# 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 - 10**
**Performance Metrics of Parallel Systems**

Welcome to the class of High-Performance Computing for Scientists and Engineers and we are discussing about Fundamentals of Parallel Computing which is the 1st module of this class. Till now we have discussed about architectures of parallel systems like shared memory and distributed memory platform. We discussed about different models of parallel programming and how to design a parallel programming.

We have seen that in a parallel computing infrastructure there are some limitations. Limitations in the sense we really cannot get linear scalability. We will discuss about scalability in the next lecture, in that sense that if we have a problem which is taking 5 hours in a sequential program or a single processor if we use 10 processors it will not take 0.5 hours or 30 minutes, it will take more time than that.

So, a parallel system is not an ideal system. If we increase more processors, we will see some enhancement in performance; however, the performance enhancement is not exactly as with the as the time divided by number of processors; that means, there is some overheads and we have discussed in detail about the overheads in shared memory and in distributed memory machines.

One part of the overhead is due to communication the time required by communication across the processors. There is also overhead due to synchronization. We see the shared memory machine communication overhead is mainly due to the sharing of similar data in different processor same data in different processors where their caches have to be coherently updated and therefore, some amount of time will be gone for this cache coherency protocol.

There will be false sharing which again has is due to this cache coherency protocol it. There will be race condition that same data is trying to be operated by different processors at the same point of time like that. So, this is a shared memory machine. In a distributed memory machine, data has to go from one processors memory to another processor’s memory using a network switch type of thing and that will require certain amount of time.

And then when there will be synchronization across different processors that at some point of time all the processors have to wait till one has finished their work. We will see there are some synchronization like critical or atomic in case of shared memory machine where one part of job though it is inside a parallel loop is operated sequentially by different processors.

And in any parallel program there are some operations which cannot be done in parallel which is inherently serial or sequential like it has to read from a file. Even if all the processors are reading from the file it is the same job everybody is doing.

So, there is no parallelism in that. So, therefore, these things will add into the overhead, especially communication synchronization over head of the parallel program. And this overhead will depend on the algorithm, on the problem that we are trying to solve as well as on the hardware and compiler which the parallel program and has least control, it is what is given to him.

And there is another overhead which is called a load balancing overhead that is latency that is that not equal amount of task is given to all the processors. So, some processors have finished their task earlier, some are still waiting and therefore, the waiting processors have gone into an idle stage.

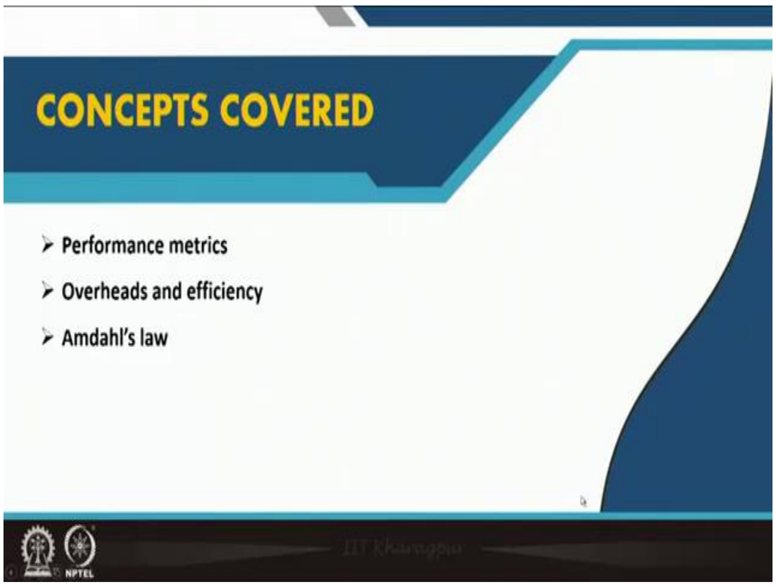
Though they are in operation though electricity is passing through these processors and they are very much counted to the operational. But we are not getting any benefit of their computation at that particular time when they are idle. And naturally in any computer’s CPU cycle has some idle time also.

So, this some idleness is due to load balancing, some idleness is due to the hardware as well as the operating system, the processors scheduler itself. So, these overheads are there and due to these overheads even if we are increasing number of processors, we may not get the best desired output.

So, it is important to know what is the best output possible out of an infrastructure, can we increase that output in some way, if you are not getting that output what are we doing wrong. If we are trying to do something, we are increasing number of processors, but we see that we will not get any benefit in terms of computational time or in terms of computational cost what can be these situations.

So, it becomes important to look into performance matrix of parallel systems. So, which is this particular lecture is about that what are the performance matrix of parallel systems.

(Refer Slide Time: 05:36)



## CONCEPTS COVERED

- Performance metrics
- Overheads and efficiency
- Amdahl's law

IIT Kharagpur

We will discuss about performance matrix; that means, what are the terms by which we can quantify performance of a parallel system. And this performance includes the overall performance of the system. It includes the performance of the hardware, performance of the computer and other software support as well as the performance of the algorithm and the program.0

Some part of this we can control as a programmer as a person working on the algorithm, we can improve the performance, but beyond the limit we cannot improve the performance because it is limit. It should be limited by the hardware and the other back end software.

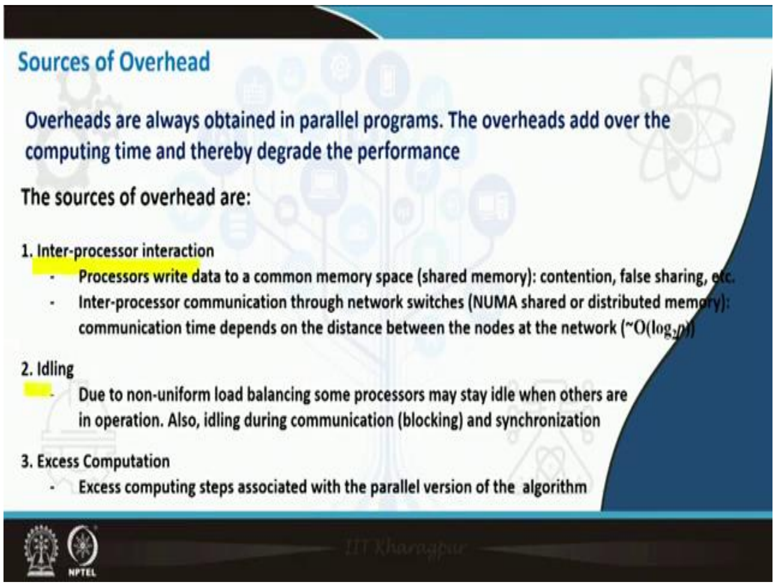
What are the overheads and what is efficiency of a parallel computing systems? If we are using problem in 2 different architecture; one is having 10 processors and other is having 80 processors. How we can see that which one in which case if we are using it more efficiently and what is the issues.

Because you see when we are using a greater number of processors, we have to spend more money in terms of electricity maintenance. We have discussed earlier as the processors are working their CPUs (Refer time: 06:53) space are being heated on and they require electricity further to dissipate this heat we require certain amount of cooling which also require power of electricity.

So, even if we are increasing processor, we are getting some betterment in time their computational time is reducing, but is it being done in an efficient way. And then we will see what is the maximum limit of performance of a parallel program or a parallel computing infrastructure as well as a parallel program.

And that is called Amdahl's law which gives a cut off that this is the maximum efficiency or this is the maximum best performance you can get out of a parallel program. Depending on the program the algorithm itself as well as depends on the programmer.

(Refer Slide Time: 07:48)



## Sources of Overhead

Overheads are always obtained in parallel programs. The overheads add over the computing time and thereby degrade the performance

The sources of overhead are:

1. Inter-processor interaction
   - Processors write data to a common memory space (shared memory): contention, false sharing, etc.
   - Inter-processor communication through network switches (NUMA shared or distributed memory): communication time depends on the distance between the nodes at the network (~$O(\log_2 p)$)
2. Idling
   - Due to non-uniform load balancing some processors may stay idle when others are in operation. Also, idling during communication (blocking) and synchronization
3. Excess Computation
   - Excess computing steps associated with the parallel version of the algorithm

So, first important thing is overhead in a parallel program. We have discussed about overhead in several times through in this discussion. So, overheads are always obtained in parallel programs. You already know that in any parallel program there will be overhead.

These overheads add over the computing time and thereby degrade the performance. So, what is the time in which the processors are crunching numbers are doing arithmetic and logical operations. Apart from that due to this overhead there is some time in which is included in the total time required by the job parallel job and therefore, the total time will increase because the over head time is included into that.

So, the sources of overhead are inter processor interaction. Here processors write data to a common memory space in case of shared memory there could be contention, false sharing, establishing cache coherency etcetera which we will which will require some amount of time and add to some overhead.

Inter processor communication through network switch in case of a non-uniform memory access shared memory, NUMA shared memory machine or distributed memory machines, communication time depends on the distance between the nodes at the network and if there are p processors the communication time in a network the communication time will be of the order of log p with the base of 2.

Then this can be shown to very easy calculations and I am not doing it here. And so, but we will take the fact that if there are p processors the communication time will be log p with a base 2.

Idling due to non-uniform load balancing some processor may stay idle when others are in operation. Also idling during communication blocking communication; that means, one processor is communicating to other while communicating it is blocked it cannot do anything else. So, and when the other processor is receiving when it knows that the other one is giving some data to it while receiving it cannot do anything. At this time when the communication is being done the processor is go into an idle stage, they do not compute anything though they are in operation.

And there are some plus minus times during communication in which they also go into idle stage. As well as when there is a synchronization that all the processors have to wait till all the other processors comes up to one part of the job. So, when this synchronization is given or for shared memory programs some part of the memory is accessed sequentially by all the processors. We will see atomic or critical comments when we will do open MP.

So, these are called synchronization steps in which processors wait for the other processors to finish up to some work and then they go into idle stage. So, although they are adding to total computing time, but they are not doing anything. Basically, they are in idle so they are adding to overhead. So, these are the main 2 sources of overhead, inter processor communication and idle time and we have a third overhead is excess computation.

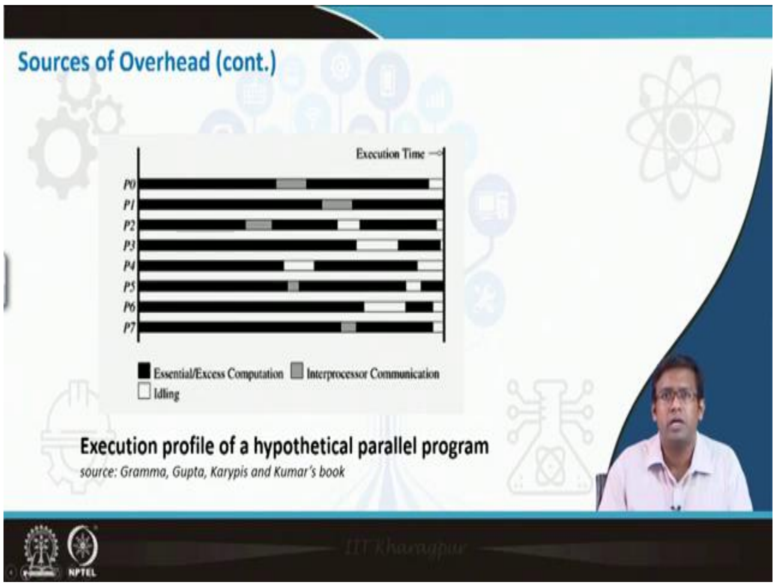
So, excess computing steps are associated with parallel version of the algorithm. If we take the sequential program there is some computing steps. But when we think about a parallel computing, we can understand that we have to find out that how many tasks are there, do some task interaction type does dependency type of graphs, which task will go to which processor, this mapping has to be done.

Once the processors have done their work, we have to take their output and combine them or map write them into a file or do an operation over different outputs coming from different processors probably.

So, some computing will be added over the serial program when we think about a parallel program. So, this excess computing will be added and this is the third source of overhead will be added to the sequential program.

So, this 3 was not present when you are thinking about a serial program, but when you have thought about a parallel program this 3-extra time, one is due to inter processor interaction another is due to idling due to communication or synchronization or non-uniform load balancing and excess computation are added to it. And therefore, they will degrade the performance of the system and now we will see that how do they behave in certain cases.

(Refer Slide Time: 12:35)


## Sources of Overhead (cont.)

Execution profile of a hypothetical parallel program

*source: Gramma, Gupta, Karypis and Kumar's book*

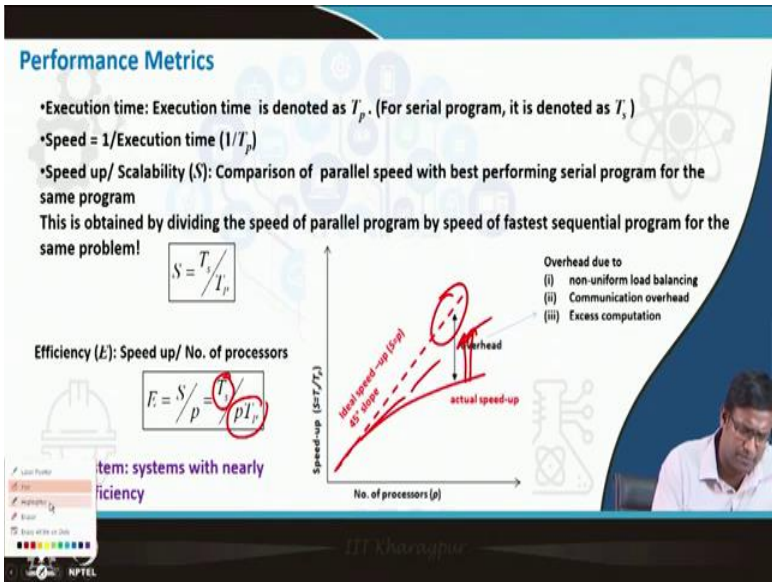
So, this is a figure taken from Gramma, Gupta, Karypis, Kumar book that execution profile of a hypothetical parallel program. Say a parallel program is running over 8 processors and when they are running, we try to find out that over a certain amount of time which processor is doing what one what thing.

The black one is the computation and this can be the essential communication which was essentially presenting even in the serial algorithm or the excess computation added due to our parallelization of the job. The gray one is the inter processor communication and the white one is the idle time.

We can see that in any of the processor it is not only computation. Computation is some part of time, but at least 10 to 20 percent time is gone for idleness as well as. So, these are the ideal times in a one processor as well as inter processor communication. Therefore, if we add up the total time of used by all the processor, a substantial amount of time which is in between 10 to 20 percent of the total time is gone for something which is due to inter processor communication and idling.

And we can understand that as we are increasing number of processors, we should get faster solution because many computers are taking care of the large problem and it will be solved faster, but the inter processor communication is also increased and probably idling has also been increased due to non uniform load balancing as we are increasing more processors the load balancing becomes more difficult and therefore, these overheads have also increased well.

(Refer Slide Time: 14:16)



## Performance Metrics

- Execution time: Execution time is denoted as $T_p$. (For serial program, it is denoted as $T_s$)
- Speed = 1/Execution time ($1/T_p$)
- Speed up/ Scalability ($S$): Comparison of parallel speed with best performing serial program for the same program

This is obtained by dividing the speed of parallel program by speed of fastest sequential program for the same problem!

$$\boxed{S = T_s/T_p}$$

Efficiency ($E$): Speed up/ No. of processors

$$\boxed{E = S/p = T_s/pT_p}$$

tem: systems with nearly

ficiency

So, let us look in detail. Now, we know that overheads are there it will always not give us the right with the most desired result. What should be our desired result? If we take 2 different parallel systems or if I take a single problem as 2 different programmers to make 2 parallel versions, and there are overheads. The performance of the versions will be probably different if they parallelize it differently. Which one is best?

So, we need some quantitative parameter to identify the performance of a parallel process or parallel program. As well as how one particular parallel program behaves in different infrastructure. We have some homegrown cluster here and we have a multi core machine in which the program works better. So, to understand that we need to find out some parameter which is a which can be quantifiable. Some quantitative estimate is required and this is obtained through performance matrix.

The first one we understand is execution time which is denoted as $T_{p,}$ that how much time this is the parallel program is taking. If you are running a program in 10 processors in one particular infrastructure, what is the time taken? And this is to be compared with the serial programs time that what is $T_{s,}$ that if we do not use a parallel program what for the serial program for the same problem what should be the time taken.

However, and execution time and the speed of the parallel program because when we think about parallelism better performance, we think that the computation will be done for in a faster time. So, something like speed is important, speed is inverse of execution time. In some cases, we can also say that how many iterations or how much computation has been done that is also a measurement of speed.

But usually if the we have to find out the total execution time of the parallel program and inverse of that gives us the speed. And speed up or scalability which is the most important one of the most important parameters in parallel program is the comparison of parallel speed with the speed of the best performing serial program. What is the best performing serial program that do not take that take the parallel program and run it in single processor?

Think that you do not have to parallelize it. Take the main problem and what is the best possible program can that can be written to solve that problem that program that algorithm may not be the good one in terms of parallelization. But what is that? So, take how much time it is taking.

So, consider you have given a problem what is the least time in which this problem can be solved in a single sequential program in a single processor. An inverse of that will give you the sequential speed. If you compare parallel speed with the sequential speed you will get speed up or scalability. This is obtained by dividing speed of parallel program by fastest sequential program for the same problem. It may not be the same program running in the same single processor.

Because when you make a parallel program there are many excess computing computational steps also, but in the simple sequential program these are not there. Also, a problem can be solved in many ways, but you think of that one which is which gives you the fastest solution in a single processor and that speed is that speed to divide the speed of the parallel program. This ratio is called the speed up that is what is your benefit when you are using a parallel infrastructure how by which ratio you have speeded up the program.

So, this speed up is denoted as S which is $T_{s/}\, T_p$. The time taken by the best performance sequential program divided by the time taken by the parallel program. T s by T p this is given as speed up. And we can understand this is always greater than 1.

If it is less than 1, I mean desired is always greater than 1. If it is less than 1 then or 1, then there is no meaning of using parallel infrastructure. We have met the parallel program is taking more time than sequential program and nobody will allow you to learn the program in parallel.

And if we draw speed up a curve for any general parallel program you consider and you see how the speed up is increasing with number of processors. In the ideal, case that you are adding one processor more and you are getting the same is same similar ratio of speed up. It should follow a 45-degree line. In a real case, this deviates from that 45-degree slope line due to the overheads.

As we increase number of processors speed up is increasing, we are getting faster solution; however, it is not as fast as the number of processors are being added. Because with increasing in processors the overheads are increasing and we understand that over head depends on inter processor communication or we have earlier seen inter processor communication time is of the order of log p to the with a base 2.

So, as we are increasing number of processors, inter processor communication time or overhead is increasing as well as synchronization overhead and load imbalance over head all these things will add. So, the actual speed of curve will deviate from the 45-degree slope line and eventually it might come to a maximum and then it starts showing some reductions showing me.

It should it will start to show the reduction and it is it will reduce the value of speed up will reduce with increasing number of processors if we use a very large number of processors. At that point of time it is not meaningful to use a large number of processors.

So, when it is reaching a maximum will probably stop. We cannot add any more processors then that is not beneficial. And it is also our desired that this curve should be much closed. So, it is also one of the desires that this curve should be much close to the ideal curve that the overhead will be less and we will get better speed.

What does it mean that if we consider another curve like this? It means that with as we are increasing the number of processors, the speed is increasing more than the curve shown here. So, that is a much more acceptable parallel program. Now, what is the best possible parallel problem? 45 degrees is the linear scalability. There are some cases when you get super linearity.

So, curve actually goes like this, but we are not discussing it here. It has more complex issues involving hardware and memory access. But our desire will be the curve should be pushed more towards this line that the over head will be reduced.

So, what is the maximum limit to which we can push that that is also we need to know. Because we if we have already reached the maximum limit, we should not spend more time in to increase the performance. It is like trying to violate laws of thermodynamics. If you have already reached Carnot efficiency you should never think of increasing the efficiency.

Similarly, so, one term called efficiency comes here and what is efficiency? Thus, the overhead is due to I have discussed it non-uniform load balancing computer communication and synchronization overhead and excess computation. So, efficiency speed up divided by number of processors. So, ideally efficiency the efficiency of the 45-degree slope curve is 1. I have discussed about super linear speed up.

Let us forget that at this stage. So, the ideal efficiency is 1, but can we reach efficiency 1 we have to see and this is therefore, given as efficiency is

$$E = \frac{S}{p} = \frac{Ts}{pTp}$$

. The serial programs time divided by parallels program time into the number of processors. And here comes another thing.

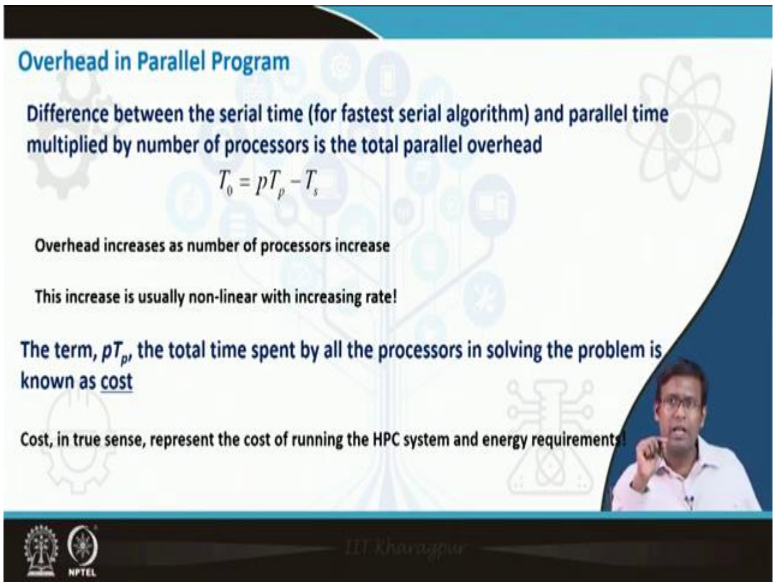
We know that efficiency, as I discussed comes from a thermodynamic discussion that, how much benefit we are getting compared to how much cost we are spending.

So, this is the T s is the serial program times single processor program time. This is the job is getting done and this is what we are spending in terms of our parallel infrastructure. What is this p T p? We will I will come to p T p, but scalable system is called a system with merely constant efficiency. We will we will discuss about p T p later this is also important.

Merely constant efficiency means if we if the efficiency remains constant if we increase the processor by 2, T p will be reduced by half. So, as we as much as we will by a factor of 2, T p will be reduced by half, as much as we will increase the processor, we will see the same effect in the speed. The speed will be increased similarly and that is called a constant efficiency system or a scalable system.

This it is very desired to get a scalable system. If you have a scalable system, we just increase number of processors in the same ratio you will get your speed up of the of computing. So, you can predict what will be the computational time just by increasing number of processors. There has to be something with the overhead also we will and we need to see that in that case.

(Refer Slide Time: 24:42)



## Overhead in Parallel Program

Difference between the serial time (for fastest serial algorithm) and parallel time multiplied by number of processors is the total parallel overhead

$$T_0 = pT_p - T_s$$

Overhead increases as number of processors increase

This increase is usually non-linear with increasing rate!

The term, $pT_p$, the total time spent by all the processors in solving the problem is known as cost

Cost, in true sense, represent the cost of running the HPC system and energy requirements!

And what is overhead difference between serial time (for fastest serial algorithm) and parallel time multiple by the pair number of processors is called the total parallel overhead and we can understand that $Tp$ is the time taken by the parallel program and there are p nodes there.

So, p $Tp$ is the total time spent by all the parallel processors for that program and what is its difference from $Ts$? It is of course, greater than $Ts$ because of computer communication and other overheads are including there and this difference is simply called overhead $pTp - Ts$ .

Overhead increases as number of processors increase. We can clearly see it from here because efficiency is not 1 therefore, $pTp$ is always greater than $Ts$ and as we are increasing p, the over heads will increase and that is the behavior of the system also. The increase is usually non-linear with increasing rate. As we are increasing the number of processors the overhead is increasing and this increase of over head is increasing with increasing number of processors.

So, as we are increasing number of processes, the over head is increasing. Increasing more than in a linear rate, the rate is also increasing it is increasing more. Because communication overhead is for p number of processors is given as $\log_2 p$. So, the rate is also increasing. As we are increasing processor it is not linearly increasing it is super linearly increasing.

The term $pTp$ and I said this is important. This is the total time spent by the processors in solving the problem, $Tp$ is the time spent for solving the problem in a parallel environment p processor are there. Therefore, combining all the processors this is taking $pTp$ time.

And what is happening in this $pTp$ time? And this is known as cost. In this $pTp$ time all my processors are on, they are taking electricity, they are taking AC power etcetera, they are take taking all the operational cost, the capital cost everything is included in true sense this is the cost of running HPC system and energy requirement.

So, while getting parallel performance many measures are important and we will discuss about different other measures. But it is also important that you should be aware of this term $pTp$ which is the cost of running the HPC system for that particular problem. And as we are increasing over it $pTp$ will increase cost will be more. So, we have to reduce over it in some sense to get better $pTp$.

Thank you.