot be executed concurrently if we write the program in that particular way which I have shown you. Fibonacci series the terms cannot be found out concurrently. So, this cannot be parallelized. So, for parallelization, the first and foremost important part is that find out which are the steps which can be executed concurrently.

Then of course, you need to map this concurrent pieces of works into multiple CPU’s running in parallel and that is different CPU’s will execute different parts of the program so, you need to map them accordingly.

And then, distribute the data associated with these pieces to the CPU, it can be input output or intermediate data .We will see how the data decomposition is done, how different CPU’s are assigned to a particular data and they do calculation pertaining to that particular piece of data.

So, it is called a data parallel model that different part of data is being considered by different CPU’s, but there are other parallelization model also. So, this data distribution is one of the methods of parallelization. In case, you can have a large pool of data which can be distinctively computed by different processes so, you can do that then .

In many times, you need to communicate across the processor. Though different processors are working on different parts of the data, in many time it is important that they need to share some of the data and you need to send their local results ,like if you think of that log(1+ x) calculation and has different processors to take part of different values of i. At the end some of the processors will have their local values, rather each processor will have its local value which is not the final sum.

So, these local values are to be combined by one processor. So, local summation from each of the processor has to be communicated and this requires a communication step. So, which is also important and when writing the program, it the programmer’s duty is to find out the communication steps.And then its synchronization is important in various cases. Again we come to that point that we have the local sums in each processor and they need they need to find out the global sum.

So, each processor has to wait till others have finished the local sum and only when the local sums are available with all the processors, it can be communicated to one of the processor who will find out the global sums.

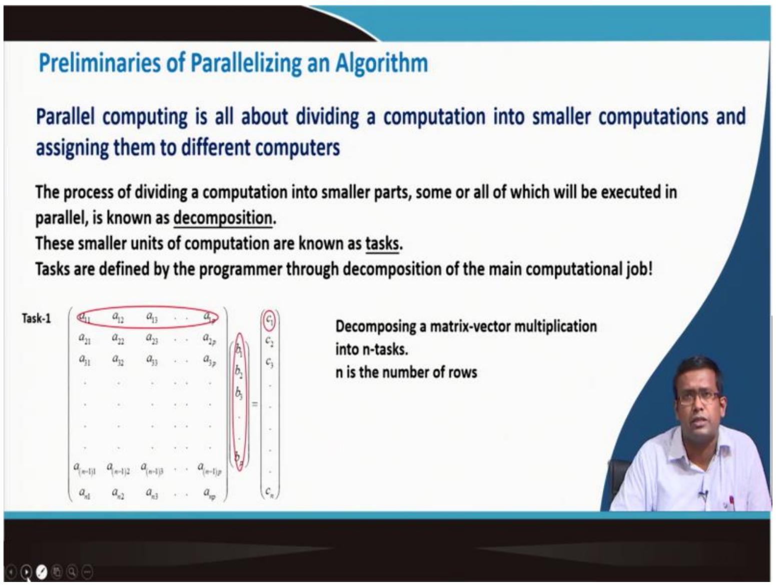
So, these are very important steps and as I say the first and foremost task of parallelizing and algorithm is identifying the portions which need to do work. We will see all these steps that how can how we can identify these things, the concurrent parts, what is the best way to identify concurrent parts.

They comes the mapping part, what is important in mapping this the work in different processor, what are the; what are the important considerations here? One of the important consideration is that when you are mapping these two different processors, one point which has to be taken care is that, that all the processors are getting almost same amount of data work.

If some processor gets more work, some processor gets less work we call that to be a problem in load balancing and if the load is imbalanced, one processor will finish its work and because there is a synchronization step, it will weight for the other processor to finish its work so, it will incur some latency on that processor and that will act to overhead.

So, mapping the piece of work into multiple CPU’s and is important as also the load balancing has to be done correctly there and then again data distributing the data . It requires both mapping this can also give you a mapping technique that the data is evenly distributed to different processors so, you get a write load balancing and data sharing and synchronization issues we have discussed well.

(Refer Slide Time: 21:35)



Preliminaries of Parallelizing an Algorithm

Parallel computing is all about dividing a computation into smaller computations and assigning them to different computers

The process of dividing a computation into smaller parts, some or all of which will be executed in parallel, is known as <u>decomposition</u>.
These smaller units of computation are known as <u>tasks</u>.
Tasks are defined by the programmer through decomposition of the main computational job!

Task-1

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1p} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2p} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3p} \\ \vdots & \vdots & \vdots & & \vdots \\ a_{(n-1)1} & a_{(n-1)2} & a_{(n-1)3} & \cdots & a_{(n-1)p} \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{np} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_p \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_n \end{pmatrix}$$

Decomposing a matrix-vector multiplication into n-tasks.
n is the number of rows

So, now we understand that parallel computing means you have to divide a large computation into smaller computations and assign it to different computers or processors. What you can do is that, you can identify each of the small computing tasks to one of the processor or if you do not have large number of processors depending on the number compared to the number of small computation tasks you got, you can give a group of computation to different processors.

The process of dividing this computation into smaller parts and sum of all or all of which will be executed in parallel is known as decomposition.The smaller unit of computation is known as tasks. When the smaller units of computation is same as the number of processors, then each processor is executing one task and multiple processors are working in parallel so, all the tasks are being executed in parallel.

In case, you have number of processor less than number of tasks well which is many times possible because your hardware is limited , your resource is restricted by the hardware which you are having. So, you found out that you can decompose it into 10000 tasks and the processors there are 1000 processors.

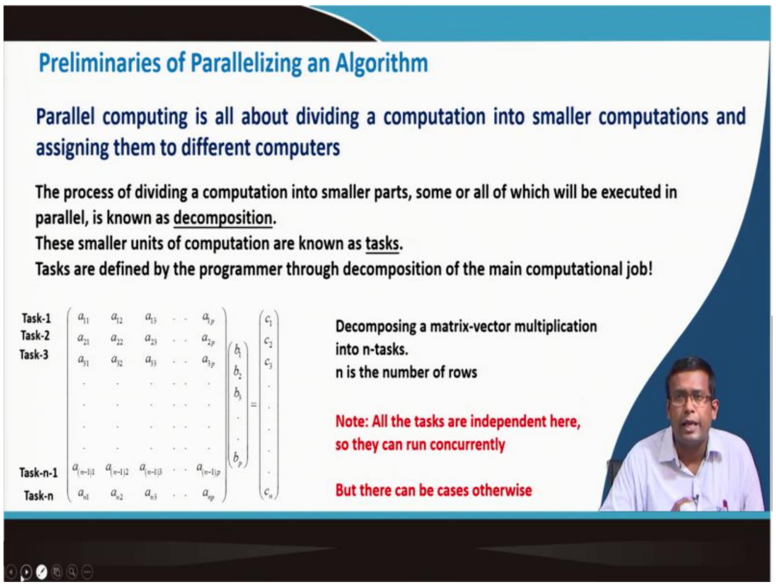
So, each processor will get 10 tasks and this 10 task within a processor will be executed serially, but at any point of time 1000 of this tasks when all the 1000 processors has being executed in parallel. So, the smaller parts or the task can be executed in parallel or all (Refer time: 23:23) I mean they also can be executed in serially, but some of these tasks will always be executed in parallel, if you are using a parallel computing architecture.

The tasks are identified by the programmer through decomposition of the main computational job and this is again, programmers job to identify that this part can be parallelized there can be multiple tasks which can be executed in parallel in different processors and when can it be done? When these task are independent and this is programmers job to look into the algorithm and find out the task and then do the right decomposition.

So, we can think of a matrix vector multiplication and if there are n rows then, calculation of the vector, vector dot product for each row which will be the element of the column vector in the matrix vector multiplication. So, this row multiplied by this column gives me this column, this row multiplied by this column gives me this column.

All these element in the right hand side column can be found out independently. So, if there are n rows, there can be n task. Each task is finding out for one element on the right hand side vector like this is my task one and we can decompose it into n number of tasks.

(Refer Slide Time: 24:57)



Preliminaries of Parallelizing an Algorithm

Parallel computing is all about dividing a computation into smaller computations and assigning them to different computers

The process of dividing a computation into smaller parts, some or all of which will be executed in parallel, is known as decomposition.
These smaller units of computation are known as tasks.
Tasks are defined by the programmer through decomposition of the main computational job!

$$\begin{array}{l} \text{Task-1} \\ \text{Task-2} \\ \text{Task-3} \\ \\ \\ \\ \text{Task-n-1} \\ \text{Task-n} \end{array} \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdot & \cdot & a_{1p} \\ a_{21} & a_{22} & a_{23} & \cdot & \cdot & a_{2p} \\ a_{31} & a_{32} & a_{33} & \cdot & \cdot & a_{3p} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ a_{(n-1)1} & a_{(n-1)2} & a_{(n-1)3} & \cdot & \cdot & a_{(n-1)p} \\ a_{n1} & a_{n2} & a_{n3} & \cdot & \cdot & a_{np} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \cdot \\ \cdot \\ b_p \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ c_n \end{pmatrix}$$

Decomposing a matrix-vector multiplication into n-tasks.
n is the number of rows

Note: All the tasks are independent here, so they can run concurrently

But there can be cases otherwise

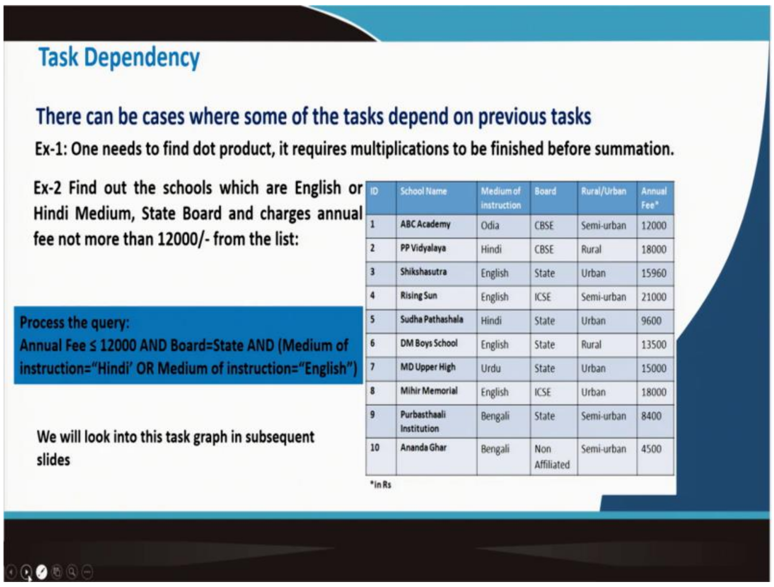
Similarly, is there are n total number of n operations, there can be n number of tasks. Now in case, I have two CPU’s, I am working in a dual core computer, I have two CPU’s. So, each CPU will get n by 2 amount of tasks.

In case, I this is a 100 by 100 matrix and I am working into 100 by some 100 row matrix I am working into a 100 CPU system, I can actually launch 100 tasks here. So,how many processors will execute how many task that is a different issue which will be taken care of by the available hardware.

The number of tasks, are the number of independent instructions, that I can find out over the process and that is called the decomposition that taking out the main computation and decomposing it into the number of tasks. All task are independent here,so, they can run concurrently. But there can be cases when not all the tasks are independent.

So, what will you do? We can find out smaller units of computation, but these computations are not always independent. Well we are we will find out how many are independent and ask them to work in different processors and then, which are dependent we look into them later something like that we will see some examples here and that is called a task dependency.

(Refer Slide Time: 26:34)



Task Dependency

There can be cases where some of the tasks depend on previous tasks

Ex-1: One needs to find dot product, it requires multiplications to be finished before summation.

Ex-2 Find out the schools which are English or Hindi Medium, State Board and charges annual fee not more than 12000/- from the list:

ID	School Name	Medium of instruction	Board	Rural/Urban	Annual Fee*
1	ABC Academy	Odia	CBSE	Semi-urban	12000
2	PP Vidyalaya	Hindi	CBSE	Rural	18000
3	Shikshasutra	English	State	Urban	15960
4	Rising Sun	English	ICSE	Semi-urban	21000
5	Sudha Pathashala	Hindi	State	Urban	9600
6	DM Boys School	English	State	Rural	13500
7	MD Upper High	Urdu	State	Urban	15000
8	Mihir Memorial	English	ICSE	Urban	18000
9	Purbasthaali Institution	Bengali	State	Semi-urban	8400
10	Ananda Ghar	Bengali	Non Affiliated	Semi-urban	4500

*In Rs

Process the query:
Annual Fee ≤ 12000 AND Board=State AND (Medium of instruction="Hindi' OR Medium of instruction="English")

We will look into this task graph in subsequent slides

There can be some of cases where some tasks dependent previous tasks. If you remember when we talked about the logarithm finding a problem log (1 + x).All the processors are finding out their local sums, but finding the global sum is not an independent ,task we have to wait till all the processors are found out their local sum. And then, we ask at least one processor to take local sum from all the processors and sum it up. So, that was a dependent task.

Similarly, if one has to find out dot product, each element of the left hand side vector and the corresponding column row element of the right hand side vector are to be are to be multiplied. And a_1b_1, a_2b_2, these multiplications can be done independently, but finally, for dot product we have to find out the summation that cannot be done independently. So, there is a at least one task which is waiting for the other task to get finished.

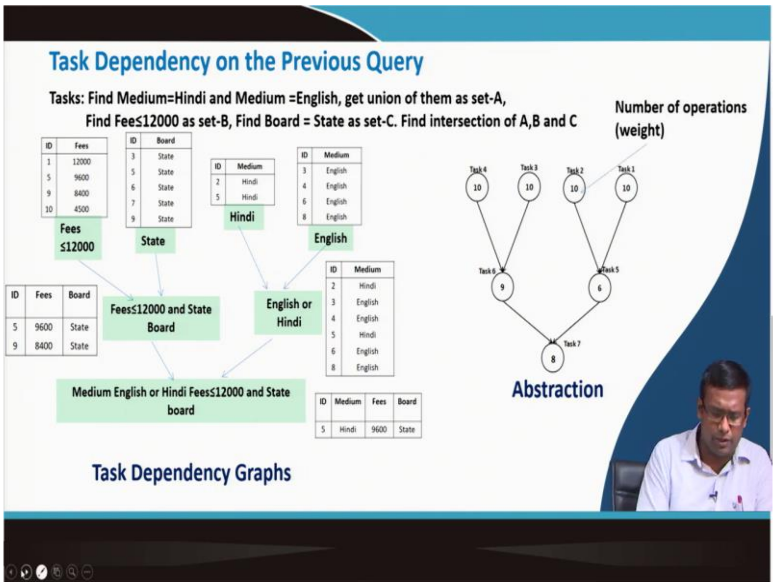
I will go to an more elaborate example which is not exactly a scientific computing exercise rather this is more query processing exercise, but this will give us a good idea about task dependency.

For example, I have for some policymaking somebody gets a list of 10 schools in his locality and this schools in his district. This schools are of rural region from rural area, from urban area, from semi area urban area. And among these 10 schools, there are schools where main medium of instruction is odia ,similarly, there is another school whose medium of instruction is Urdu and other school sub medium of instruction English, Hindi or Bengali. They they belong to different board and they have different annual fee.

Now, one of the query for certain policy purpose, certain award etcetera, somebody wants to find out that which are the schools which either English or Hindi are their medium of instruction or first languages in the schools they belong to the state board in the state they are operating and their annual fee is less than is not more than 12000.

They are not charging more than 12000 and that schools have to be found out from the list or one has to process the query annual fee less than 4000, board is state and medium of instruction Hindi or medium of instruction English and how to find out this.

(Refer Slide Time: 29:49)



Task Dependency on the Previous Query

Tasks: Find Medium=Hindi and Medium =English, get union of them as set-A, Find Fee≤12000 as set-B, Find Board = State as set-C. Find intersection of A,B and C

ID	Fees
1	12000
5	9600
9	8400
10	4500

ID	Board
3	State
5	State
6	State
7	State
9	State

ID	Medium
2	Hindi
5	Hindi

ID	Medium
3	English
4	English
6	English
8	English

ID	Fees	Board
5	9600	State
9	8400	State

ID	Medium
2	Hindi
3	English
4	English
5	Hindi
6	English
8	English

ID	Medium	Fees	Board
5	Hindi	9600	State

So, now if we look into this particular query processing exercise, we will see that there are number of tasks. What are these tasks? we will look into the task graph to understand that. First task is find medium of English, Hindi,because the school has to be either of them, then find a union of this set. So, that these are the schools either medium of instruction English or medium of instruction Hindi, call that as set A.Then find out the fee less than 12 equal to 12000 call set B, find board is state board as set C and then, find what is the interaction of A, B and C. So, there are of course, there are 4 tasks we can find out that taking the same data, there can be 4 different operations and these operations are independent that find out whether the medium of in the instruction is Hindi or medium of instruction is English, find out what is the fee, find out what is the board. This can be done in 4 task can be done independently.

So, these are 4 independent tasks out of here. Then finding out union, finding out intersection their dependent tasks. So, first we found out 4 schools where fee is less than 12000, 5 schools which belong to state board, 2 schools where Hindi is the medium of instruction, 4 schools were English is the medium of instruction.

Now, we have to find out the union of this sets; medium of instruction English or medium of instruction Hindi. So, our first we had 10 schools among which we found out these many classes. So, this exercise among a list of 10, I have to find out where fee is less than equal to 12000 so, we have to go for do a 10 floating arithmetic logical operation and find out this.

The next one is again among 10 schools we have to find out the state board again 10 operations here. So, all these are different tasks which are doing 10 operations and perform are independent.

Now finding out the union of this set, medium Hindi and medium English ,this will take care of 2 schools here and 4 schools here and then find their union.And of course, they are exclusive set because medium is either Hindi or English so, there are total 6 entry and it is doing arithmetic logical operation over this 6 entries and summing up that the lists.

Here, fee less than equal to 12000 and board is equal to state, 4 schools are there, 5 schools are there and we will try to find out what is the intersection of this set. So, we have 4 entries here, we have 5 entries here among this 4 plus 5 9 entries which are common we will identify. So, total 9 operations will be there.And we found out that there are 2 schools which belongs to state board has fee less than 12000.

Here, we have found out there are 6 schools where medium is Hindi or English and then, you can see that this job that finding out these 4 sets can be done concurrently t, finding out this intersection and this union that is also independent and they can be processed concurrently.

And then medium of English, Hindi and fee less than 12000 and state board there that can that has to be found out so, you take this 6 schools here and this by instead of this 6 schools here and set of the 2 schools here over the set of 8 data you found out which are common and finally, you come up with the solution and that there is 1 school which is

ID 5 it is medium is Hindi, fees 9600 and board is state. So, it satisfy the criteria we have given.

Now, there are number of operations. We can see that 10 operations are here for finding out these boards because there is total 10 schools, each operation is finding out one criteria of them among them. So, there are total 40 operations, but we can process them in 4 different computers simultaneously.

Similarly, there are number of operations, 9 operations here and 8 operations here we can process, but instead if we ask one computer, it has to process 17 operation instructions, if we ask two different computers, each one is doing 8 and 9 so, we can save time and but these are dependent. We cannot process this instruction unless these sets are available to us. So, this what we get is called a task interaction, task dependency graph.

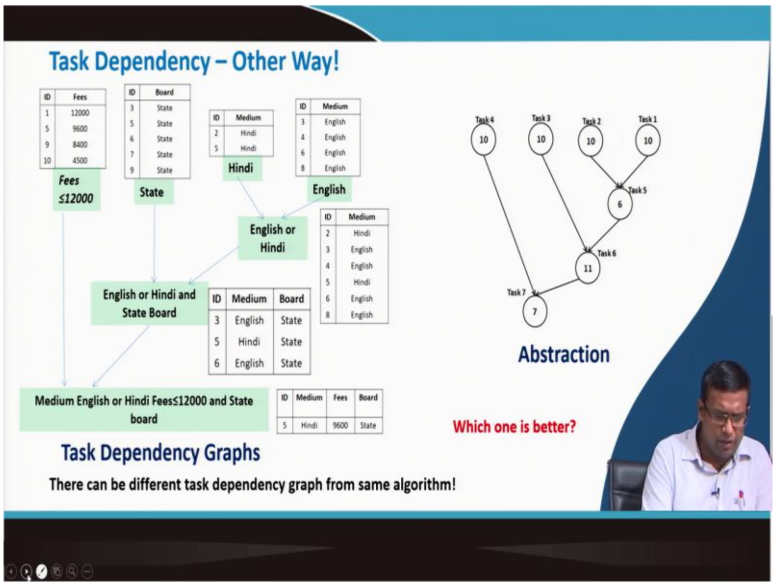
The task 1 is finding out medium of English instruction English, among 10 schools it finds 4 schools. So, if you wrote a computer program, it has looped over a set of 10. Similarly, it has 2 finds out the schools with Hindi, it has also looked over set 10, task 3 and 4 similarly looped over state 10. So, there weight age is 10 the number of operations each task is doing is it so it.

Then, once these task are done, then you can do the subsequent tasks. When that these 2 tasks are done that finding out union of English and Hindi and finding out intersection of fee less than equal to 12000 and state board then, you can get this two sets for from which you can find out the union.

So, these two tasks also can be done independently, but provided the previous tasks are done and the final task which is finding out the school which satisfies all the criteria can be done once all the previous task are done.

So, this graph is called a task dependency graph. You can identify now that for a large problem you are trying to parallelize it, which processes can run independently and therefore, they can go to different processor and which processor has to or which task as to weight for the result of which of this task so, that you can map that as correctly. Now, this is called the weight which is the number of operation for of each task well.

(Refer Slide Time: 36:03)



Task Dependency – Other Way!

ID	Fees
1	12000
5	9600
9	8400
10	4500

ID	Board
3	State
5	State
6	State
7	State
9	State

ID	Medium
2	Hindi
5	Hindi

ID	Medium
3	English
4	English
6	English
8	English

ID	Medium
2	Hindi
3	English
4	English
5	Hindi
6	English
8	English

ID	Medium	Board
3	English	State
5	Hindi	State
6	English	State

ID	Medium	Fees	Board
5	Hindi	9600	State

Which one is better?

Task Dependency Graphs

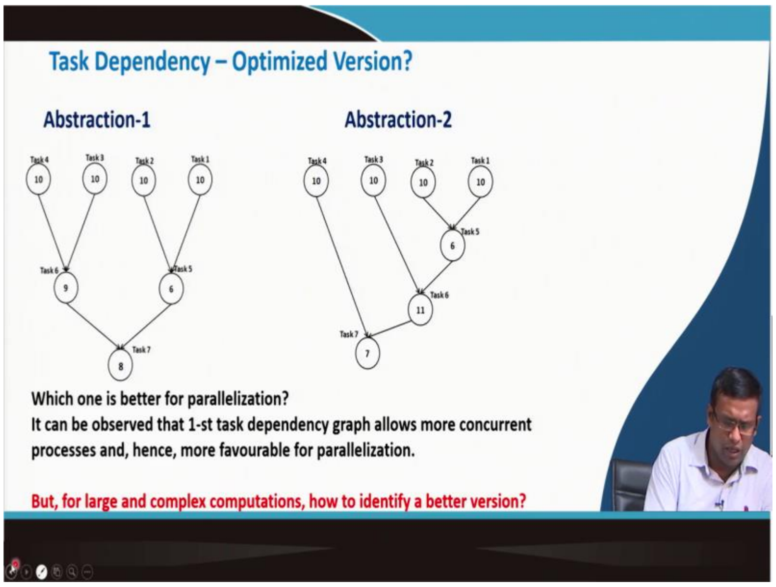
There can be different task dependency graph from same algorithm!

Now, you can get as the task dependency thing done in the other way. First you find out this sets that fee is less than 12000, board state Hindi and English, then you first find out what is the union of this set. And then once you get union of this set or set A, then you find out what is this union with set B, that is the board is the state board and find out the intersection set C or state board and find out the set which is both with both satisfies either English or Hindi medium as well as state board and you got 3 schools here.

Then, you take the list of the school whose fee is less than equal to 12000 and get an intersection of these two sets and get that done. So, you got another task dependency graph out of here. Now the question will be which one is better and you can solve parallel problem different way.

You if identified the tasks involved in the problem, you can get the task dependency graph differently and you can identify the independent parts of the problem in different manner and assign it to different processors, but which one is better that will be the question.

(Refer Slide Time: 37:29)



Task Dependency – Optimized Version?

Which one is better for parallelization?
It can be observed that 1-st task dependency graph allows more concurrent processes and, hence, more favourable for parallelization.

But, for large and complex computations, how to identify a better version?

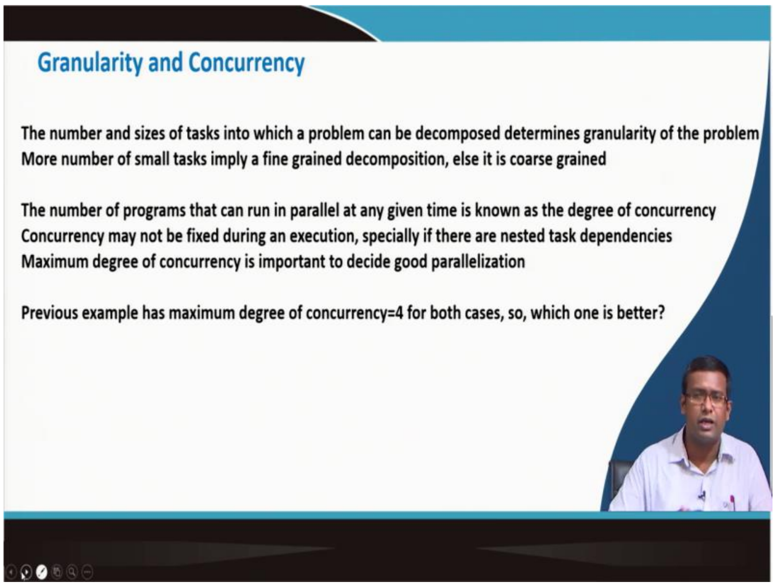
There can be different as dependency graphs from the same algorithm which one is better.We have two task dependency graph. Just visually looking into it, I can tell that the left one is better why because first there are 4 independent processors here, there are 4 independent processors here.

There are 2 independent processors so, this to be these 2 can be performed in parallel in different processors, but after that there is no independent number of independent processor at one stage each process is dependent on the previous process. So, here we can give it to 4 processors, but after that we cannot parallelize it.

So, which one is better for parallelization? It can be observed that first task dependency graph allows more concurrent processes and hence more favorable for parallelization. But for large and complex computation just by looking into these figures even getting this figures will be difficult, but just by looking into this figures I cannot make a decision which one is better version.

Some point I will see one graph is showing more concurrency, in some case one graph is showing less concurrency. So, we need to have some matrix to find out which is more favorable for parallelization.

(Refer Slide Time: 38:46)



Granularity and Concurrency

The number and sizes of tasks into which a problem can be decomposed determines granularity of the problem
More number of small tasks imply a fine grained decomposition, else it is coarse grained

The number of programs that can run in parallel at any given time is known as the degree of concurrency
Concurrency may not be fixed during an execution, specially if there are nested task dependencies
Maximum degree of concurrency is important to decide good parallelization

Previous example has maximum degree of concurrency=4 for both cases, so, which one is better?

And we need to look into some of the definitions. One called granularity; the number and size of tasks into which a problem can be decomposed determines granularity of the problem. So, the number of small tasks you can find out from a large problem gives a granularity.

If you can get large number of small tasks for a large problem say we are talking about a matrix addition, for each row column id of the matrix you can identify one task and so, one element of this matrix added with the another element of that other matrix will be the resultant element of the solution matrix.

So, if there are m x n size matrix, there are m x n number of independent tasks. So, the granularity is high, but if you think of matrix vector multiplication, each row has to be multiplied with a column. So, in the number of independent tasks are less because the matrix vector product in the product each element can be found out independently so, number of the column elements in the column vector in the product matrix gives you the number of independent task or gives you the granularity.

So, when one particular case, you can get more granularity, where you can get more granularity of course, you understand that parallelizing it will be easier. So, one idea is granularity.

More number of fine tasks imply a fine grained composition else it is a coarse grained composition. In some of the cases, say you are we are trying to find out something over a query and you cannot decompose it into many small number of works ,it has to be a large piece of work, it is a coarse grained one. So, if it is fine grained, if you can find out many small units of task it will be better for parallelization.

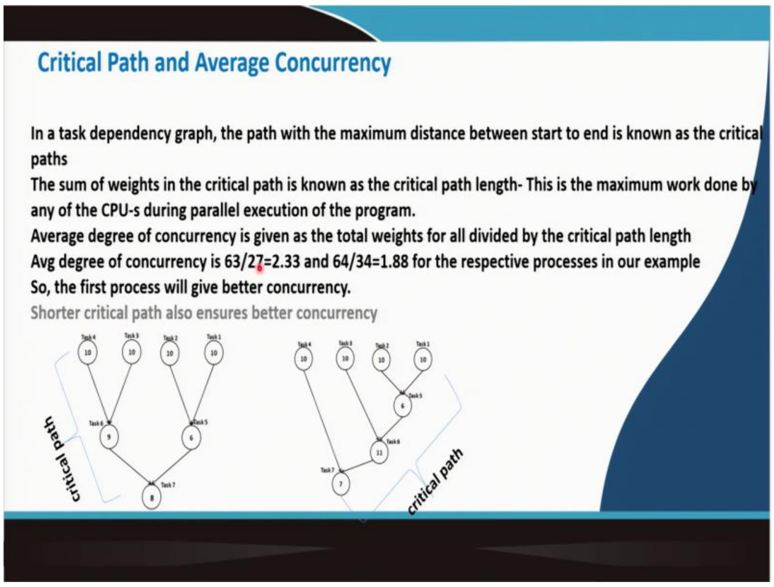
The number of programs that can run in parallel at any given time is known as degree of concurrency. So, even if you have good granularity there can be dependence across the tasks, but if you can find out that this many tasks can be operated in parallel at one particular instant, it is the degree of concurrency.

And as more is the degree of concurrency you can use as many processors to do the work and you can get better parallelization and you can tell that concurrency of course, concurrency is not fixed during the execution we have seen that in the previous example, at the first stage for the first part there are 4 concurrent tasks, in the later part there are 2 concurrent task. So, number of concurrency is not fixed especially there are nested task dependencies.

Maximum degree of concurrency is important to decide good parallelization that at any point of time what is the maximum number of tasks across different stages of the program, what is the maximum number of tasks which are being executed in parallel; that means, that is the maximum number of processors I can devote for solving the problem.

And in the previous example, if we go back , the degree of concurrency is 4 here, degree of concurrency is 2 here, 1 here. Degree of concurrency is 4 here and the rest the one. Therefore, maximum degree of concurrency is 4 for both the cases, but in case they are different it can be a good important parameter to determine good parallelization.

(Refer Slide Time: 42:45)



Critical Path and Average Concurrency

In a task dependency graph, the path with the maximum distance between start to end is known as the critical paths

The sum of weights in the critical path is known as the critical path length- This is the maximum work done by any of the CPU-s during parallel execution of the program.

Average degree of concurrency is given as the total weights for all divided by the critical path length

Avg degree of concurrency is 63/27=2.33 and 64/34=1.88 for the respective processes in our example

So, the first process will give better concurrency.

Shorter critical path also ensures better concurrency

Previous example has maximum degree of concurrency 4 for both the cases again the question is which one is better. We visually we can tell the left one is better, but for a large problem we need to find out a measure like one measure was maximum degree of concurrency some other measure to see which one is better and that is obtained by critical path and average concurrency. In a task dependency graph, the path with the maximum distance between start to end is known as critical path.

So, if we say what is the first task and what is the last task and the maximum distance; that means, as many tasks are possible in between that from first to last, what is the total mass number of tasks covered and total weight age among them.

This distance so, this is from first task to last task, this is the maximum distance possible here, this is called the critical path .Here all the critical paths are the same but, if we can see that distance is measured by sum of the weights so, the weight is more here therefore, this path is given as the critical path.

From first to last, if we first task to last task if we take any of the path and sum of weights of the tasks, the maximum summation of weight will give us the maximum distance and that is called the critical path. Here, this is the critical path.

The sum of weights in the critical path is known as critical path length and this is the maximum work done by any CPU during the parallel execution. So, if we devote multiple CPU's for doing this work, first 4 CPU's are active, here you only need 2 CPU so, 2 CPU's became inactive, here you only need 1 CPU so, 1 CPU is only active.

So, the CPU which was active at the beginning, at the intermediate and at the last stage has performed 10 operations, here 9 operations here and 8 operations here and this is known as the critical path length. So, we will find out what is the critical path and total number of operations the CPU is performing through the critical path.

Average degree of concurrency is given as the total weights of the entire process divide by the length of critical path. So, we can say that for the first the total weight is 10+ 10+ 10+ 10+ 15+ 8=63. Here total weight is 40 +24=64 and critical path length here is 27, here critical path length is 16+ 18= 34. So, average degree of concurrency is 2.33 here, 1.88 here.

The first process which has higher average degree of concurrency will give us better concurrent concurrency and therefore, this will be more favorable to parallelization. That is the decision out of task dependency graph that you find out the critical path, draw that as dependency graph, find out the critical path, then find out what is the average degree of concurrency.

The critical path length you use it to divide the total weights of the problem and these this division will give you the average degree of concurrency which one will give you more average degree of concurrency that will be betters algorithm to parallelized.

Also shorter critical path ensures better concurrency because shorter critical path means the denominator will be smaller and you will get higher average degree of concurrency. So, this is one example where multiple avenues of parallelizing a particular problem is present and we can look into the task dependency graph and decide which one will be the best way to parallelize it.

We will look into some of the other examples also.We have only seen that what is the feasible way to do a task, what are these dependency it is feasible only when the dependencies are solved. But we have not looked into the point that there is also communication and synchronization steps present there ,that the results of one of the tasks have to be communicated to the subsequent tasks and this is also another important task parameter here.

In the subsequent lecture, we look into the task interaction graph that ,the data which one task is using will be required by some of the other tasks and that is called task interaction and what is the effect of task interaction in communication and synchronization of parallel algorithms and how it has to be considered while parallelizing a particular problem.

So, what is its importance in considering the right parallel algorithm.We will look into task interaction and then we will start discussing about decomposing a problem, that this is one for one particular case we have done tailor made decomposition.

But there will be cases in for which you need to do decomposition in a more general way because the there will be algorithms which fall in certain category and some techniques for decomposition is already very well established for this, we look into this decomposition techniques in the subsequent lectures.

Thank you.