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 - 01
Introduction to High Performance Computing-01

Hello. I welcome you all to the first class of this course High Performance Computing for Scientists and Engineers. And, this is the 1st module which you are discussing fundamentals of parallel computing. And, as I said this is the 1st lecture which is Introduction to High Performance Computing.

While welcoming you all to this class, we need to see what is the focus of this course and who are the target audience or the students and application engineers, who belong to that group for which this particular course is designed. Of course, I will say it is not the exclusive group whom I intend to attend this class, there can be students and professionals from different areas who are also welcome in this class.

So, high performance computing is a very important topic in today's computational science as well as engineering because in the 3rd decade of 21st century, we really try to understand many complex physical phenomena many complex biological systems many complex social behaviors and we talk about data driven sciences.

We talk about very accurate and high fidelity physical simulations which require high computational infrastructure. Because we try to understand the problems with much rigor the computational complexity is increased, and simple computers or a single computer with probably the fastest processor and largest RAM is not sufficient to handle that complexity.

And we will also discuss about supercomputers today and most of the real engineering designs as well as most of the invar scientific computations, as well as many other computer application problems in many different fields like biomedicals, like economics, like sociology, like public health are solved using these supercomputers. And that is why today there is a need that application engineers, scientists and other domain experts should be able to work in the supercomputing platform.

In most of the cases, they will deploy some of the software which are already developed to port in that supercomputing platform. However, they need certain understanding of the platform itself as well as the methodologies that this high-performance computing or supercomputing technique is using.

We can see and I will discuss it later also that our government, the government of India has launched national supercomputing mission few years ago. In a view to promote supercomputing and trained manpower who are who can have jobs in maintaining supercomputers and also in porting different applications into supercomputers.

This is one side that we need application engineers, application scientists, domain experts who are using high performance computing facility the other end is that as these applications are being demanding day by day therefore, we need a pool of computers engineers, computer scientists and computer application specialists who can help the domain scientists to work in the supercomputing platform.

So, there is a need both from the computer scientists computer engineers perspective as well as from the perspective of application scientists and domain experts to step into supercomputing with a view that they understand each other's requirement. That, the computer scientist or computer engineer understands the requirement of a domain expert, as well as the scientist can communicate his requirement the domain scientist or the design engineer can communicate his requirement to the computers engineers community.

Unfortunately, the courses in high performance computing are mostly looking into computer engineering perspective. However, people from science and engineering domain many times fail to jump into these courses simply because of the fact they are not trained for that they do not have the right prerequisites.

And therefore, a gap is created in which computer engineers talk in a different language and while they looking into networking the algorithmic point of view, the hardware point of view of supercomputing the application of supercomputing which is in parallel computation is scarcely looked upon by them which is required by the application engineers.

And as we are looking for going into next decades computing with exaFLOPs computing, we are setting up many petaFLOPs supercomputers all across the country there will be a need to bridge that gap,therefore, this particular course, I am myself is not from a computer science background, however, I have been using supercomputing facilities for nearly 2 decades.

And I am also teaching courses which look upon applying high performance computing a supercomputing techniques with the most cutting edge techniques like GPGPU computing into scientific problems. Hence, the goal which I set in this course is to bridge the gap that is we will discuss about high performance computing technique, we will discuss about the computer science as well as mathematics part involved in the high performance computing applications. However, will keep a focus that our goal is to look into scientific problems or engineering problems.

Well, let me jump into the class. And another important thing is that because we as a country we have a mandate to develop manpower trained in supercomputing,so, in this class probably teachers or as aspirant teachers who want to pursue a career in teaching computational science and scientific computing probably can get some benefits. Well, they can use some of the materials being discussed in this class to train the next batch of students.

(Refer Slide Time: 07:28)

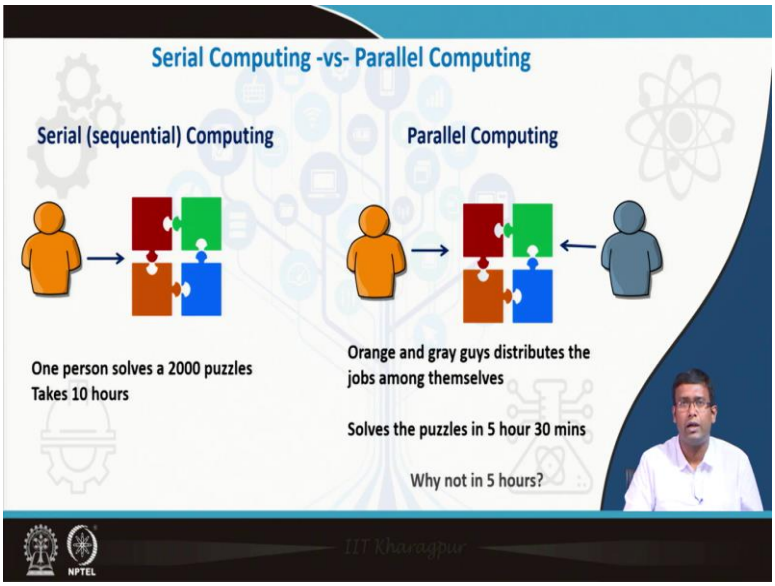
Well, let us start the discussion for today's class. As a first class will first discuss what is serial computing and what is parallel computing; in a supercomputer or in a high of high performance computing architecture we typically take a large job distribute into many small jobs and ask many computers to work in parallel to solve that job. So, what is that? In an in as an abstraction how that can be represented?

Then, from a scientific computing perspective what is the requirement of large scale computing. And I am a computational fluid dynamist by trainings so, I will take a computational fluid dynamics example and see that in certain cases to understand the right fluid mechanics we need supercomputing or large scale computing with a simple sequential computer we cannot even finish the job in any time scale.

Then, we look into some high performance computing architecture. We will see what are the elementary steps in parallel program because parallel programs are the pointer which will refer several times and it is what we are trying to do over this course. We are trying to see how can we write parallel programs and make them efficiently work over different high performance computing infrastructure using several APIs. We will look into it later in this class.

So, the first module of this course will probably discuss mostly on the fundamentals of high performance computing, but in the subsequent modules we will discuss how to write parallel programs in for different platforms. And then what I have been discussing over last few minutes that what is the structure of the course I will try to discuss that in a more structured manner that what are we going to discuss in the course and what will be the books for that.

(Refer Slide Time: 09:15)



So, let us see what is serial computing or our standard sequential standalone computers computing and how does it compare with parallel computing. As we can see from the figure, we talk about solving a puzzle. And this puzzle is a simple puzzle like putting 4 pieces of our Lego kit together and building a block. So, if 2000 pieces are given to a person and he is a tedious monotonous job, but he can finish that in probably 10 hours that taking 4 of the pieces from this 2000 which matches in say in certain sense in color or orientation and putting them together to make a single puzzle.

Now, a person wasting 10 hours for this has certain implications. So, let us ask another person, so this the previous guy was wearing orange shirt. Now, I asked another person who is wearing a grey shirt to join hands with him and we will also take a few of these puzzles and concurrently they will work on the same problem. So, 1000 puzzle, 1000 puzzle each it will go and we will see that they are solving these the all the puzzles are cover combined and this is finished in 5 hours 30 minutes. That means each one is solving 1000 puzzles each.

So, if one can do 2000 in 10 hours, why two are taking more time? Of course, we can estimate the fact that they are wasting some time in chatting among themselves. But let us see that is not the case. We are using robots for doing that who can keep on doing these monotonous job without having human wastage of time human nature of time wastage.

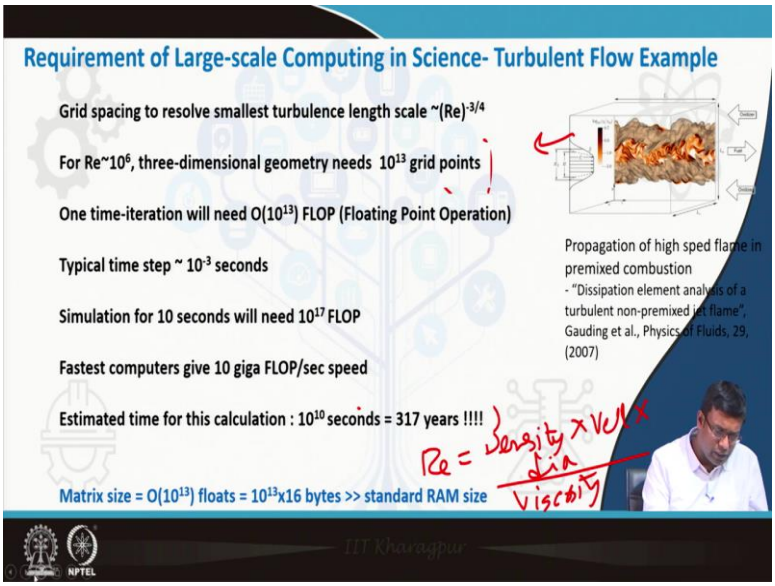
So, still it is 5 hour 30 minutes because some time will be give state or some time one has to be spent while stacking 1000, 1000 puzzles each in from 2000 puzzle. One has to take 1000, another has to take 1000 this will take some time and some time will be gone for again putting all the solved puzzles together or all the Lego blocks together which will adding up it will take close to 30 minutes and there is need of some synchronization in between them.

So, question is why not in 5 hours? There is a question because best we could have done it in 5 hours. So, we have to waste some time while working in parallel in compared to the fact that if we can perfectly divide a job into divide the serial job or sequential job into multiple computer. So, instead of learning one do loop here we are asking two of these to learn two different loops.

However, it is taking some more time and we need to see can we reduce this time and solve it in solve it in close to 5 hours or if not what else can we do. So, that is one of the goal while will look in to look through this class. Especially, in the first half of this course when we will discuss fundamentals of parallel computing we will see what are the overheads when we asks multiple computers to work in parallel like solving the puzzle. Many people are coming and solving the puzzle together distributing the number of pieces among themselves. So, the overall time requirement can be reduced.

However, there is some overhead it is not reduced exactly by the number of people, I will divide by as dividing the total time for a single person by the number of peoples so, there is some overhead, how to do and handle the overhead.

(Refer Slide Time: 12:56)

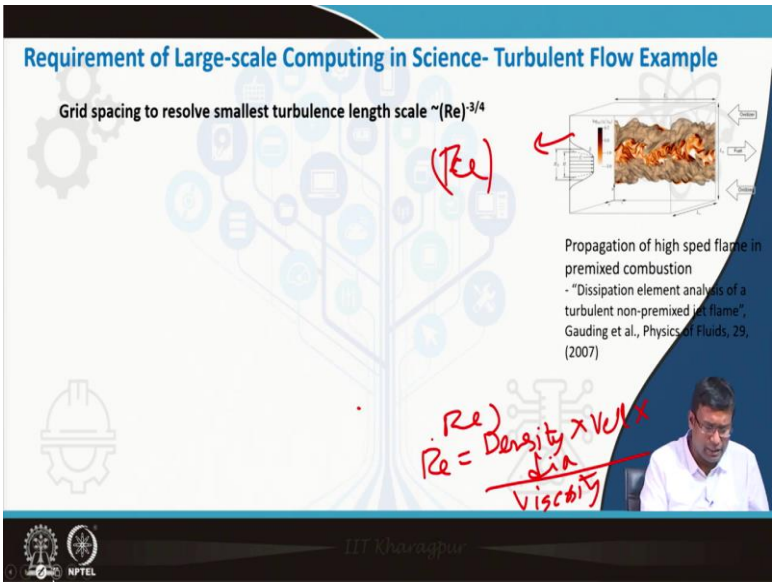


Well, so now, see why is it required. We will understand that it will cut down some time, but the fact is that if you run a multiple computers it require capital investment in terms of purchasing the computers, operational cost in terms of running the computers. So, it is not a person, it is not a man power which we are using, we are using computer powers instead of doing it 5 or 10 hours or 5 hours 30 minutes in two computers who could have done it in single computer. Some time you have to wait. Does it matter? So, let us look into that.

And I told you that I will give an example from my field of work. So, we take a fluid mechanics problem that a high speed flame in pre mixed combustion is a propagating through a chamber through a cubic enclosure. So, this is an enclosure with 1 meter by 1 meter by 1 meter in size say and flame during combustion is propagating through it.

And this is typically a turbulent flow problem. So, for turbulent flow problems this problems are governed by a parameter called Reynolds number which is given as Re sorry, Re and Re is equal to density x velocity x diameter or the length of the or width of the channel divided by viscosity. So, if we see that for Reynolds number the grid spacing to resolve the smallest turbulent scale.

(Refer Slide Time: 15:07)



So,we solve it using a computational fluid dynamics process. So, we can see that this problem is a function of the all the parameters velocity pressure everything is some way a function of Reynolds number.

Now, if we try to use some computational to solve it we can see that the grid spacing the distance between two points while resolving this flow, for turbulent flow to resolve the smallest scale of turbulence will be Reynolds number to the power minus 3 by 4. And that tells us that the for Reynolds number 10 to the power 6 which is the Reynolds number we typically handle for high speed some of the high speed combustion problems.

In a cubic geometry there will be 10 to the power 13 grid points required. And, so, solution over 10 to the power 13 grid point with our best possible algorithm for fluid mechanics problem for one time step ,one set of solution it will take 10 to the power 13 floating point operations. Well, typically a time step is 10 to the power minus 3 seconds of solving for 10 seconds will require 10 to the power 4 time steps or 10 to the power 13 into 4 or 10 to the power 17 floating point operations.

Lets assume I have a computer with 10 gigaflops per second. So, what is the total time requirement to solve this problem? 10 to the power 17 FLOP divided by 10 gigaflops which is 10 to the power 10 seconds or 317 years. So, we will just solving in a one cube geometry of a turbulent flame propagation which is much more simplifier simple case

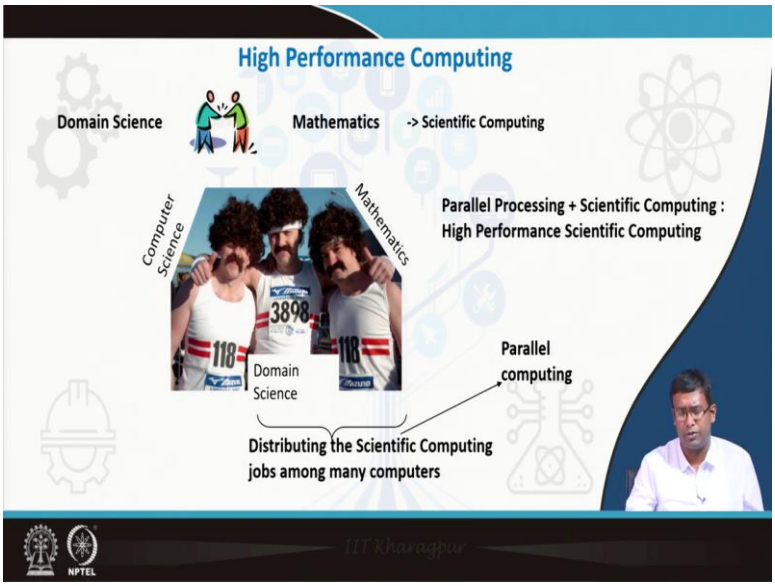
than and actual industrial problem. If we try to resolve the smallest scale of turbulence we take 317 years. So, what do we do?

Typically, we do not resolve the smallest scale of turbulent lengths, turbulence lengths at even in even today's date. But if we try to do something close to date, we have to wait for 317 years I need a computer which will operate over 317 years, then I need a legacy of my students who will be living or whose student's student, somebody will be living after 317 years and they will see that what one of the person 317 years back started learning what is the results of the job and probably it will have no use that time I mean our science will progress in a direction where we might not be interested to know this. And forget that this is not doable; this is not feasible that a problem can be solved in 317 years.

But later still assume that we have a some superhuman set up to wait for 317 years to see what happens in a one cube combustion flame for 10 seconds. Still, we can see that the matrix size which will come out for solving this will have at least 10 to the power 13 rows in the matrix. The minimum number of size of the matrix will be 10 to the power 13 into 16 byte its much more than any of the available RAM size. We cannot load the matrix to computer and solve the problem.

So, we forget waiting for 317 years, it cannot be solved even for a single iteration. Even we cannot load this problem into the RAM of the computer. And therefore, if we try to solve a large scale scientific computing like turbulent flow there can be many other examples, molecular dynamics can be one example, protein simulations can be examples, cloud simulations can be examples, a plethora of black hole simulations can be example. A plethora of examples can be given where a single computer cannot handle this job. So, we have to think of something different and that is where we tell that we need high performance computing.

(Refer Slide Time: 19:37)

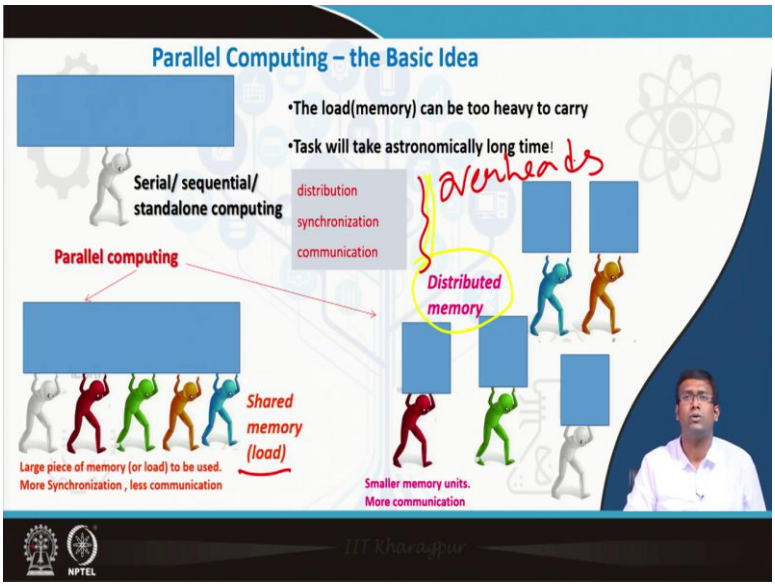


Right now we thought of a computational science which is domain science is shaking hand with mathematics and they are using some computer methods simple things like C or FORTRAN or python program to get their job done and then this is called scientific computing.

Now, we brought the third guy in who is computer science. So, mathematics plus domain science which is scientific computing, now they have to distribute the scientific computing job into among many computers and these computers have to communicate, these computers have to synchronize and solve a problem completely in totality.

So, this is a this is parallel computing where the scientific computing jobs are broken down into many small jobs and given to many of the computers. So, this is parallel processing plus scientific computing which we are discussing in this course is basically high performance scientific computing. Well, let us go to next slide.

(Refer Slide Time: 20:41)



So, what is a basic idea that I have a large job, its RAM size is large, it is a variables with gigabyte, megabyte, more than gigabyte more than 100 gigabyte, more than terabyte memory and one computer is asked to solve it.

So, it is typically a serial sequential or standalone computing, we will use this terms again and again. So, one computer solving a problem. If it is a large problem it is like one person has to lift a very heavy load and work with that. If it is extremely heavy he may not be able to lift the load. So, that can be the first difficulty. The load or the memory can be too large, so that one computer cannot load it or it will give you memory allocation issue how whatever way you try.

The other one is that if the computer still can take the load it will take astronomically long time. Astronomically long time means 317 years, like 4 cycles of Halley's Comet, it is an astronomical time.So, the solution becomes parallel computing that is ask many people to carry the load, a large piece of memory is used and many computers are together working for the same problem on the large memory. It needs much synchronization among themselves, so if they have to work with the large load they have to work together

Otherwise, that break the load into multiple small blocks and ask different computers to work on that. And there are small memory units, but there is requirement of more

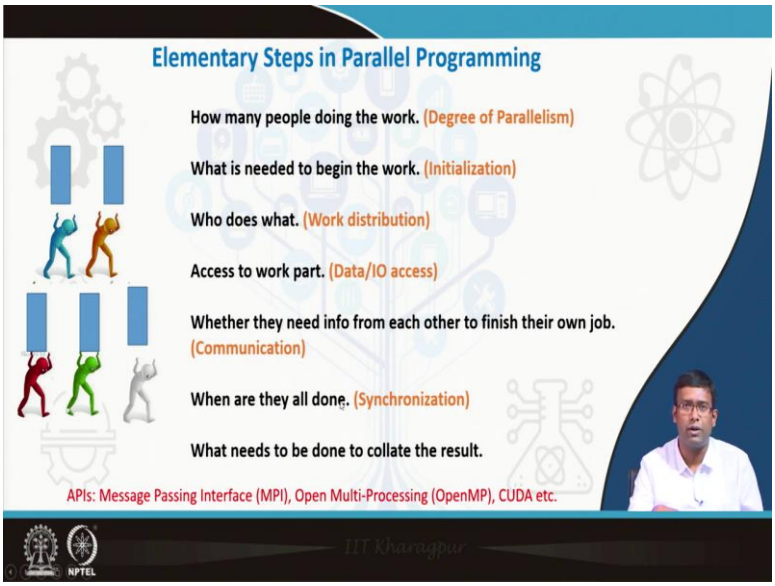
communication because they are having different pieces of the same load, they have to know where each others are going.

The left one is known as a shared memory load and the right one is known as distributed memory. These are the two paradigms in parallel computing. We can distribute the main problem, the memory of the main problem into several computers as small memory units and ask all the computers to take care of a small memory unit and work into it which is a distributed memory. Or we can take every entire memory in the in a single RAM or the way single RAM works in a single motherboard many pieces of rams are given and it practically behaves as a contiguous memory and ask many computers to work on different areas of this memory.

So, whatever the way we do it and we will discuss about shared memory and distribute memory in march detailed in a week or so. So, whatever we way we do it requires distribution that the job including memory part or without considering the memory part. It has to be distributed among many computers. There is a requirement of synchronization among the work of the computers and there is a requirement of communication.

And these are the component, this distribution, synchronization, and communication, these are the components which gives overheads .For this part ,what is doable in 1 hour by 1 computer in 10 hours, 4 computers cannot do that in 2.5 hours, there is some overhead. So, overall they take more time.

(Refer Slide Time: 24:20)



So, the elementary steps of in parallel programming will be how many people are doing the work, what is the degree of parallelism. In many cases we can see say, there is a block which can be divided only into 5 pieces, maximum 5 people can work on that. There is a matrix which has 100 grows. Maximum 100 computers can if we have to add 2 matrices with 100 (Refer Time: 24:48).

Maximum 100 computers can be asked that they will work on each of the row or maximum we can have 10 to the power 4 computers that they will work each of the elements. So, we cannot use more computers than that. So, what how many people are doing that work that depends on what is the maximum number of decompositions or distributions possible on the main work.

Also, what is the maximum number of computers available to me? What is my resource limitation? If I have 10 computers, I am working in 1000 by 1000 matrix, I cannot ask and cannot distribute into 100 jobs. That is, there will be some sequentiality in each computer; it can be distributed into 10 computers only. So, that is degree of parallelism.

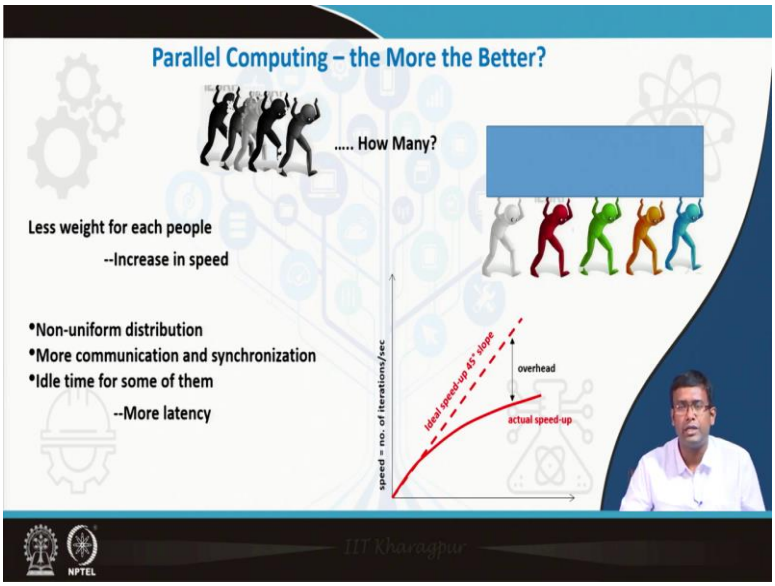
What is needed to begin the work? Some initialization is needed. So, that the people who are working together on a problem or the computers which are working in the single problem they know that they are working in this problem and these are the initial data needed to everybody. So, that they can in parallel or concurrently work on the problem.

Who does what? The work has to be distributed. Access to work part which data who which one requires and what will be the I/O by each computer. Whether they need info from each other to finish their own job. Whether there is a requirement of communication on fly while doing their own work or maybe some time it is required after their job is finished they need to send some data to some other computer because final work cannot be done without that. This is communication and when they are all done it has to be synchronized.

And then the results have to be collated. So, 4 pieces of chord going 4 rooms will not be of sense, if you are shifting a chord from one room to other. So, there is some requirement of synchronization and finally, all the pieces of chord, all the stands of the chord, the main beds has to be put together to get your own chord, so what is required to collate the results that is the next thing.

And for doing all these things we can see that distributing jobs into many computers, initializing them, communication synchronization a simple C program cannot handle this we need APIs, which are message passing interface MPI, OpenMP, open multi-processing, CUDA, these are the APIs. And precisely these 3 APIs how to use this to make a parallel program we will look through this particular course, well.

(Refer Slide Time: 27:12)



So, one question is that if we can increase number of computers will it be better. And say I have I can divide a job into 10 to the power 4 small parallel concurrent jobs. If I use 10

to the power 4 computers will it be better. And we will again look into it in much detail in subsequent classes. So, that will say that if we increase number of people for carrying a load, it will increase speedy as a sense less weight for each people.

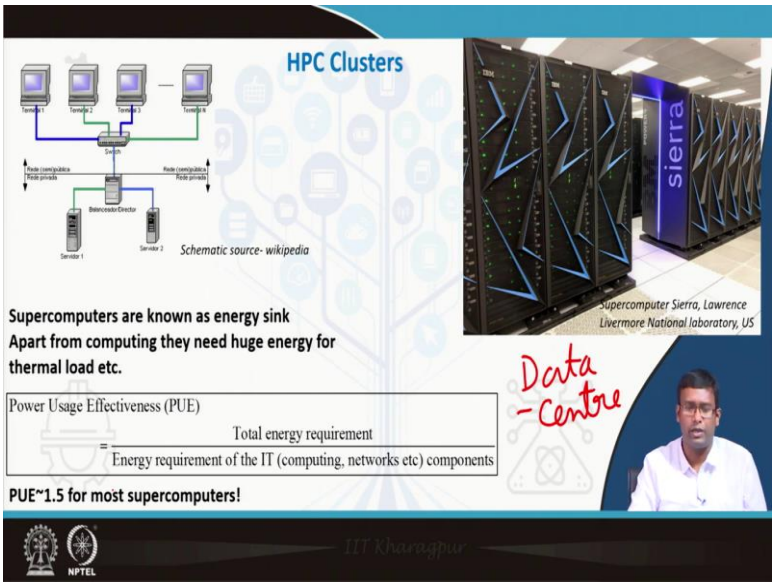
However, there can be non-uniform distribution, somebody is getting heavier parts, somebody is getting lesser parts or someone is sitting idle whether, someone is slowing down the process because he has much more weight. There is requirement of communication and synchronization and there will be ideal time for some of them. And therefore, which will add to latency or overhead.

So, as we are increasing number of computers we will see we are getting better speed in a sense that time requirement is reducing. However, this latency is also increasing. And will see that as we increase number of computers the speed or number of operations we are getting per second is not following a 45 degree slope, there is some overhead where the actual speed is reducing with increasing of number of computer, actual speed up is reducing. And there will be certain stage where it will be flattened down we should not use more computers after that.

But what is again the question is how many computers, how many processors we can use concurrently to solve a process, what is the optimum number. And this is something where an application scientist have to make a call that is it gives better speed up if you increase of computers, but we can see that the line is flattening down.

So, should we still you increase using more computers or should we say that will not use more further computer,and solve with this number of computers. It is still not decreasing, still increasing, but the slope is reducing, so what will be my optimum number. That is one of the issue we will be look into these things I am telling.

(Refer Slide Time: 29:15)



So, now these are solved in HPC clusters, high performance computing clusters. And while discussing the previous point it is important that we discuss this also. And this is one of the fastest supercomputer here which is many computers stacked in a rack and many racks are connected via interconnect switch. You can go to the to one of the controlling computers and ask them to distribute the job among many computers and work better.

And again this architecture we will look into in detail . From hardware point of view these supercomputers are known as energy sink. They absorb huge amount of energy, because running computers require a lot of energy as they run they warm up and the processors gets warmed up and you need cooling to keep the processors functioning. So, huge thermal load is requirement.

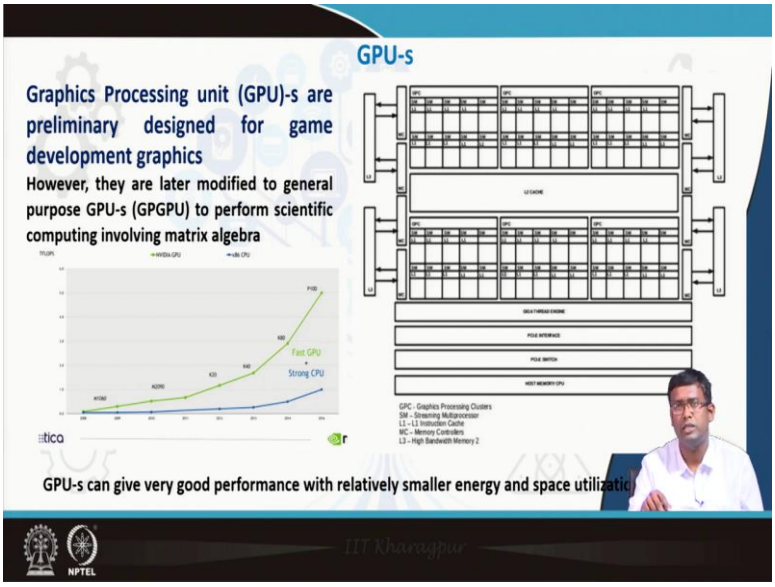
And one important parameter is power usage effectiveness of PUE for any supercomputing center or data center. You typically call the houses which hosts these supercomputers as data centers. So, the one of the parameter in seeing what is the energy requirement of a data center is PUE, which is total energy requirement divided by energy requirement by the IT or computing and networks unit. So, which the total energy is the IT part energy plus thermal energy, lighting energy, other losses etcetera.

So, it is seen for most of the supercomputers this number is close to 1.5. That means, if one unit of energy is spent for running the computer and networks 0.5 unit is spent for

AC, and thermal load and other relevant loads. So, if we run a computer we are not spending money or energy in terms of the electricity to run the computer, but we are also spending energy in terms of electric air conditioning and other loads.

Therefore, running multiple computers is also very expensive which is another parameter while deciding how many computers should we run that what is the energy requirement of the computing job. And supercomputing is not very energy friendly that is why we nowadays we talk about green computing energy friendly supercomputing etcetera, well.

(Refer Slide Time: 31:47)

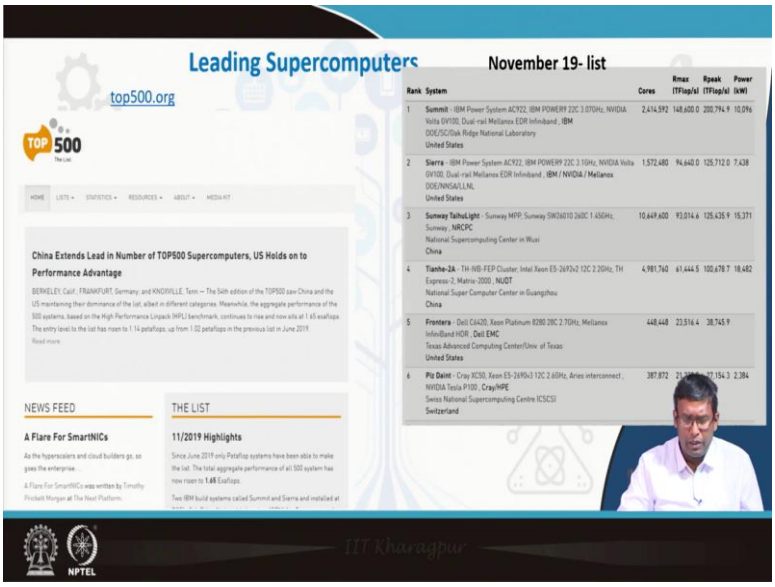


One of the ways of green computing is GPUs, where graphics processing units or GPU which are preliminary design for a game development, these are many small computing units with relatively, type of memory and very small cache connected in a board. And these are used as graphics cards in many computers, but they are also developed as general purpose graphics processing units and GPGPUs are shown to have good performance in scientific computing.

And it is seen that over last few years, last 2 decades GPUs give much more speed than CPUs. And they require much small amount of energy also you know. In a small, instead of building a large, having a large building with many computers, ACs etcetera, a small computer, a small workstation with a GPU card; GPU cards are costly than computers usual computers; with a GPU card you can do get good performance. So, these are one of the ways of having green computing.

However, because GPUs are very small cache and very rudimentary processor architecture, the computing in GPU has requires some special attentions. Very simply we cannot do GPU computing in terms of memory management as well as in terms of decomposing the jobs, a special attention and special training is required. So, they are known as very good very good performance with relatively small energy and space utilization.

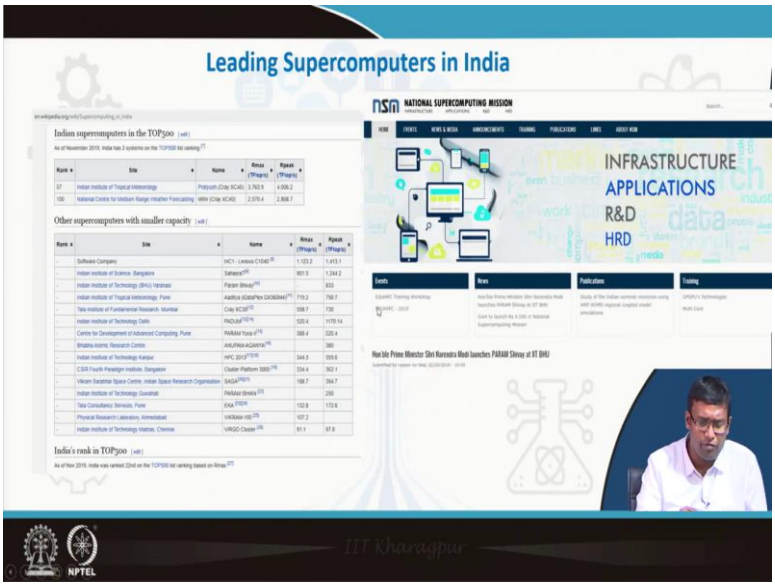
(Refer Slide Time: 33:22)



Now, if we look into the leading supercomputers, is important you can look into top 500 dot org which publishes the top 500 supercomputers. We can see the first test supercomputer gives 200,000 teraflops per second speed and it has 2 million CPUs, 2 million cores into it. And of course, there are few GPUs.

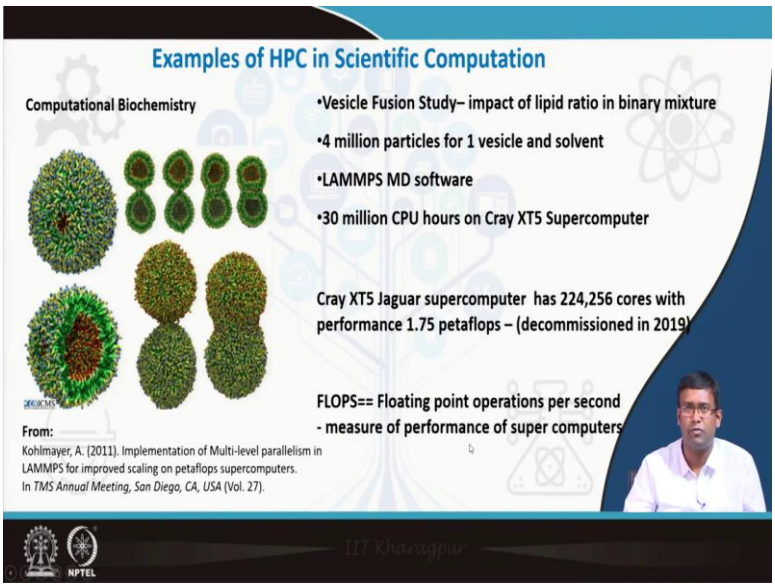
So, there are first few supercomputers and they are mostly from the United States, and China, and Switzerland. China and United States are very good presence in top 500 list. India is not doing very well, not in the top of top 500 list.

(Refer Slide Time: 34:06)



The 57 and 100 computer in top 500 list are from India, and these are Indian supercomputers which have my whose speed is much following much behind than the leading supercomputers. And in order to promote supercomputing in India, I talked about national supercomputing mission or NSM has been launched by government of India which looks into several aspects like setting up high performance, computing infrastructure, training manpower, promoting science which involves supercomputing etcetera and we hope to improve our ranking in top 500 supercomputers.

(Refer Slide Time: 34:46)

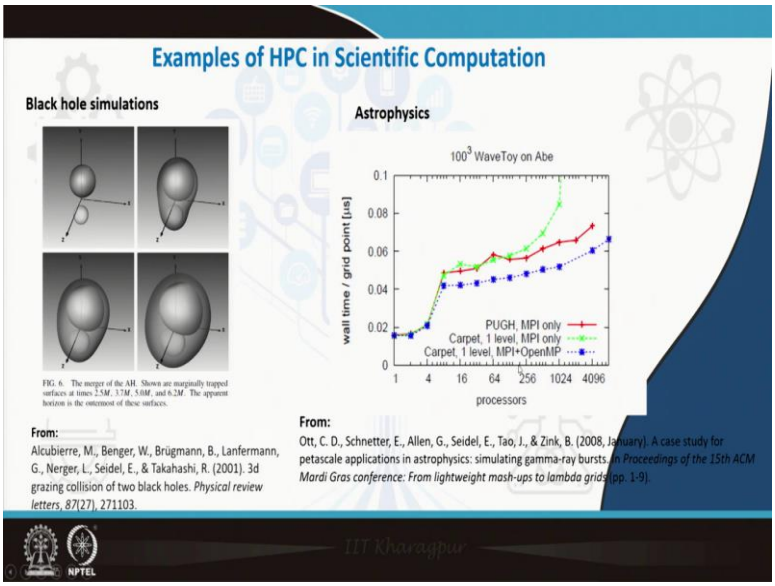


So, some of the examples. One example is computational biochemistry, where vesicle fusion study how lipids are propagating into the cell, vesicles that has been studied. It is seen that 4 million, 4 into 10 to the power 6 particles are required for 1 vesicle and solvent. And LAMMPS molecular dynamic software is used over 30 million CPU hours in Cray XT5 supercomputer. And this supercomputer was earlier it was in top 500 list, but it is now decommissioned in 2019. It has 224 into 250 224,256 cores and the peak performance was 1.75 petaflops.

So, now we will discuss about FLOPs which is one measure of how good our supercomputing is, which is the FLOPs is given by floating point operations per second. How many floating point operations this supercomputer can do in a second that is the FLOP speed. And, earlier we saw that our the turbulent flow calculation it was taking 10 to the power 7 second, 17 FLOPs.

So, if we can have multi petaflop system, multi, 10 to the power 9 FLOP floating point operation per second or even more than that we can instead of 317 years you can probably think it sorry we can in a year or in less than a year. So, this is important that how many operations can be done in a seconds. This a measure of performance of super computers.

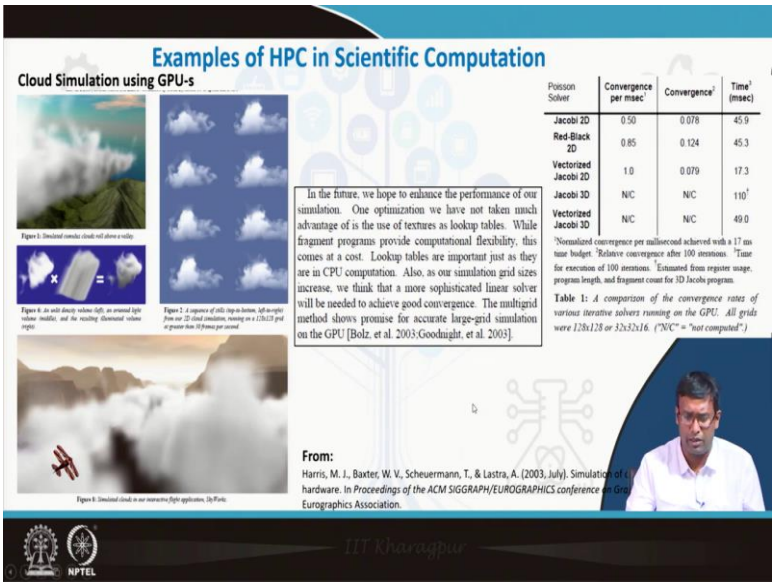
(Refer Slide Time: 36:13)



The next one is an black hole simulation and it is also seen with different uses of that MPI or a OpenMPI; API I talked about communicating across the processors and

increasing the number of processors. The time requirement from 0.1 microsecond per grid point comes down to 0.04 microsecond per grid point. So, there are different uses of super computer parallelization and different uses of hybridization of this APIs which can give better speed.

(Refer Slide Time: 36:46)

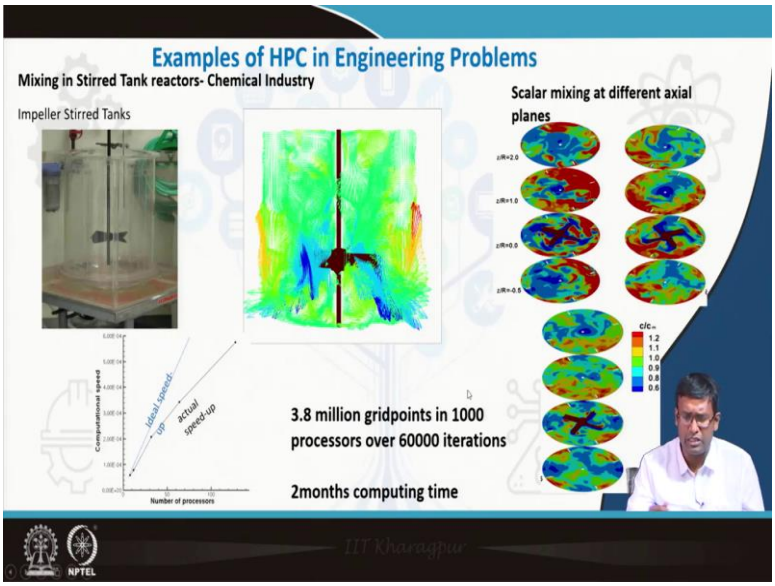


The next one is a cloud simulation using scientific computing. Here we can see that the time is in different computers is a function of how vectorizing is again parallelizing the solver, how the solver is parallelizing. And we write parallelization of the solver, what was a Jacobi 2D was taking 45 seconds similarly vectorized Jacobi 2D takes 17 second. We can bring it down.

And this computation is done in GPUs. But it was much primitive GPU; in some time in 2003 this computing is done; now GPUs are much faster. And one of the point is that our optimization have not taken much advantage of usage of textures as look up tables. While fragment programs provide computational flexibility it comes a cost. Look up tables are important as they are important in CPU computation, but that time the look up tables were not being optimized for GPUs.

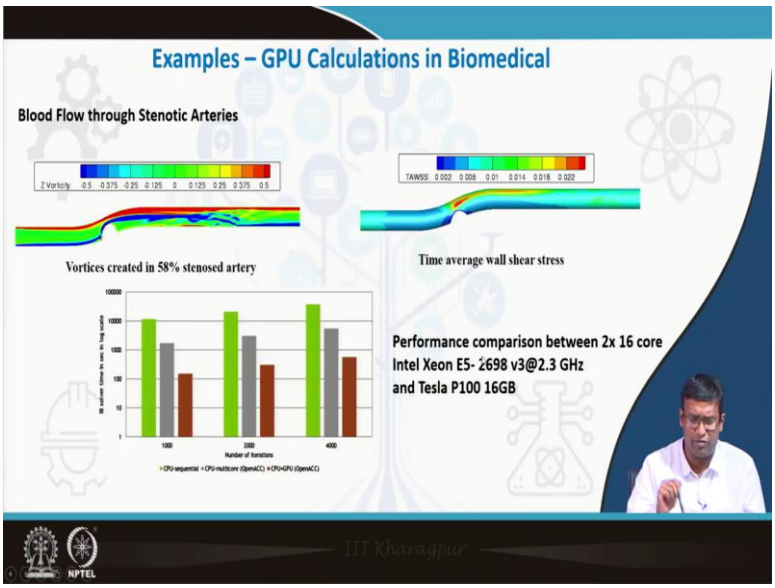
Also, the linear solver is Jacobi because there even in 2003 the advanced linear solvers were not ported for GPUs. Now, these are done and you can get much better speed. So, one of the forecast this group was making Harris in 2003 has come true that GPUs have made certain development and the computing speeds have increased.

(Refer Slide Time: 38:08)



This is in from our group, a computational chemical engineering problem in chemical industry how liquids mix and what are the mixing times etcetera. It took 2 months computing time in 1000 processors over 3.8 millions iterations, and we can see that what is the ideal speed up and how actual speed are varied.

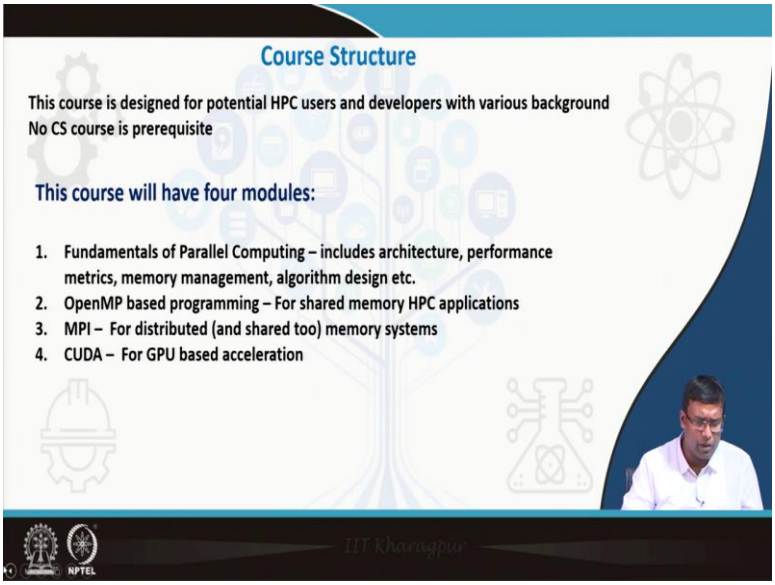
(Refer Slide Time: 38:26)



Some computation using bio medicals flow, in a stenotic artery, in GPGPUs and we can see compared to multi-processors, multi CPUs or single CPU, the graphics processing unit or P100 GPU has give almost 70 times better speed than a single processor and 10,

15 times better speed than 32 core GPUs; so CPU. So, GPU is much faster than multi CPUs also which came through biomedical computations, sorry.

(Refer Slide Time: 39:00)



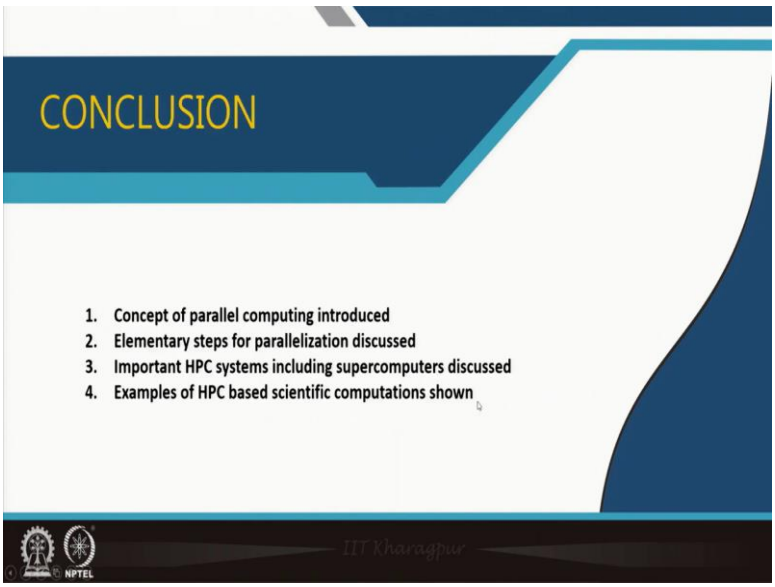
Now, one of the important part after you have seen some of the examples. The core structure I said it is designed for potential HPC users as well as developers. With various background you get some understanding of computer programming, some understanding of preliminary understanding of computers and its architecture, though the no computer science course is prerequisite as well as no scientific computing courses also prerequisite. Whenever required I will try to give you the preliminaries of this parts.

The 4 modules have fundamentals of parallel computing which will include parallel computer architecture, performance matrix, memory management, algorithm design etcetera. Then, we will se shared memory applications how OpenMP based programming work. For distributed and also for some of the shared memory cases how message passing interface or MPI programming works. And for GPUs, we will look into CUDA.

(Refer Slide Time: 39:55)

These are few books. One of the textbook will follow Grama, Gupta, Karypis, Kumar, Introduction to Parallel Computing. Then, Parallel Programming using C and OpenMP by Michael Quinn. Then the Canonical book on MPI by William Gropp from MIT press. For CUDA, will look into CUDA programming guide developed by NVIDIA, both in C and PGI CUDA which also look into FORTRAN and Shane Cook's CUDA programming guide. And Livermore National Lab has a good website for HPC. You can go into this website and get lot of tutorials out of it.

(Refer Slide Time: 40:33)



So, as a conclusion we have developed some concept of parallel computing, so some idea we got about what is high performance computing, parallel computing, some. The elementary steps of parallelization are discussed, but we will discuss in much detail about the design of parallel algorithms and how given one particular algorithm problem we can decompose it into multiple problems which can be solved concurrently in parallel computers. Some important HPC systems like GPUs, HPC clusters, and some of the leading supercomputers are shown and some examples based on HPC computations shown.

So, well, we finish the first introductory lecture and will start working on the later part of the course, especially we will take the module on fundamentals of parallel computing including architecture, the memory management and the parallel computer architectures. And we will see then, what are the matrix to understand whether the parallel computing program is performing well or not in subsequent lectures and then we will go to MPI, OpenMP, etcetera.

Thank you.