Foundation of Scientific Computing

Prof. T. K. Sengupta

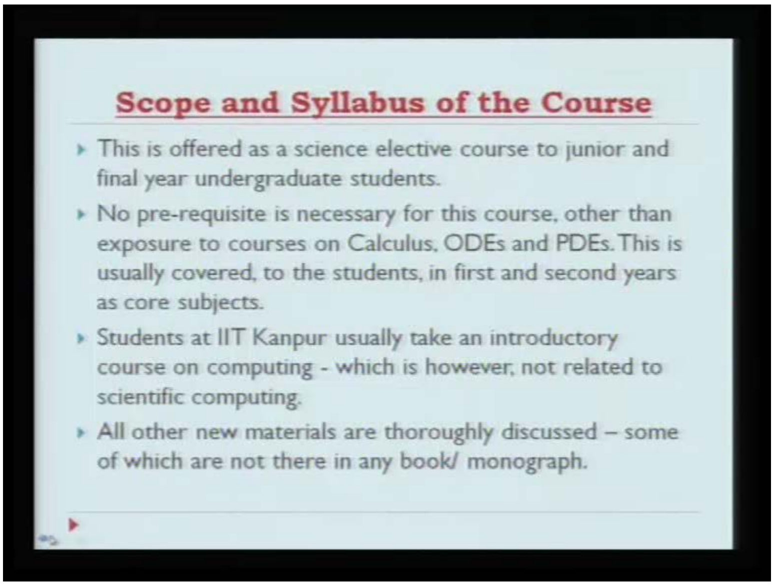
Department of Aerospace Engineering

Indian Institute of Technology, Kanpur

Lecture No. # 01

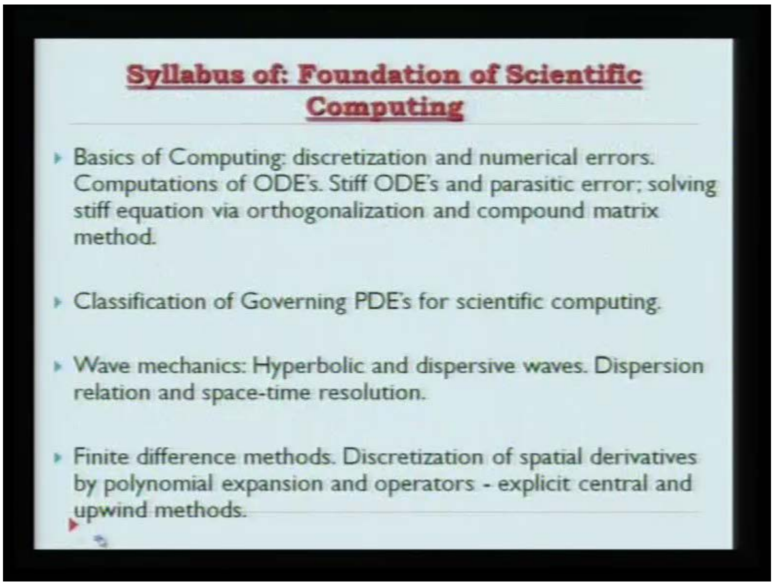
First of all, let us talk about the scope and syllabus of this course titled Foundation of Scientific Computing.

(Refer Slide Time: 00:20)



This is offered as a science elective course to our third year and final year undergraduate students. No specific prerequisite is necessary for this course other than exposure to courses which are given to our first and second year students on Calculus, ordinary differential equations, and partial differential equations. Students at IIT Kanpur usually take an introductory course on Computing, which is however not related to scientific computing, the subject of this lecture series. However, I must emphasize that any new material that will be taught in this course will be done so very thoroughly and for that, there is absolutely no need for any book or monographs.

(Refer Slide Time: 01:22)

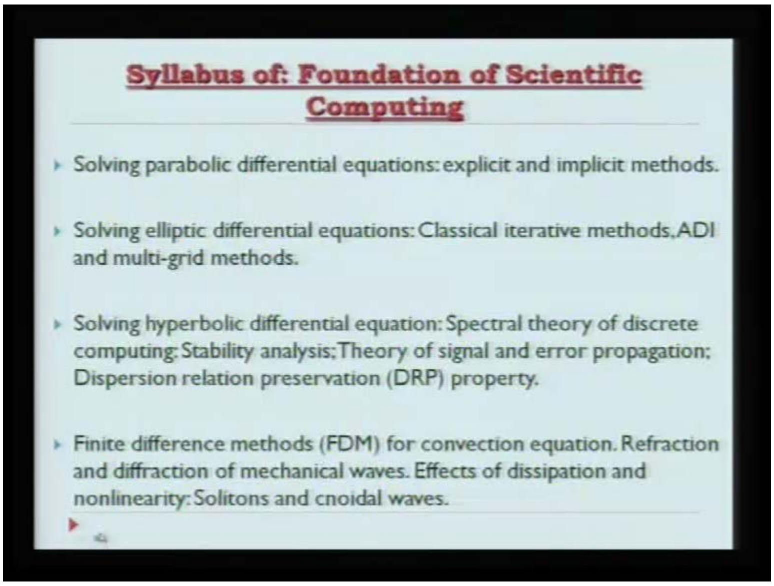


Now, going to the details of the syllabus for this course, we will begin by talking about basics of computing. We will thereby talk about discretization of governing equations and associated numerical errors. As usual, we will start with computations of ordinary differential equations and since some of the students may be familiar with it, we will introduce what is known as stiff differential equations. These are a very important class of problems and they need to be handled with care because this depends on parasitic error growth.

Solving stiff equations is performed via various routes. Here, we will just briefly mention about orthogonalization but spend a little more time on a new method which has come up over the last 30 years or so. It is called the compound matrix method. Once we are through with this, we will also talk about various two time linear methods of non-stiff ordinary differential equations and then, we will move on to classification of governing partial differential equations that one encounters in scientific computing.

We will see that these equations are classified into parabolic, elliptic, and hyperbolic equations. One of the interesting aspects of scientific computing is irrespective of whatever class of equations we solve, we computationally handle it as if we are treating the problem as a parabolic or hyperbolic equation and in that respect, wave mechanics is a very important issue that we should be talking about. There are two types of waves that one comes across: one is those governed by hyperbolic partial differential equations and the other one is called dispersive waves, which are governed by anything other than hyperbolic PDEs. All these waves are basically governed by dispersion relation. That actually in essence means how space and time variation is governed by these equations. This is an important concept. So, we talk about web mechanics in quite a bit of detail. Next, we will move on to finite difference methods, talking about discretization of spatial derivatives via polynomial expansions or by operator notations. In this regard, we will be talking about explicit central methods as well as upwind methods.

(Refer Slide Time: 04:16)



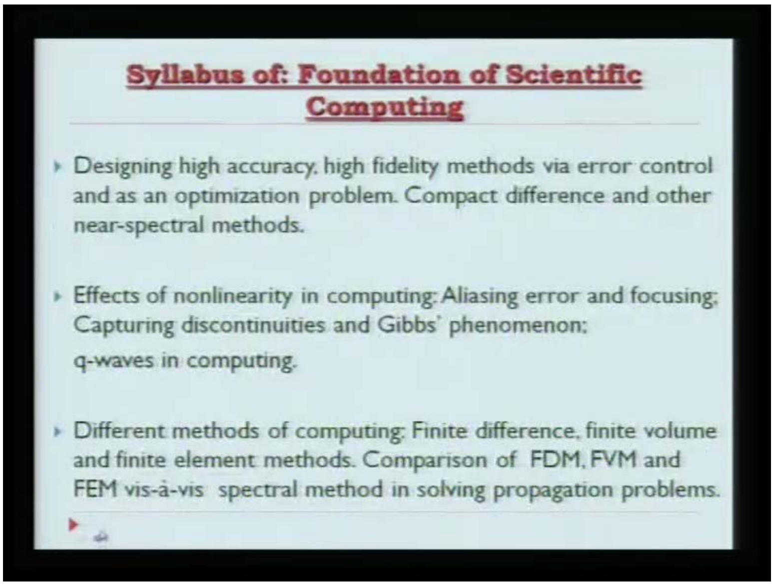
Once having done that, we will systematically go over to discussing about parabolic differential equations and in this topic itself, we will introduce explicit and implicit methods of solving differential equations. Once this is done, we will move on to elliptic PDEs. In this method, we basically talk about the classical iterative methods which will actually in essence treat this elliptic partial differential equation in a parabolic framework. So, we are coming along in that sequence.

We will also talk about what is known as alternating direction implicit method or ADI method. This was quite popular about 40–50 years ago, though it has been overtaken by other methods. One of the newer methods we will include in this course is called the multi-grid method. This is a huge area of research which can actually cover a full course itself, but we will just simply introduce the scientific aspect of multi-grid methods. Having solved the parabolic and elliptic PDEs, we will naturally move over to hyperbolic PDEs and while doing so, we actually introduce the spectrum theory of discrete computing because many of the ideas of computing have evolved while performing stability analysis of partial parabolic PDEs but essentially, the error is governed by methods or equations which propagate the error as waves. That is why we emphasize quite a bit of our time on wave mechanics.

While talking about stability analysis, we will specifically talk about the theory of signal and error propagation and once again, we will talk about dispersion relation preservation property. We want to thereby highlight that the space-time dependence of exact differential equation and numerical methods must be close to each other; that is what is meant by DRP property.

Now in this context, in solving hyperbolic PDEs we will be talking about convection equations, one-dimensional convection equations, we will also talk about refraction and diffraction of mechanical waves. We will talk about this with respect to surface gravity waves that one sees in the treatment of water waves by linearizing and treating it as an inviscid flow property. Subsequently, we talk about how these surface gravity waves are affected by dissipation and nonlinearity. In that context, we will be introducing the students to Soliton and a special class of periodic waves called cnoidal waves.

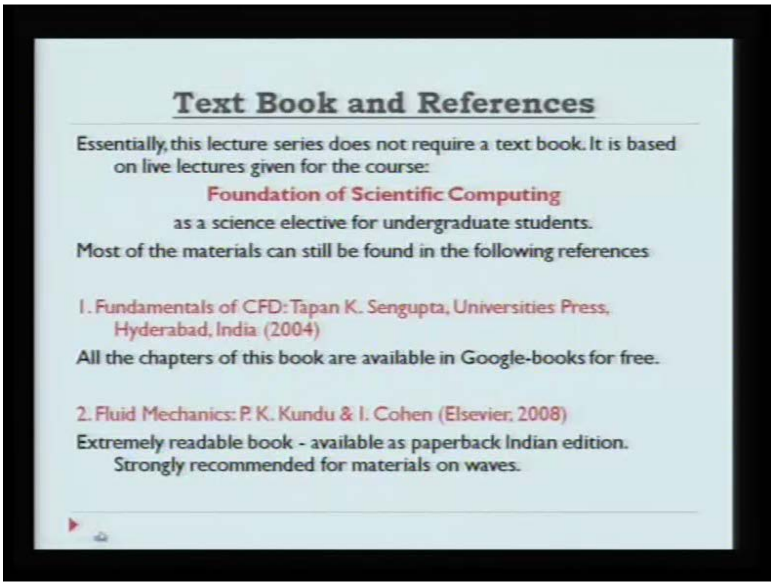
(Refer Slide Time: 07:40)



Having done that, we are in a framework where we can talk about how to design high accuracy, high fertility computing methods. That would involve severe control of errors. We can pose this itself as an optimization issue. In this context, we will talk about near-spectral compact difference methods and it will occupy quite a bit of our discussion space. Now, while talking about spectral analysis, we will introduce the effect of nonlinearity in computing. That brings about aliasing problem and also focusing of error. We may be already familiar with the effect of nonlinearity in compressible flows where discontinuities come about in terms of shock waves and associated Gibbs' phenomenon.

Here, we would like to talk about one particular topic which is very specific to computing. This is called spurious propagating waves which are called q-waves; this is what we will be talking about. Having exhausted various topics of finite difference methods, we will move over to finite volume and finite element methods and systematically compare these methods with finite difference methods. We will basically keep our attention focused on propagation problems because that is what is important in scientific computing. I think this will more or less finish all the available time that you have over this semester. We would like to basically give you some information about textbook and references.

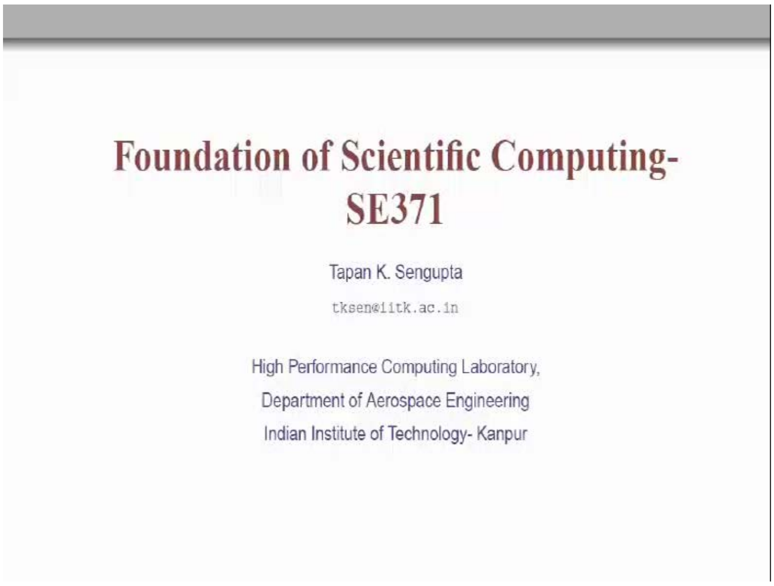
(Refer Slide Time: 09:34)



For this course, what we are going to do is we are going to record the live lectures given to the students here for this course on Foundation of Scientific Computing. It is a science elective. So, it is not necessarily a compulsory course; those who are interested opt for it. Most of the material can be found in the following references. Number 1 is a book that was written sometime ago by me, although it is not really up today.

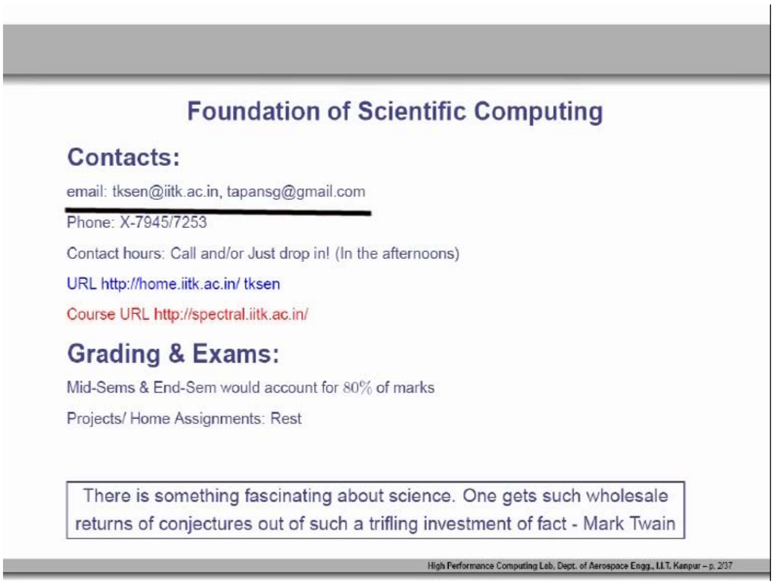
The good news is that all the chapters of these books are available on Google Books for free. So, there is very little effort one has to pay in downloading these materials. Also, of course, all the slides that I am going to show to you during teaching should be made available to the students of this course. I would also like to point out that there is a book on Fluid Mechanics by Professor Kundu and Cohen. There is a paperback Indian edition available for now and if anyone is particularly interested in reading materials on waves, this is an excellent book; I would wholeheartedly recommend this. Well, a warm welcome to all of you. Beginning of this semester; I suppose this is the first class. We have assembled here to talk about Foundation of Scientific Computing.

(Refer Slide Time: 11:17)



The course number is SE371. That is me. You could look at me here.

(Refer Slide Time: 11:32)



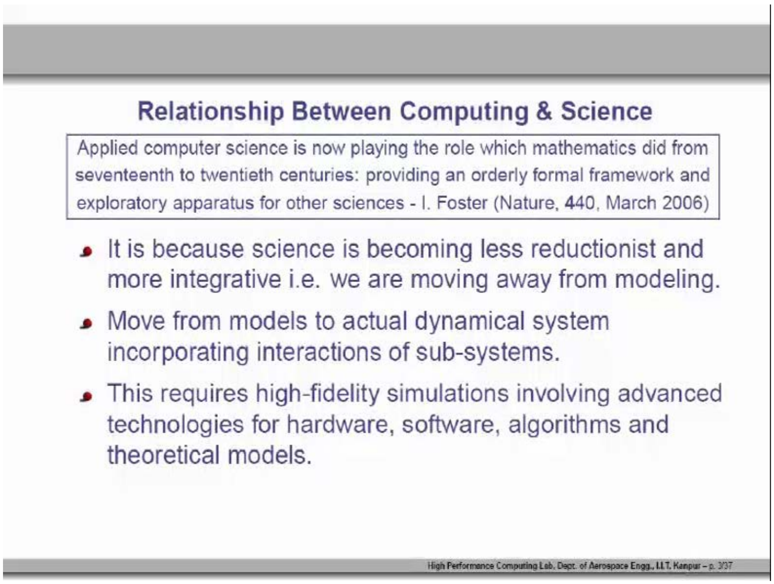
This is the way to contact me anytime you need to. Those are a couple of my e-mail ids. You can call me on internal numbers on 7945 or 7253. There are no fixed contact hours; you can just simply call me and drop in. Since I will be busy teaching couple of courses in the morning half, it would be preferable that you look for me in the afternoon. Do not bother about this URL but the next one, you should please make a note of it. This is a course URL we have set up on a server called spectral.iitk.ac.in.

We will load some material from time to time on this. I do not wish to keep it permanently, maybe for a week or so and then, we will download it. As far as the grading and exams are concerned, it is fairly straightforward. It will depend mostly on your mid-sem and end-sem; that should account for 80 percent of your total grade. Well, of course, do some projects. That should take care of that.

Now of course, when we talk about a subject like this, Scientific Computing, nothing could be said more eloquently than by what Mark Twain had said: fascination with science always yields some dividend and one could come out with fantastic amount of returns. Let me also tell you that scientific computing is not the type of computing we talk about, which you do using a commercial software bought in the marketplace.

To take the analogy to an extreme, it is almost like comparing astrology with astrophysics; it is the same thing here. Scientific computing is distinctly different from the so-called engineering computing that you do from those software in the market. So that is why let us probe a little bit more about the relationship between computing and science.

(Refer Slide Time: 13:39)



I just quoted this paper from Nature. It appeared a few years ago, talking about what would be the state of art as far as computing is concerned in the year 2020. I do not know why people get this fascination for 2020. Everybody writes about 2020. It has probably something to do with short-sightedness; people want to keep that way. Well, let us look at how science is improving. Science is becoming less reductionist. What exactly do we mean by that? This is that cause-effect model that we usually talk about: we have a definitive cause and we see an effect. That is your reductionist approach.

However, we see all kinds of systems around ourselves which are far too complex. To model those complex systems, we need to have a different approach. That is what we call as the integrative approach or inductive approach. That means that we are just no more content in visualizing systems by constructing abstract models, simple models, paradigms, and concepts; what we instead like to do is bringing in the complexity of a real-life system and that is what we talk about when we say we are going from reductionist to integrative.

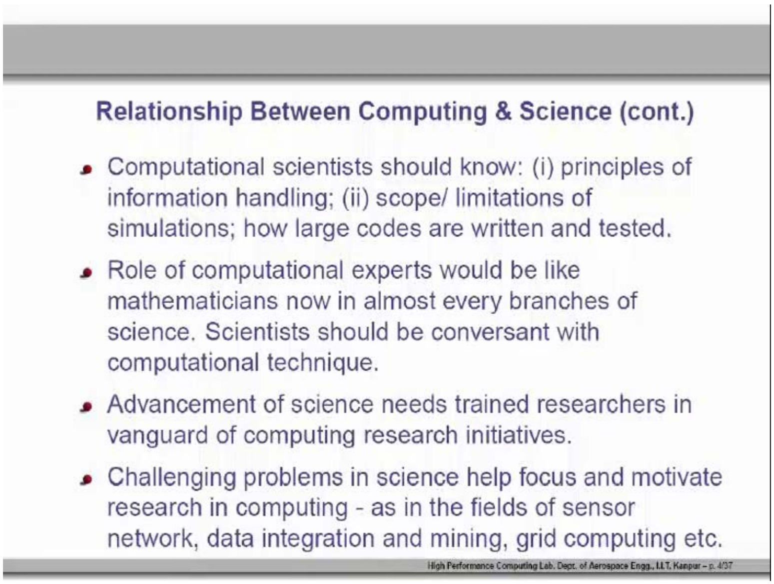
So, in that particular aspect, this statement is very cogent here. If we look at what we are going to do with applied computer science, we should play the same role that the mathematicians have been playing for two–three centuries, basically providing an orderly formal framework that will help us explore newer avenues for understanding science. So, we are just taking it to the next higher level. What theoretical tools cannot deliver, we aim to deliver some of them using computing.

Then, we will go from models to actual dynamical systems and when we are talking about dynamical systems, most of you are familiar. You have a transfer function that characterizes the dynamical system. You have input and then you get definitive output. The only thing is this dynamical system is not as simple as what you may have done in your Basic Electrical Engineering courses; it is going to be lot more (( )) the main dynamical system by itself may consist of many, many sub-systems and the input also could be multiple.

Think of a very simple example. For example, when we toss a coin, it is such a simple event. Even today, we cannot model it because of multiple inputs and because probably our model that we try to look at is not rooted to a simple system that we are used to seeing, a mechanical system governed by Newtonian law. So, what happens is basically that is why we resort to statistical tools. We cannot model the system. If we cannot do that, we incur some error. When our model itself is faulty, we call such error as the process noise; we do not even know enough about the process itself.

Then of course, you have the problem of modeling the input and finally even the measurement of output; measurement noise also plays a major role. So, when we take care of all of that, we actually go to our integral dynamical system approach. In the context of the present day, what we can do is we can fall back upon calculations which have to be very high fidelity calculations and we should draw tools from advancement arising in hardware, those arising from software. We could come out with better algorithms to solve the actual problem and all of this should also be supplemented with improvement of our theoretical model itself.

(Refer Slide Time: 18:23)

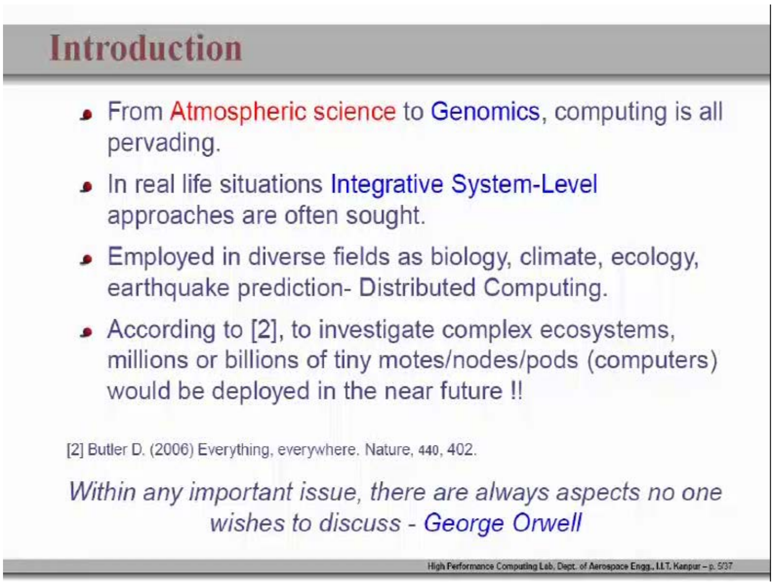


So, in the ideal scenario, computational scientists should know the principles, the way the information is handled inside a computer. That is rather important because you are looking for advancement; you should know how the data are put in and how they are handled. Moreover, one should also know the scope or limitation of the simulation itself. We have the general tendency to treat this as a black box system. So, we tend to shy away from knowing the nuts and bolts of what constitute those modeling tools. That should not do; if you want to contribute significantly and meaningfully, it is quite likely that you need to really know how this whole black box is working so that you can yourself contribute to its improvement. That means actually to know how to write large codes and how we test them in a modular manner.

These are not like the operating systems you buy; they keep on adding patch over patch and today, whatever operating system you have if you try to put it in a computer of 10 years ago, the memory will not be sufficient because that is a very faulty model of developing, even the operating system that we are saddled with. Now, drawing the analogy again, computational experts should view themselves as mathematicians like the way we see them functioning in different branches of science. At the same time, the theoretical scientists also should be conversant with computational techniques because then only, they would be able to meaningfully contribute and this synergy will of course take us to greater heights. So, we basically need scientists trained in these advanced methodologies.

The challenging problems that we find out in science should help us focusing and motivating our research in computing. You have seen some such benefit already accruing in the field of sensor network, data mining, data integration, grid computing, and cloud computing; you have a whole host of new branches coming up.

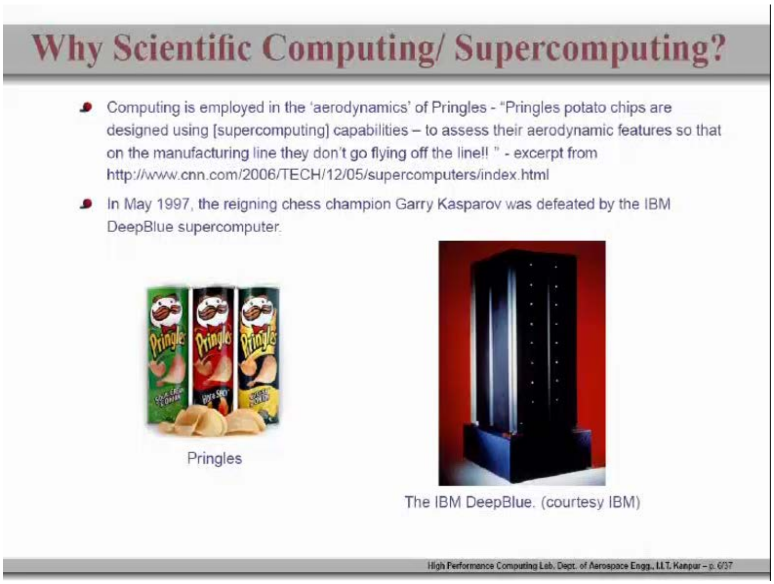
(Refer Slide Time: 20:57)



These are some of the things you can see where scientific computing can go – all the way from atmospheric science which has a very definitive history on fixed development to new subjects like genomics, which are probably only 20 or 30 years old. We have seen that in real-life situations, an integrative system-level approach will be sought. These are already being seen enacted in fields like biology, climate, ecology, earthquake prediction. They all essentially depend upon large computing resources which are distributed in nature. We will talk about distributed computing, a brief introduction of it.

This is one prediction that in the year 2020, we should be looking at complex ecosystems with millions or even billions of computers which will be called tiny motes or nodes or pods. They would be deployed to track complex systems. However, having said all this inspirational talk of we should be doing, what we are doing currently also leaves us some room to probe further and I am not talking about Conspiracy Theory. This is what George Orwell said: that within any important issue, there are many more other issues which people are reluctant to talk about. So, we will be talking about some of those issues and find out what is a better way of computing and we will set the thing.

(Refer Slide Time: 22:37)

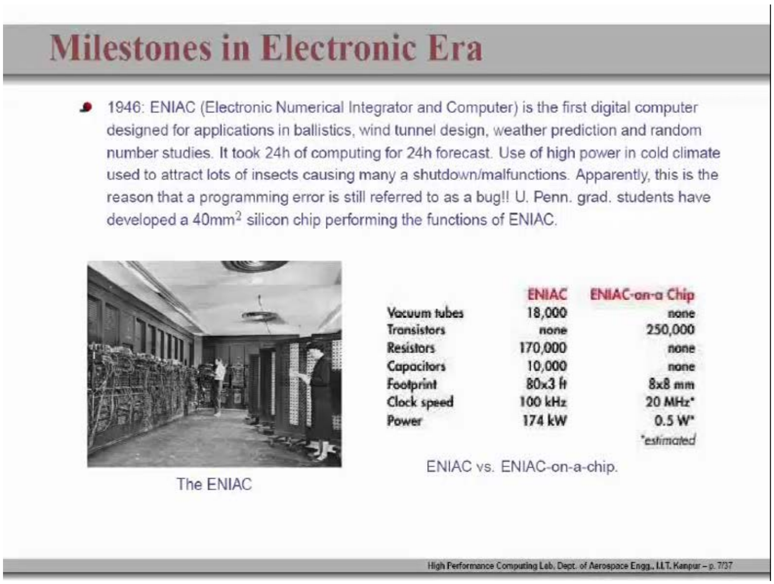


Now, this is a lighter aspect of computing. We all see from time to time that there is intense competition among diverse groups saying what they can do with computing and they generally use the word supercomputing. We never made out what really supercomputing is because these are the kind of diverse activities being claimed. For example, people talk about aerodynamics of Pringles and this is taken from a CNN site which talks about Pringles potato chips.

The story is that it is not really potato chips. How many of you know that? It is all synthetic savory, it is an industrial product. If you look closely at the content, you will find out that it contains less than 50 percent of potato flour; the rest of it is different types of ingredients. So, it is really not potato chips, but then people are making a big hoo-ha about it on a popular TV program saying that we need to know the aerodynamic features of this chips because they are going on a conveyor belt and we do not want them to fly off. Such a noble goal, huh?

Well on the other extreme, people have been looking at some real esoteric activities like creating a computer playing… sorry, the chess-playing computer. This began way back in 1956, I will give you a little bit of milestone shortly, and it took 41 years when a computer was really able to defeat a grandmaster. So, Garry Kasparov will be remembered among all the grandmasters for this dubious distinction.

(Refer Slide Time: 24:45)



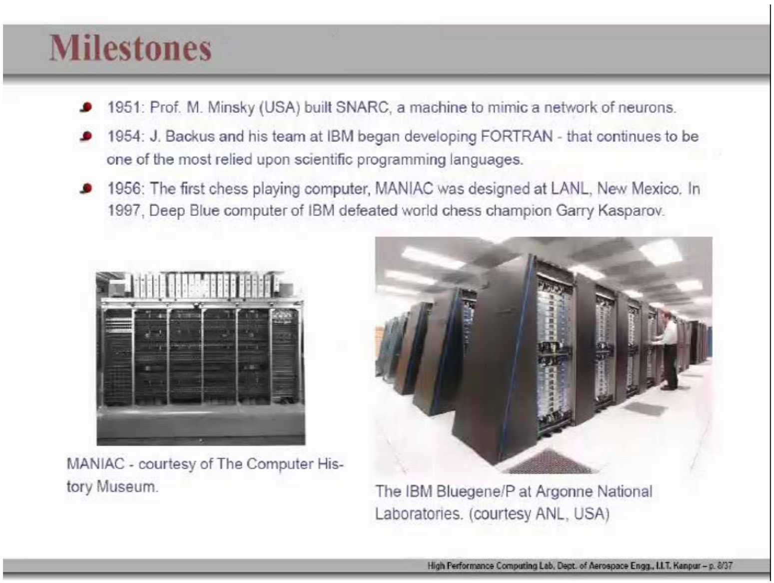
So, this is how it all began in 1946. The first computer was called ENIAC – Electronic Numerical Integrator and Computer. What was it doing at that time? Of course, war had ended but the preparation never ceases about fighting war. We always like war, right? So, people are looking at application in ballistics and then of course, better wind tunnels, to design deadlier aircraft or maybe even benign peaceful uses.

This is one thing that we always give credit to – this group of people who have been ((.)) do almost an impossible job, that is, weather prediction. So, they were trying to do better weather prediction and look at their success rate. To predict 24 hours of weather, they had to run the code for 24 hours. So, it is concurrent information processing. You look at the output and look out of the window and see what is happening, are you calculating it properly. That is a joke because talking about weather prediction is on a global scale and looking out of the window is micrometeorology; that is done in a different context. I do not think people were that interested micrometeorology in 1946. They were ((.)) at the global weather prediction.

Of course, our favorite is random number studies and you may be interested to know that this used to consume quite a bit of power. You can see it is 160 kilowatt. Where is it? Well, it is here, 174 kilowatt. That used to attract lot of insects from outside. This was housed in a big room. The footprint of the machine itself is 80 by 3 and those were made from those vacuum tubes, used to generate lot of heat and those bugs, insects used to be attracted. They used to come and sit on them, they would die and sooner or later, the vacuum tube will give up and then, they will say “We have a bug problem.” Even today, when we write a program and we get into some kind of trouble, we say we have a bug; that is the origin, it all originated there. ((.)) At that time, it was talked about in the context of hardware, failure of hardware. The University of Pennsylvania graduate students have actually developed the same functionality and a chip now which has a size of 40 millimeter square.

You can see the comparison here side by side. Power consumption has come down to 0.5 watt now because it does not have those vacuum tubes, it does not have those resistors and capacitors and of course, the size is really miniaturized. So, that is how it all began.

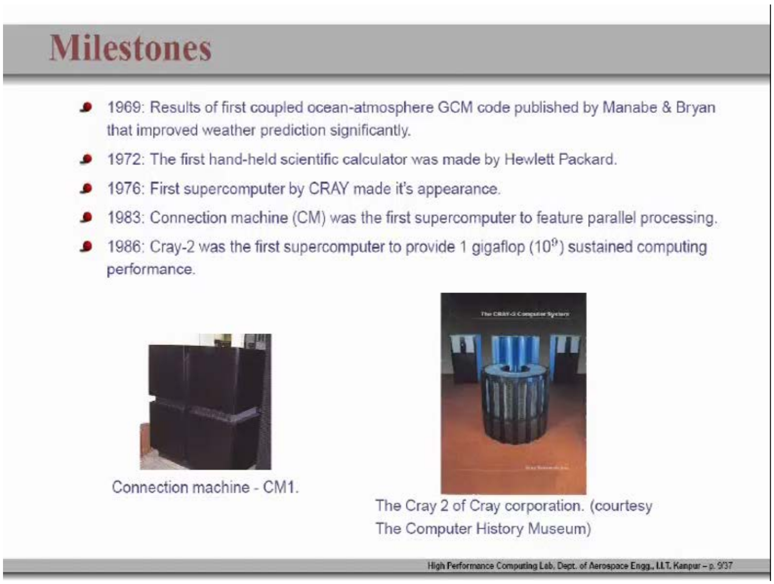
(Refer Slide Time: 28:07)



Then in 1951, Prof. Minsky built this computer to mimic networks of neurons in the brain called SNARC. This is a momentous day. Backus and his team at IBM started developing the scientific programming language called Fortran. I will talk about Fortran and other languages in a little while, but please do understand what Fortran started with, it continuous to deliver despite all the fashion statements that we hear from different languages from time to time. It is the surviving scientific language tool.

This is what I told you. In 1957, the first chess-playing computer MANIAC was designed at (( )) and of course in 1977, we know it finally came to the goal of defeating a grandmaster. So, this was the original startup point of chess-playing computers and this is where it shone up its potential. We will talk about this aspect in a short while.

(Refer Slide Time: 29:30)



Now, when we come to the actual usage of this high-performance computing that came about, one of the nice milestones was achieved in 1969 when a general circulation model for coupled ocean and atmosphere code was run by Manabe and Bryan and this significantly improved weather prediction. Needless to say that it was helped because at the same time, I suppose, our satellite technology was also improving and as I told you about integrative approach, weather prediction is one such integrative approach because anything and everything that we can get from the atmosphere, we try to put in those information in our weather prediction code and all those came around the same time.

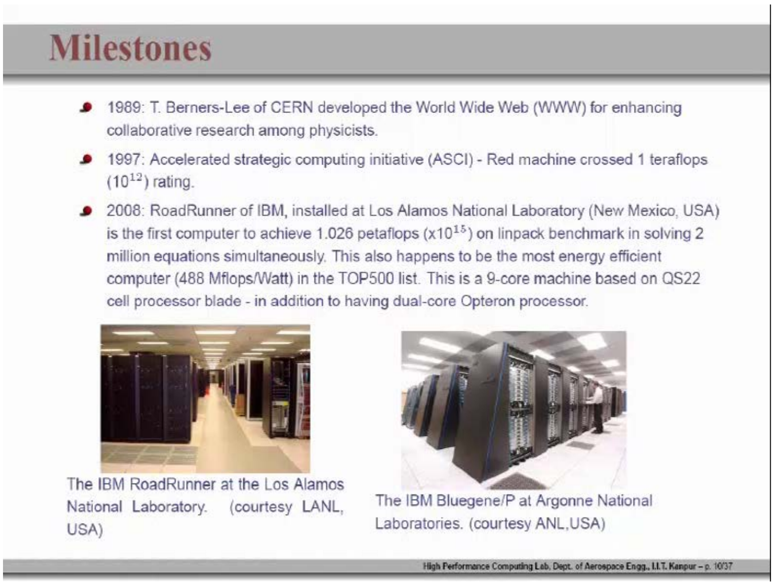
Our computing ability became better. Then, we have all these other inputs coming from different technical sources; that really significantly improved weather prediction. In a temperate climate like in Europe or in North America, about latitude of 30 degree north, the weather prediction has really improved significantly. Now, I suppose people can get very good quality weather prediction for 48 hours upfront. This is with the qualifier that even today also, the same set of people will not be able to track a cyclone very clearly. Those are problems of fluid mechanics; if we have time, we will talk about them but general day-to-day weather prediction with some weather events occurring, those could be done, are done quite routinely now.

Well, 1972 was the time the handheld calculator made its appearance and it took only four years before Seymour Cray launched the first supercomputer. Now, I will give you a formal definition of supercomputer. You would be very interested to know what that supercomputer delivered in 1976, I think all these laptops will beat that supercomputer hollow in terms of computing power. So, supercomputer is a euphemism; it is a fashion statement dependent on the time frame we are looking at.

Now, we have more computing power at our desk than this supercomputer delivered. In fact, Cray-2 was delivered in 1986 and it gave us a computing power which is referred to by a first gigaflop machine. All your laptops and desktop PCs today give more than a gigaflop. So, you can see that that much of computing power you have. Well, it depends on us, how we use them. Are we using them as a supercomputer or a chat network? That depends on us.

Now, one of the major aspects of supercomputing in science depends on how we solve the problem. One of the algorithmic developments came about with the advent of parallel processing. So, you are basically solving a chunk of integral problem in parallel by slicing it into bits and your computer is going to work on them bit by bit and integrate the result; then, you go to the next step and so on and so forth. So, that is the parallel machine; the concept of parallel processing came about with the advent of the Connection machine machine. The CM machines came and that also revolutionized scientific computing.

(Refer Slide Time: 33:39)

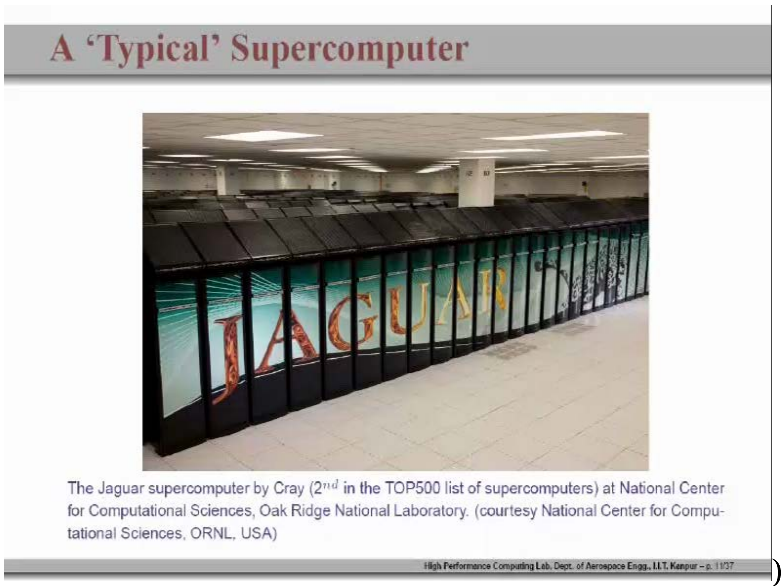


Of course, this is what we all depend on now. The World Wide Web was developed by the physicist Berners-Lee. He thought he could synchronize the computing power in different parts of the world and solve big problems of physics. We know how the World Wide Web is used today. People are still not solving those physics problems for which it was actually thought it would be used. The next milestone came about in 1997, where again we see a thousand-fold increase in computer power from gigaflop to a teraflop machine achieved by ASCI Red machine.

To cut the story short, when we look at 2008, this is the fastest supercomputer available, once again at Los Alamos. When it was introduced in 2008, it had 1.026 petaflops. How do we benchmark these speeds? They are basically done using a set of linear algebraic equations available in this package called linpack and in this activity, they were actually solving two million coupled equations simultaneously. That is a moderate number, not a big deal. I mean, we ought to be ready to do that any point in time in solving big problems.

What was more interesting about this Roadrunner was that it is one of the most energy efficient computers. For each watt spent, it delivers 488 megaflop power. You probably know the existence of this list called TOP500. You can go to the Web and you can find out. Every year, I think in the month of June or November, they update the list. So, the list that I am showing you is up to date. I think it has crossed 1.05 now. So, it is a marginal improvement over the last 6 months. It is probably because of the addition of more processes but we are still hovering around 1 petaflop rating.

(Refer Slide Time: 36:10)



This is the number 2. This is really fancy, is it not? This is a Cray computer, Jaguar. This is at the Oak Ridge. This also comes a close second.

(Refer Slide Time: 36:32)

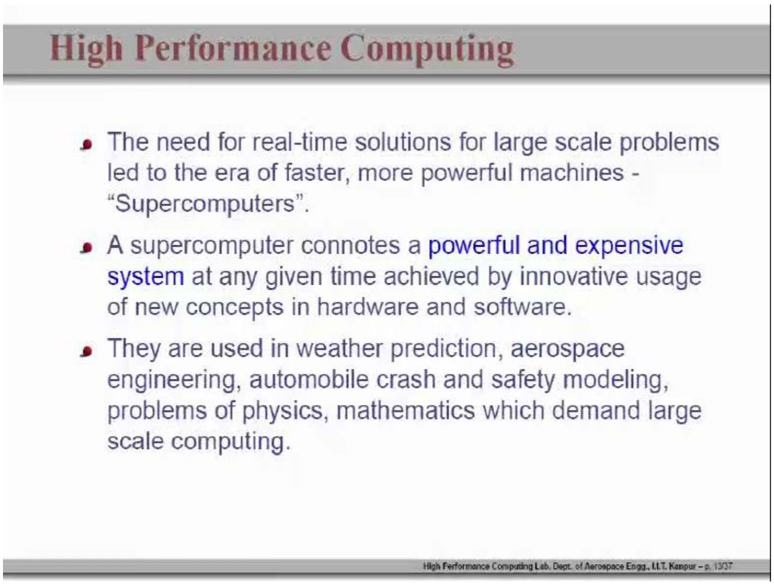
Rank	Site	Computer/Year Vendor	Cores	Power kW	FLOPS(peak) TFLOPS
1	DOE/NNSA/LANL United States	Roadrunner - BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 GHz, Voltaire Infiniband / 2008 IBM	129600	2483.47	1456.70
2	Oak Ridge National Laboratory United States	Jaguar - Cray XT5 QC 2.3 GHz / 2008 Cray Inc.	150152	6950.60	1381.40
3	Forschungszentrum Juelich (FZJ) Germany	JUGENE - Blue Gene/P Solution / 2009 IBM	294912	2268.00	1002.70
4	NASA/Ames Research Center/NAS United States	Pleiades - SGI Altix ICE 8200EX, Xeon QC 3.0/2.66 GHz / 2008 SGI	51200	2090.00	608.83
5	DOE/NNSA/LLNL United States	BlueGene/L - eServer Blue Gene Solution / 2007 IBM	212992	2329.60	596.38

Let me tell you that all the TOP500 machines that we see ((.)) process power and these are the top five that you can see: there is the Roadrunner, this is the Jaguar. This is the new kid in the block; it just made its announcement last month, a German group but you can see the number of processes they are using. Of course, we do not call them any more processors. What do we call them as? Cores, because each processor can have CPUs with multiple cores. Most of you probably would be using a Core 2 Duo or dual core or quad core machines.

This one actually ((.)) processors. You would be interested to know that this was actually developed for PlayStation by Sony. These are graphics processors. Graphic processors have an enhanced ability for number crunching. So, they are very fast. In case any of you fancy to put up a fast computer together, the basic unit should actually be drawn from this graphics processor.

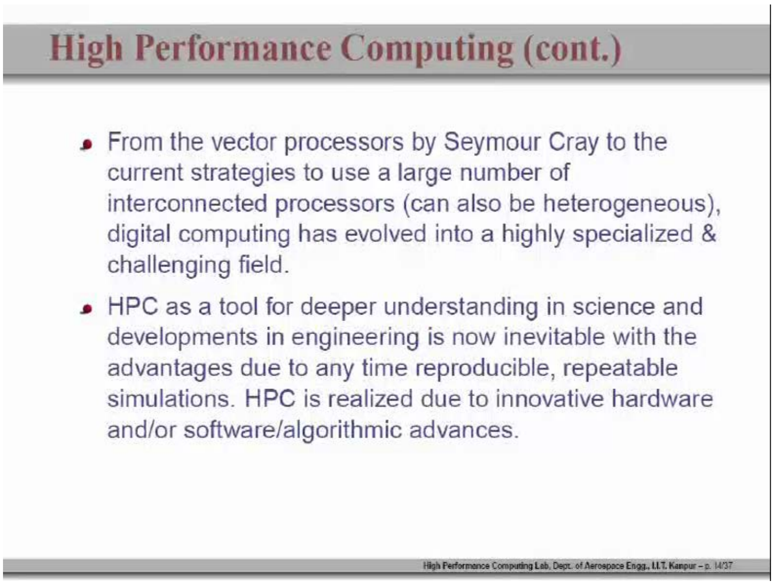
You can actually see different machine use different cores. This has come about to about… crossed a petaflop barely but using about almost 300,000 processes. These are the top 5 that you see in the TOP500 list. Let me also tell you that do not pay too much of attention on this kind of listing because there are lots of people, lots of organizations who do not want to divulge what they are doing, especially the defense research in USA will never compete for this. So, do not put too much of faith in this list. Although in India we are used to newspaper headlines that we are number 5 or number 18 now, those are not so good but there are certain good aspect even for those activities. I am not belittling them; they are good but let us keep that aside.

(Refer Slide Time: 38:52)



Now, you would have that you want to solve a real-time problem with a faster machine, more powerful machines. So, it connotes a powerful and expensive system. All those machines actually cost more than hundred million dollars. Of course, they are used for weather prediction. We use it quite often, aerospace engineers; even for designing new cars, for looking at its crash-worthiness or safety aspects, such activities are taken and of course, any problem on physics and maths, which require large-scale computing, would use the best in the market.

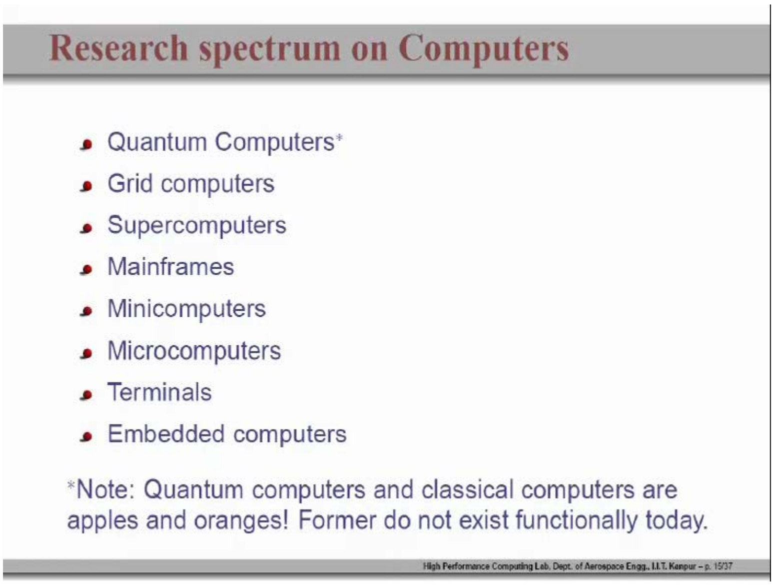
(Refer Slide Time: 39:44)



Apart from those benchmark numbers, let us see what really distinguishes these high-performance computing machines from the other machines used for known high-performance computing. It all began with the idea of Seymour Cray where he actually conceived of having a vector computer. So, if you have an unknown array, you break it down into few vectors and process them as vectors; that is how all this began in that 1976 machine.

Right now, we can go to not necessarily vector, we have machines which use very, very large number of interconnected processors and these processors need not necessarily be homogenous; they need not belong to the same class or category they could be heterogeneous. So, you could have a room filled with different types of PCs and you can put them together in a cluster and you can derive enhanced power which all originated in vector processing and the current activity is what we call as parallel computing.

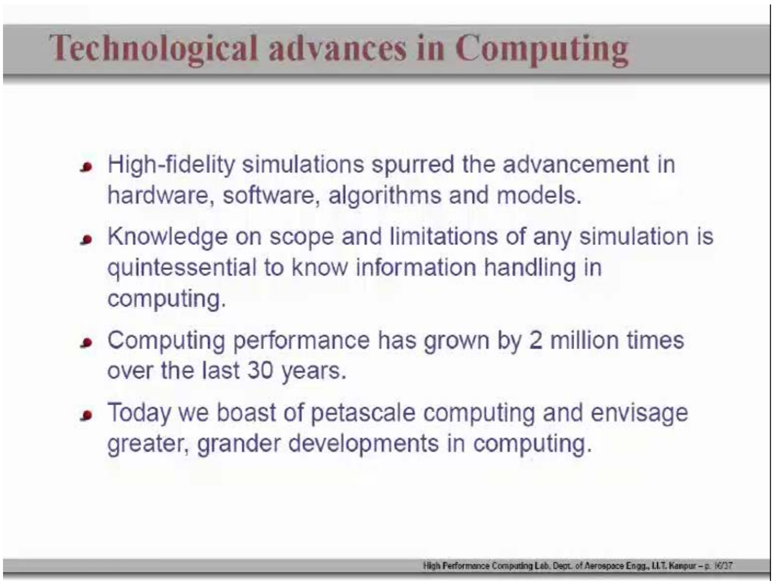
(Refer Slide Time: 41:20)



With this definition of high-performance computing, we can also see that there are other ways which we talk about. You may have heard of quantum computers, but let me warn you quantum computers and the traditional computers that we talk about are apples and oranges. They do not perform the same task, they are not even fruits; even apples and oranges are being charitable to quantum computing. They actually solve an entirely different class of problems.

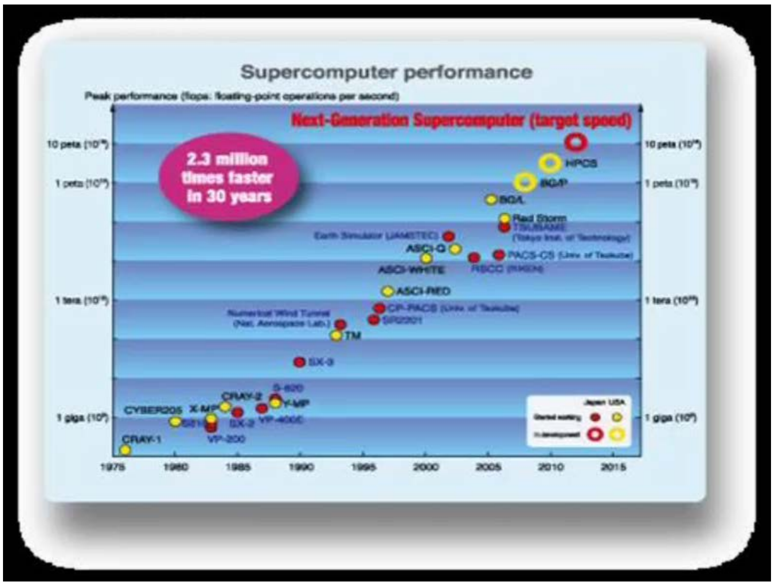
What you do with classical computers I do not think any quantum computer will be able to do right now. Then, you may have heard of grid computing. That is probably what the World Wide Web was conceived of. So, now that is being exploited. Grid computers, then we know supercomputers, we have the mainframes, we have the minis, micros, then of course we have the front end, the terminals and we may have even embedded computers; that should come in most probably in the near future in all white goods – domestic purpose usage; your fridge, your TV, everything should probably come fitted with computers. So, we have all possibilities looking at us.

(Refer Slide Time: 42:47)



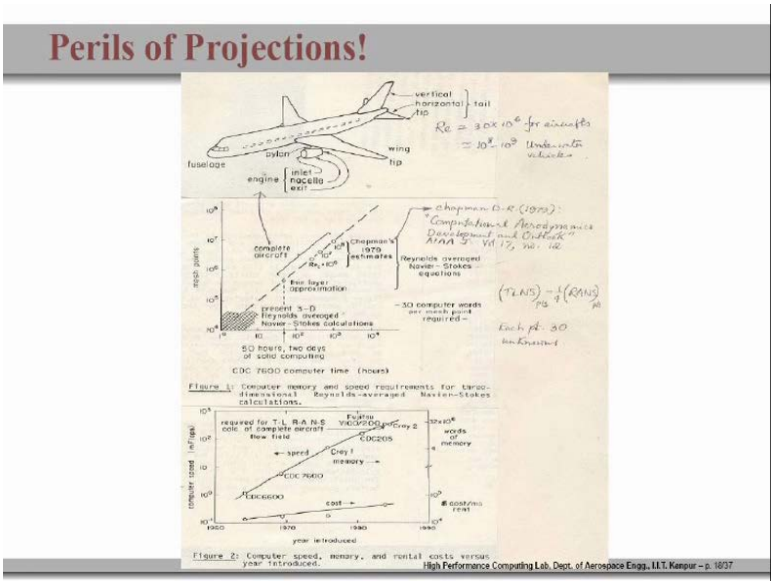
We have talked about it. The main thing to realize that computing performances grown by about two million times in the last 30 years and these last 30 years means I am talking about 1975 to 2005. Today, we have petascale computing and we hope to arrive at a time when we would be doing great things, greater things.

(Refer Slide Time: 43:13)



This is a sort of a comparison between USA and Japan. The red dots are the Japanese ones and the yellows are the USA machines and we are right now here (Refer Slide Time: 43:26). This is one petaflop; so, we are already here. Open circles are the ones that are projected, but we have already reached here; in 2008, we have crossed 1 petaflop rating. Japan plans to get to 10 petaflop machines. So, this is a 10 petaflop machine. They are hoping to get it by 2011 or 2012. As you can see, we started from here with the first supercomputer Cray-1 and then, we reached in 1980 the first gigaflop machine. Then, this was the teraflop and then, this the petaflop. So, the march is on.

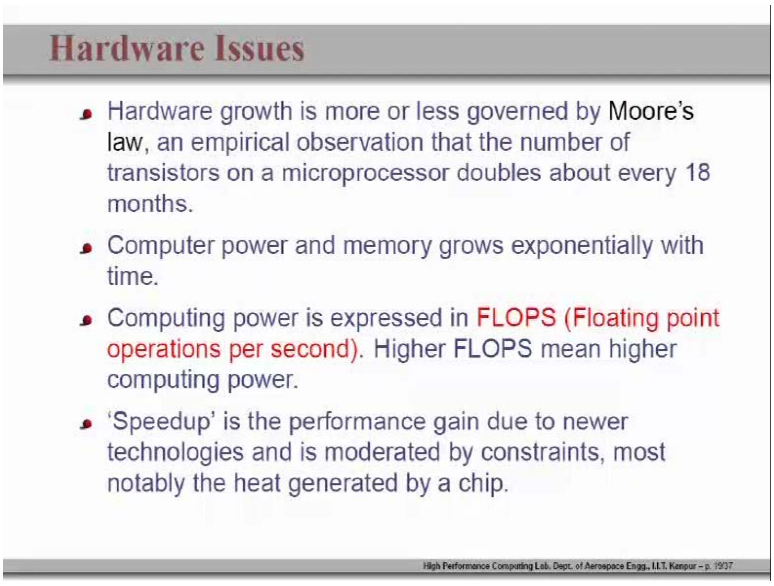
(Refer Slide Time: 44:21)



Perils of projection. You ask anyone and everyone, they will tell you “Give me this and I will deliver this” and this is what was predicted by a scientist from NASA ((.)), Dean Chapman. He wrote this paper Computational Aerodynamics Development and Outlook in 1979 and I just want to draw your attention that he said that when we get this kind of number of points, ((.)). So, he showed that when we reach around one million points, we should be solving the flow ((.)) complete aircraft.

Now, of course, you can do it in your PC, right? You have more memory than that. How many people are computing flow ((.)) full aircraft. Not many I know of. Of course, aircraft companies are doing it but that does not come with this kind of projection; they work on different issues. Well, probably, we human beings are optimists so we always look at the positive approach of what we can get.

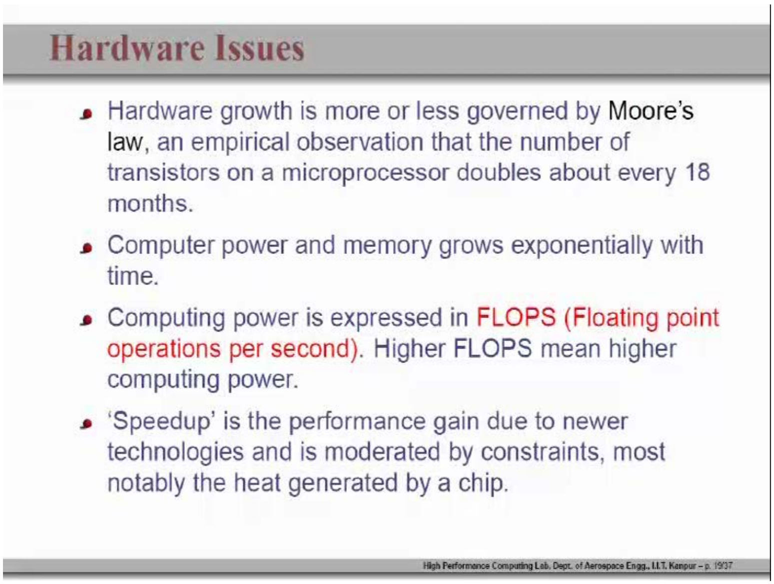
(Refer Slide Time: 45:37)



This you probably all know. Anything that you cannot prove you can proclaim it as a law; so, Moore did that and he said that every eighteen months, the number of processes in a chip doubles. This excludes all those supercomputers. Supercomputers actually outstrip Moore's law. So, there is already always an exception to Moore's law. I do not know why computer scientists call this as Moore's law.

This computer power and the memory actually are growing exponentially with time and this memory that we are talking about is the RAM, not the hard disk part; those grow actually even much faster at double exponential rate. We have already said that computing power is denoted by floating point operations per second. That means that if you have a higher flops machine, you have higher computing power and then, we can keep on trying to speed up the performance gain of the computers but we must be alert to the constraint that when more and more number of processors are packed in a chip, we are actually creating more heat and that is a significant constraint.

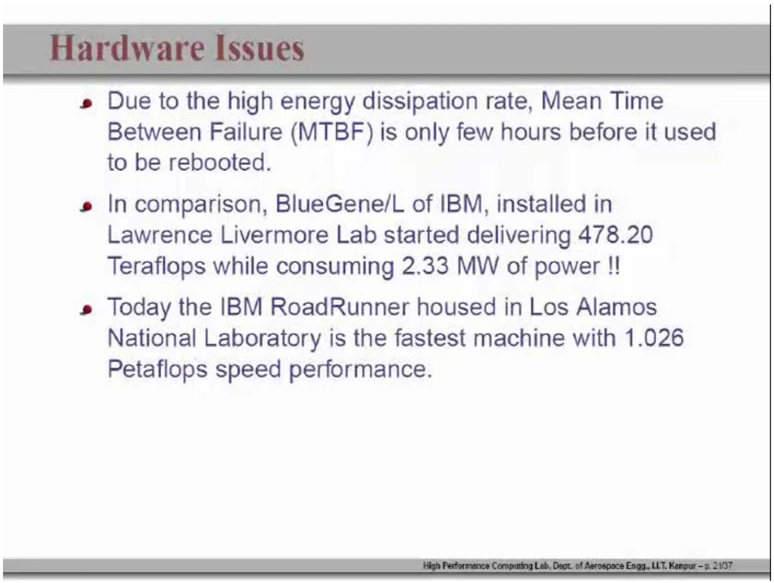
(Refer Slide Time: 47:03)



For example, by 2010, people project that there will be one billion transistors in a chip and that would create about 2 kilowatts of thermal energy. If you look at the energy density per unit area, this heat creation is more than what actually nuclear reaction does on a per unit area basis. So, it is a fascinating figure to keep at the back of your mind that this is a serious problem.

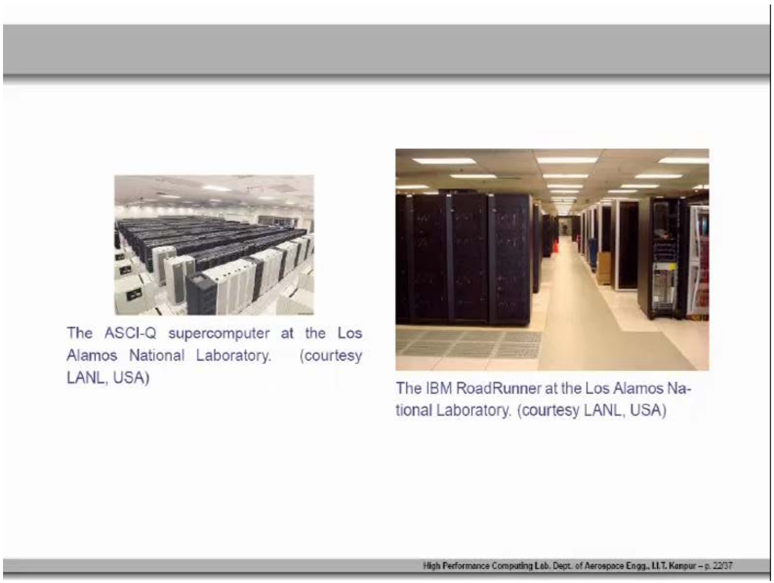
For example, look at this ASCI-Q computer which was housed in Los Alamos in 2002. It delivered about 30 teraflop, it had about 12,000 processes in 2048 nodes, 12 terabytes of RAM, and 600 terabytes of disk storage. This was the building that was housing this, 300,000 square feet. It had to have those cooling towers, rows of them. Those cooling towers would radiate heat into the atmosphere. If the computer needed 3 megawatts of power, the cooling needed 2 megawatts of power. So such a serious problem that you encounter. What happened at the time of introduction was that every time a computer is started on, it would run for few hours and then it had to be rebooted.

(Refer Slide Time: 48:39)



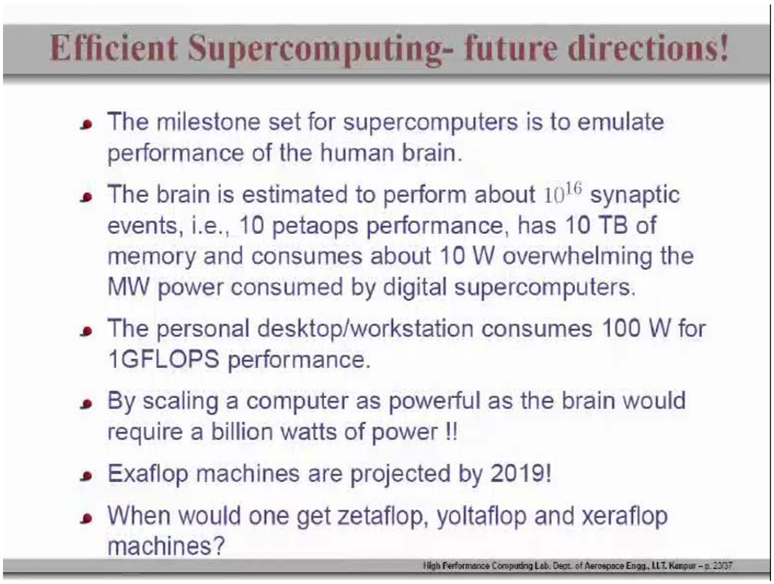
We have this nice thing called mean time between failures, MTBF. At that time, this MTBF was only few hours for ASCI-Q. Of course, in comparison, today, for the Blue Gene of IBM that was installed in Lawrence Livermore, the power significantly came down to 2.33 megawatts and it had a much higher power rating you can see; as compared to 30 teraflop, this is 480 teraflop. This is the fastest machine that we have already talked about.

(Refer Slide Time: 49:14)



On the left is your ASCI-Q and this is your Roadrunner of today. You can get a pretty good idea of what supercomputers look like. Where do we go from here?

(Refer Slide Time: 49:34)

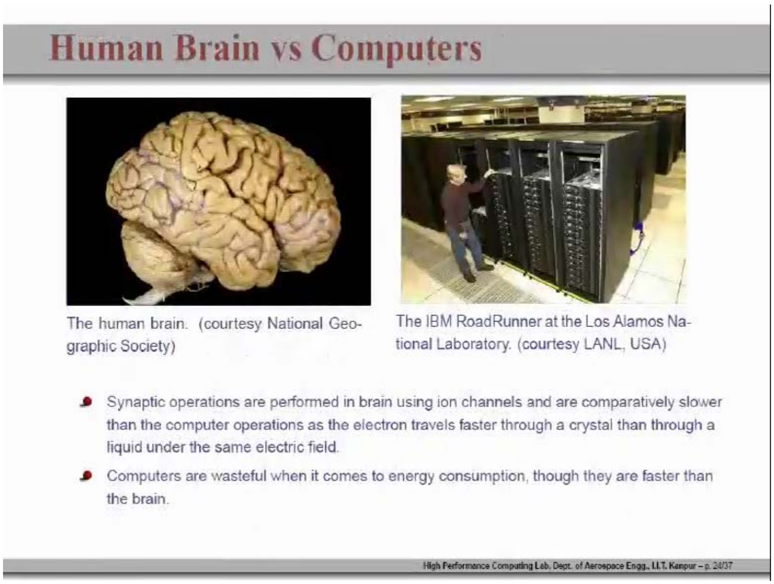


The milestone set for supercomputers is to emulate the performance of a human brain. Why? It is because it is still a little into the future and we can continue to give excuses for some more time for not being there. What the human brain does is it performs about 10 to the power 16 synaptic events. That is equivalent to something like 10 petaflop. The human brain has only 10 terabytes of memory. So, it is worth what? Rs. 60,000 now? Can you get a brain for Rs. 60,000? I do not know.

The interesting part is it only consumes 10 watts of power, only 10 watts of power. Now, you know your computer requires at least 100 watts to deliver 1 gigaflop. So, if you can scale up to this, you would require a billion watts of power, but the interesting part is not about the speed of computing, I think it is the quality of computing that is distinctly different in the human brain as compared to a computer.

For example, we are predicting that by 2019, we should get an exaflop machine. That is your 10 to the power 3 petaflop machines; so, next milestone. Of course, I do not know how this etymology comes about but every time you increase it thousand-fold, you have a fancy acronym here: zetaflop, yoltaflop, xeraflop; it will keep coming. I think some of you can keep on thinking about putting some more numbers to the right.

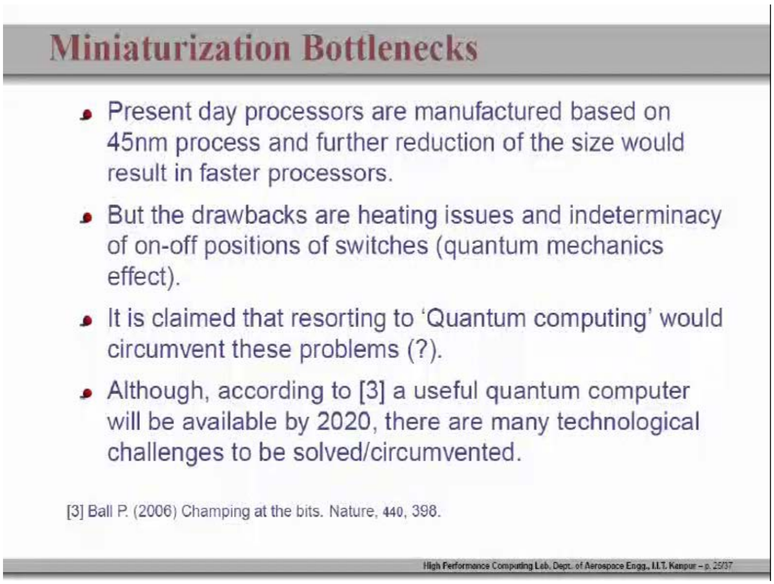
(Refer Slide Time: 51:29)



There it is. If you look at a Roadrunner versus the human brain. I told you that qualitatively, the human brain functions in a different way. Synaptic operations in the brain are actually done through ion channels. These are basically chemical signals, not electrical signals per se. You have got to realize that the electron travels much, much faster than liquid in the ion channel under the same electric field.

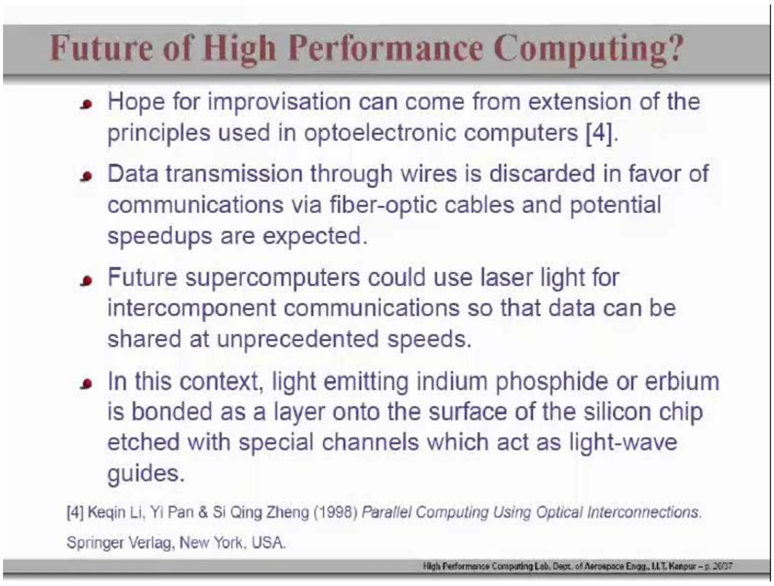
The bottom line is that computers are faster but they are tremendously wasteful. They are much more faster and that is why they could defeat a grandmaster because BlueGene/L came to that stage where it could do much faster calculation, faster than a grandmaster could think of; we know that it defeated a grandmaster.

(Refer Slide Time: 52:29)



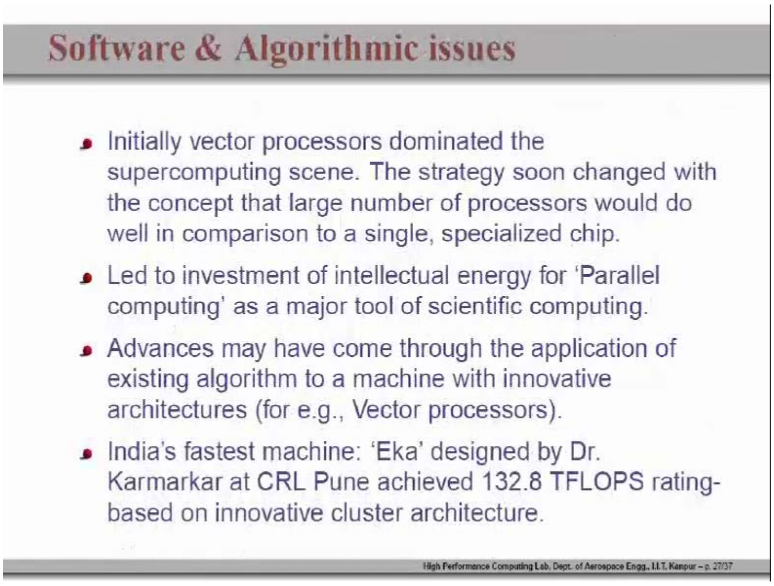
Present-day processors are manufactured on a 45 nanometer scale. I think it has come down to something like 32 or something in a recent document I have seen, but we are getting there virtually, hitting the wall. This hitting the wall will come about because of the heating issues plus this is a dangerous thing lurking in the corner: indeterminacy of ON/OFF positions of switches and this is what is called as quantum mechanics effect. So, we would not even know whether a switch is in an ON state or OFF state. It has been claimed that quantum computing will take care of this. I have put a question mark because I do not believe it is as trivial as that. This paper in Nature once again states that quantum computers should be available by 2020 because there are many, many technical challenges and those have to be circumvented. One of the reasons that heat is generated is we have too many interconnect wires and there is Ohmic heating which creates this heat.

(Refer Slide Time: 53:54)



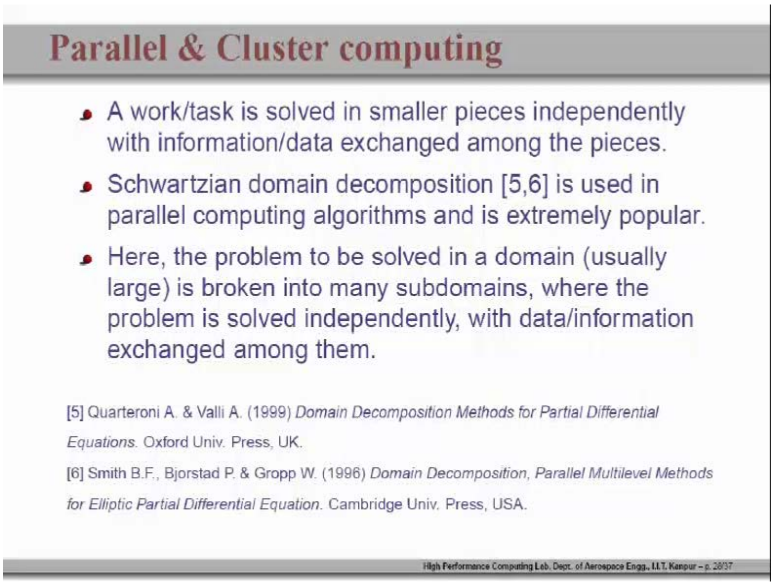
Suppose we replace them by optoelectronic connectors, then we can probably improvise and deliver more computing power. So, many future supercomputers would use laser lights for communicating data streams. People have already started working on materials like indium phosphide and erbium; they are etched on top of the silicon chip. Some references are given here, you can take a look at this if you are interested. I am not an expert, I do not know what is in store really.

(Refer Slide Time: 54:33)



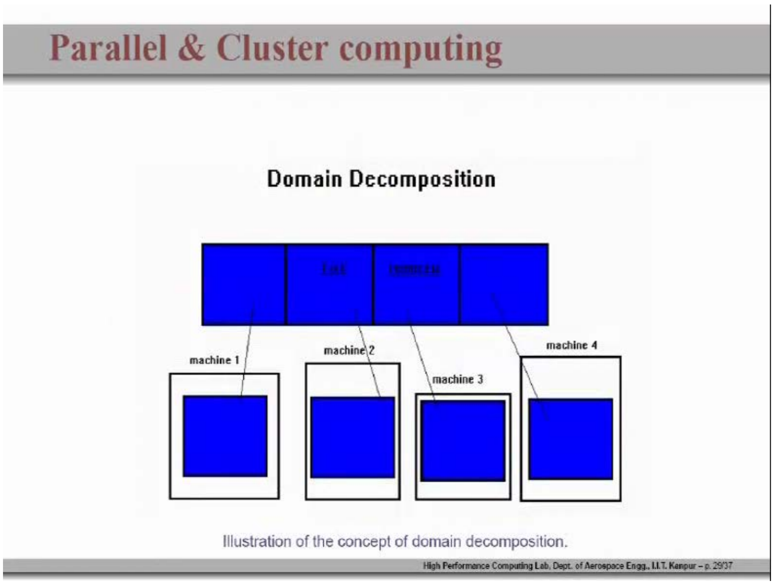
Software and algorithm played its role, I told you. Starting with vector processing, we came to parallel processing. We also have to think of sometimes a new innovative architecture. This was adopted by Dr. Narendra Karmarkar at CRL Pune. That delivered us that EKA computer, which is the fastest Indian computer in the TOP500 list; today also, it is in the eighteenth spot. He actually conceived of an innovative cluster architecture; he sourced all the hardware from the market and came out with this kind of computing power.

(Refer Slide Time: 55:26)



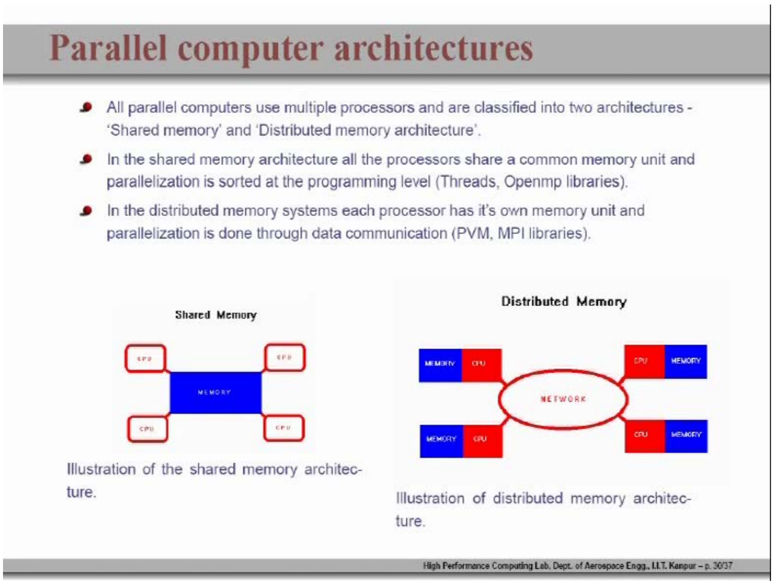
Parallel and cluster computing essentially involves breaking of work or a task into smaller pieces. You do some kind of what is called as domain decomposition and use a technique developed by a mathematician by name Schwartz; so, we call it Schwartzian domain decomposition; that is the backbone of all parallel computing. What we do is we take the domain and break it into sub-domains and the problem is solved independently in each processes. After that activity, individual processors communicate.

(Refer Slide Time: 56:07)



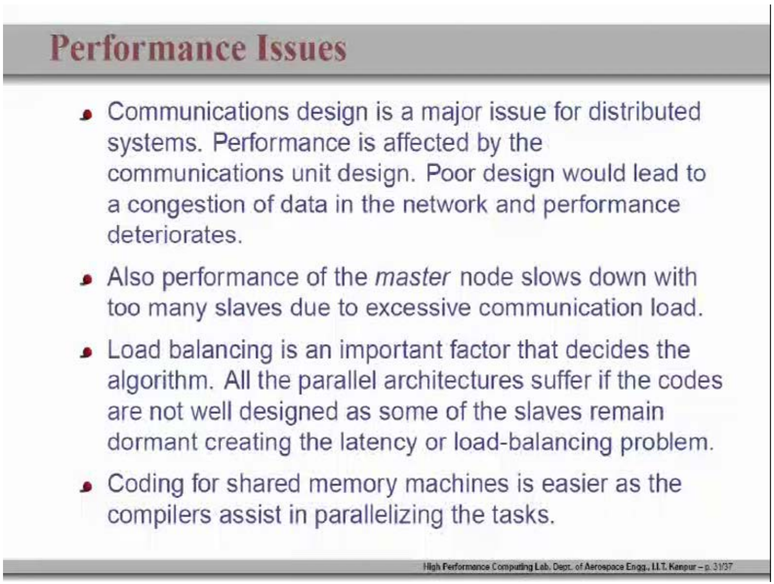
This is basically the thing. We have the main problem and we split it into four. Independently, we work here at this level. After the step is done, we again go back and integrate the data for the whole domain together and from there again, we restart. So, basically, you can realize that there is lot of I/O involved, there is lot of input/output transactions involved.

(Refer Slide Time: 56:32)



What happens is the architecture is also built around in two ways: if you have a shared memory or you have a distributed memory. I am sure all of you know of it and so, it is not really worth explaining to you what they are but they do use some kind of special-purpose software libraries like PVM (parallel virtual machine) and MPIs; without them, it is very, very difficult to work. In fact, even I myself do not do parallel computing; I depend on my brain.

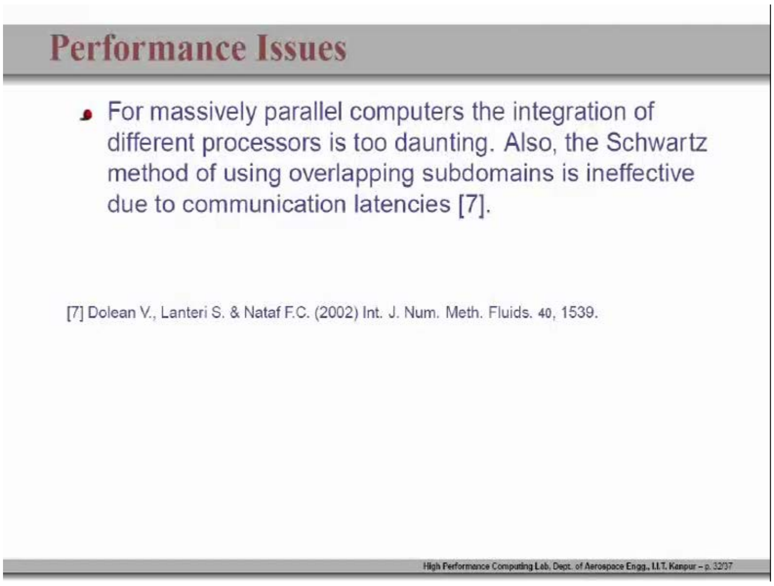
(Refer Slide Time: 57:15)



Performance issues, we can see that communication is a major issue for distributed computing. If we have a poor design, we will have congestion of data and the performance deteriorates. What we usually do is we have something called a master and under the master, we have some slaves; so, this is what happens. Suppose the slave is not doing its part, then of course, the information is not sent back to the master after one step and if all the information is not there, you cannot go to the next step. That is what is the problem of latency.

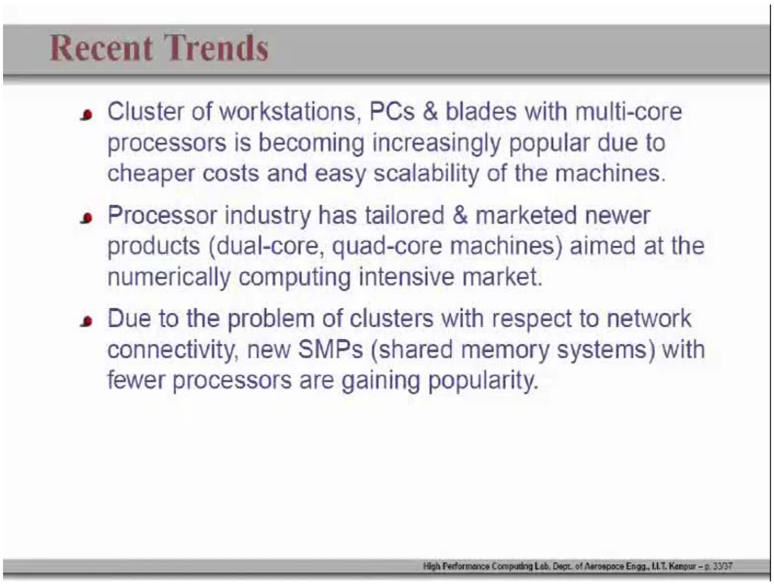
That latency is a major issue that we need to worry about. We need to understand what is called as load balancing. We should distribute the load equally, equitably among all the sub-processes so that they all conclude at the same time and send back the information to the master so that we are ready to go for the next step. You also understand that between shared memory and distributed memory, shared memory will be preferred because then from the slave memory, all the sub-processes are tapping the information and putting them back. So, you do not have to do that much of I/O transactions as you do with a distributed memory.

(Refer Slide Time: 58:44)



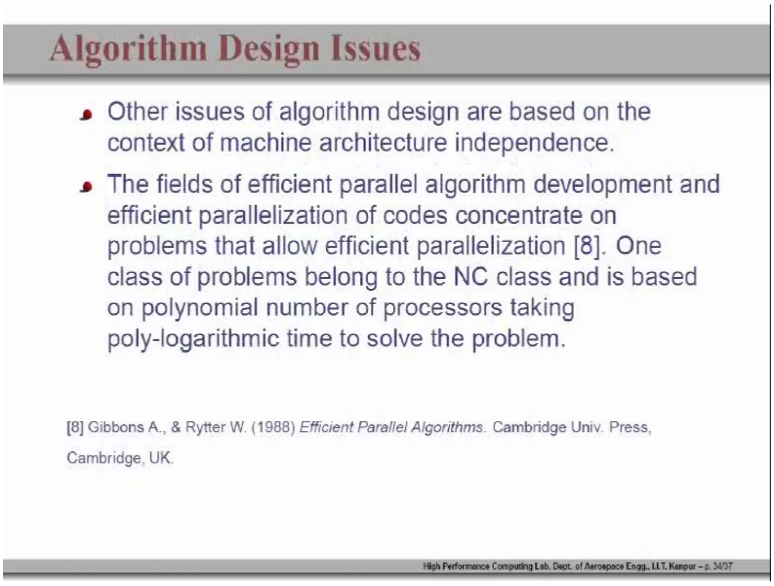
So, this is about massively parallel computers. What happens is although we said that in parallel processing, we will break up the problem into smaller sub-pieces, those pieces are not completely distinct from each other; there will be some kind of overlap because we have to communicate among the processes. This is what we call as overlapping sub-domains and if we have a massive parallel computing going on, this overlapping sub-domains actually creates some kind of problems in communication latencies because what one processor is doing, the conclusion of its task it is supposed to transmit that information near that overlapped region to its neighbors. If that is not done efficiently, that would also create a latency problem.

(Refer Slide Time: 59:47)



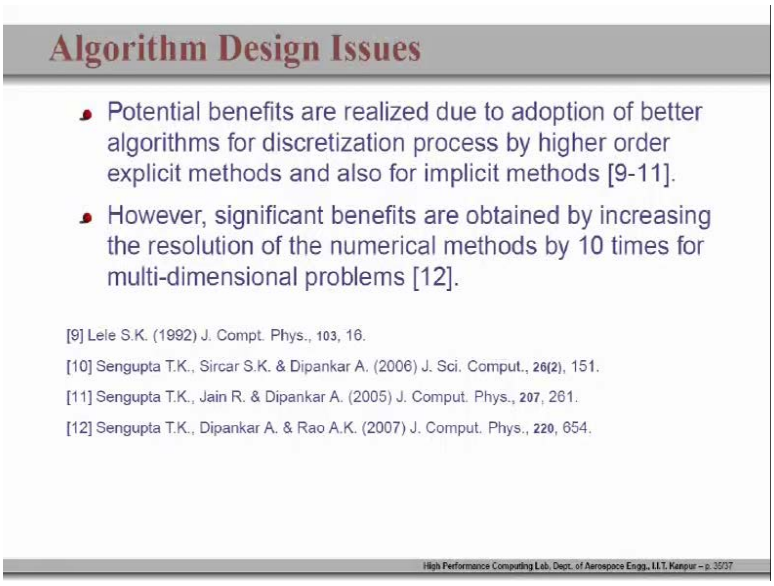
The current trend is of course to go for cluster workstations and then, we have come to this level where we have multi-core machines, which we are seeing more and more and people are again re-looking at shared memory systems.

(Refer Slide Time: 60:08)



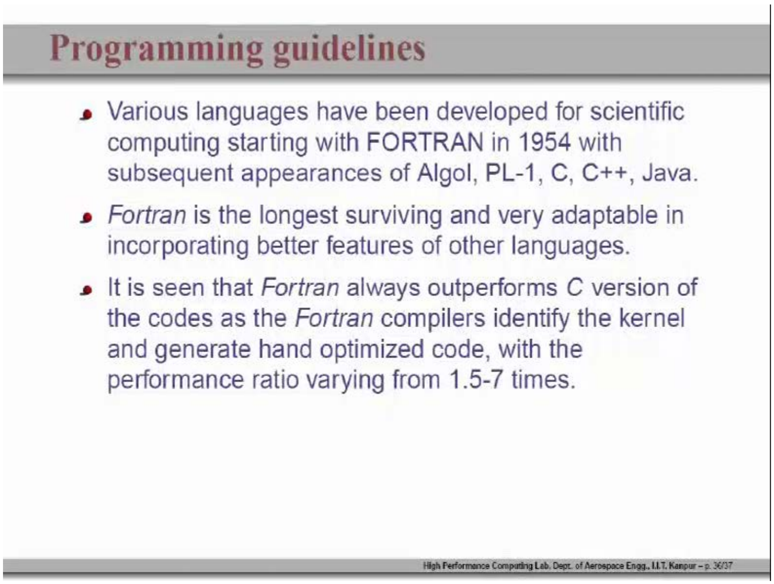
As far as algorithm issues are concerned, this somewhat depends on machine architecture. Parallel algorithm development and efficient parallelizing are the subject of many such researches. I have given here one reference and you can look at some problems which are called nick keeping ((.)) class of problems which are efficiently parallelized but this information may be dated, I am not very sure about that.

(Refer Slide Time: 60:46)



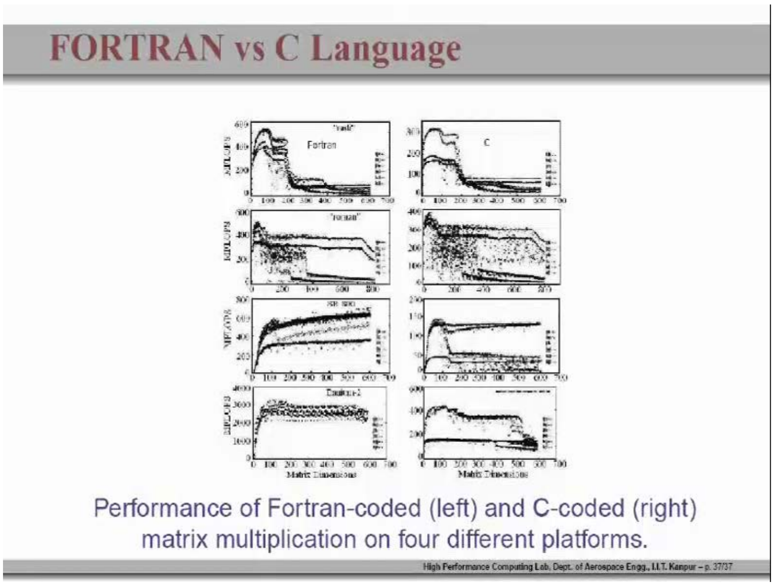
There are other ways where we could contribute. We could develop better methods of calculation itself and we have done a bit by ourselves. These are by some new implicit methods where we get an order of magnitude improvement. In fact, that gives us a computing power improvement of the order of maybe hundred times to of thousand times. So, we end up doing lots of problems on our desktop, which probably people elsewhere use supercomputers to solve.

(Refer Slide Time: 61:27)



As far as programming guidelines are concerned, I told you Fortran came in 1954. Then, we had all kinds of languages coming up: Algol, PL/I, C, C++, Java, but Fortran is the longest surviving and very adaptable language. This adaptability actually is the key to its longevity because it kind of incorporates all the better features of all other languages. It is seen that Fortran actually outperforms C version of codes because what happens in Fortran compiler is you actually can identify the kernel which is doing the main computing and that could be hand optimized which is not possible. I will show you an example here.

(Refer Slide Time: 62:20)



This is from a paper. Simple multiplication is done, matrix multiplication. On this side, you have the number of elements of this matrix size and on this side is the speed. So, on the left hand side, you are seeing Fortran and this is C. You can see the scale is different. This is 600 and this is 300. So, there is a factor difference and what you also notice is that depending on the architecture, every machine has a fair bit of plateau where it performs well but then, it has a performance degradation if the size of the problem becomes bigger than this. In this case, you can see a drastic fall. These are four different machines you can see. In all of them, you can notice that Fortran always outperforms.

Unfortunately, I think it is not being taught here but all the students pick it up; it is very easy. So, when it comes to scientific computing, I am sure any of you taking up any new problem would be well advised to look at Fortran more seriously because if you continue to work in scientific computing, that is a language of preference. I think this is a kind of a general introduction I wanted to give you in this class. From the next class, we will get into the subject proper and we will talk about various aspects of scientific computing.

As I told you, I have a core of students, some of them are here; they are basically working on various aspects of our computing activities. So anytime I give you some assignments, etc., you are most welcome to visit our lab that is in the Aerospace building and you can discuss about your problem. There are your other classmates. I can see quite a few of them. They have already used our lab. So, you can also come and join them. This is where I stop.