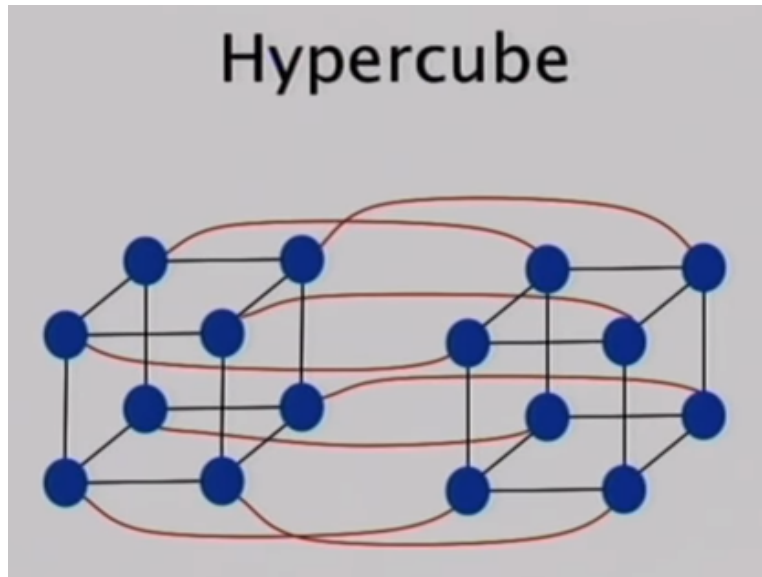


**Parallel Computing**  
**Prof. Subodh Kumar**  
**Department of Computer Science & Engineering**  
**Indian Institute of Technology – Delhi**

**Module No # 01**  
**Lecture No # 04**  
**Parallel Architecture (case studies)**

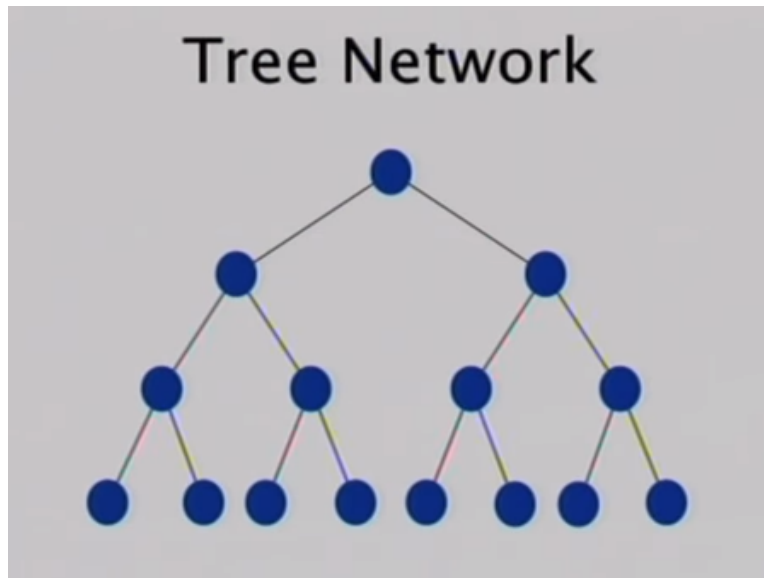
**(Refer Slide Time: 00:26)**



What we talked about at the end of the last class was a connection that we call hypercube is one way to connect some N number of nodes it is not just away to connect the nodes but also away to organize the nodes in the various levels of the also this can be over the network as well but when you actually building this on a chip or on a kind of layered network which is not spread out then we can arrange this way and it helps okay.

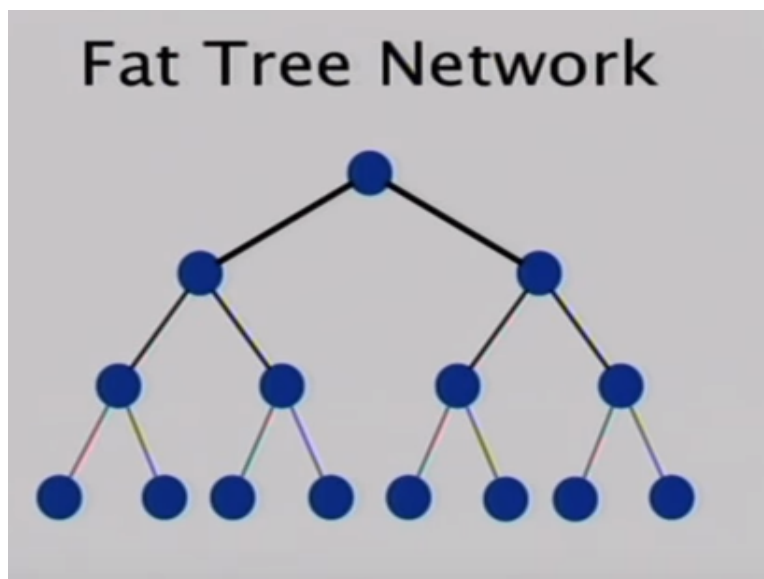
The entire machines have been built which had this interconnect will actually look at those or pass through those basically nobody uses those many more but you can find them in museums.

**(Refer Slide Time: 01:51)**



Another a tree network and some variants of tree networks are fact used in the context of making high performance computers of using off the shelf nodes as well again it has a similar behavior in some ways to the hyper cubes the structure is a tree like structure which mean that if we want to connect or communicate with any node you need to figure out what path in the tree will take you from this node to that right.

**(Refer Slide Time: 02:39)**



And you take the shortest path you go to the common least common ancestor then you go down to that node it is also called the fat three network where pass due network where as you go up the levels of the tree the bandwidth or the physical number of wires actually are going to go up

because more nodes when communicating with each other will end up using the top levels than the once using bottom levels.

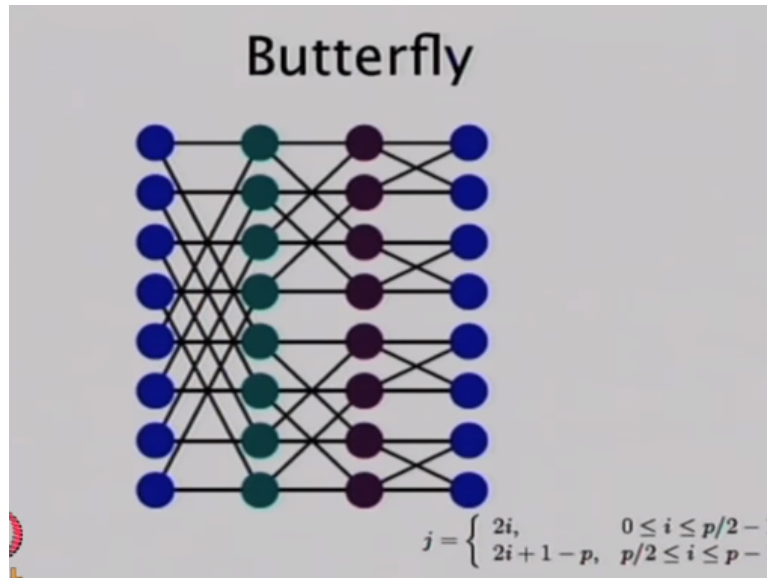
For example if you look to just the bottom levels only two leaves will use those to talk to each other but if you look at the two wires or two buses or whatever you call them two links connecting the roots then many nodes may want to use them right half of the nodes on the left side when they interact with any of the nodes on the right side will have to use those two links.

So many more nodes will use those link so you would want those links to have a higher bandwidth not necessarily this is just talking about how you communicate then about what those notes are for all you care these nodes may be just switches doing nothing communication data from one point to another a lot and there is a lot of hardware area also spent in trying to resolve collisions or give hints about whether to send now or not like whether this link is likely to be busy over the next minute clock.

Collisions would be there in many networks including cube for example it is not just bus because it is you do not have unique path from every node to every other node right here if there are lots of reach the and then they are either going to share the link at the same time or somehow time multiple in which case they have to decide when to go well depends on what your calculation right if two people are asking the arbitrator to send this piece of data on arbitrator is deciding.

Then that circulation its collision is probably the wrong word or slightly overloaded words should probably call it contention.

**(Refer Slide Time: 05:32)**



So there is contention for that link another network which we will come back to interestingly when we talk about sorting is called the butterfly network okay there is a variant of it also called omega network that you will see places slightly different connections but it is essentially in principle exactly the same thing and so you have got the set of nodes and they are connected to each other or two other switches in the layered fashion.

So if you look at the left most column of nodes yeah these are nodes that I want to communicate with each other and let us just say that the non-blue nodes are just communicators switches. And Blue nodes are the nodes that want to talk to each other in the left column is the same as the right column all right so I have got right nodes here we which want to talk to each other any node may want to talk to any other node and this butterfly like connection where the at the first level you have got every node connected to node who is big representation binary representation has the most significant bit different okay.

So the top four guys are going to bottom four and the bottom four guys are going to top four and the next level it is the next bit that is different. So the top two the exchange sometime it is also called shuffle and the bottom two exchanged the top four exchange and he bottom four exchange and then that the next level you look at the least significant bit and then every pair exchanges right.

And so at any of these exchanges you have got basically two controls either it is going to let an incoming data go along the same line right horizontal link or if you are going to switch it to the other length because everything is basically two data's in fact I think I have more detailed picture of this here. So if you look at the link control it basically taking two wires. It is connecting those two wires to and it is usually a single control bit that says either pass both of them through or switch swap them okay.

And so now if I want the zero if node to talk to the first node what would you do it goes straight and then switch right. And similarly if you wanted to have zero at node talk to the seventh node what we do switch all the way and very conveniently just looking at the big patterns of the two nodes you can determine what the control should be and let you what the details are it is relatively straight forward and then typically in an N node network this will have login status all right.

**(Refer Slide Time: 09:03)**

## Current Processor Speed

- ~ 2.x GHz clocks
- > 15 Gflop/core
- > 60 Gflop for Quad-core
- ~ \$1000

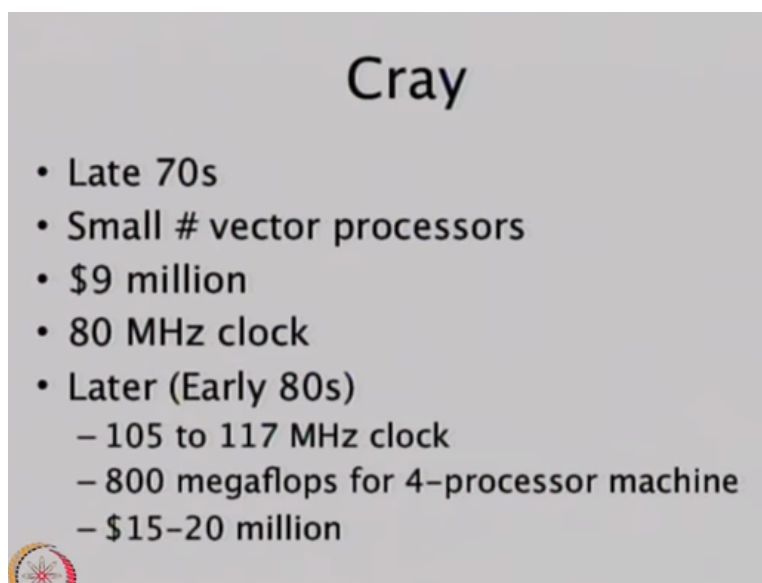
Let us get little bit more into it hypercube does give you very similar guarantees this this has slightly less contention both will have contention but this has less contention. And the logic here is much simpler for the switching so here is an a boiler plate set of number for what you might see in the PC on your desktop ok. And it is going to run at some 2.6, 2.8 may be at even 3 Giga Hertz law.

It is going to have a approximately depending on how you count it 14 to 18 gigaflops. So flops are as you hopefully know a way to measure performance one way to measure performance and it says how many floating point operations can be performed per second it is through put for us it actually it usually much more than 4 or 6 cycles that happens they all of current architecture is some forms superscalar.

For example there is this notion of hyper threading where the same core is able to run two instructions every clock under certain conditions right. So this is the peak how much you can possibly get you probably never get this but again do not get bogged on by the numbers because it is just a range it is just to compare what a desktop computer from today does versus a desktop computer or a super computer from 10 year ago or even 5 years ago did okay.

So just again ballpark numbers you are talking about Quad core 60 gigaflop CPU it you can buy at around 1000 US dollars okay.

**(Refer Slide Time: 11:54)**



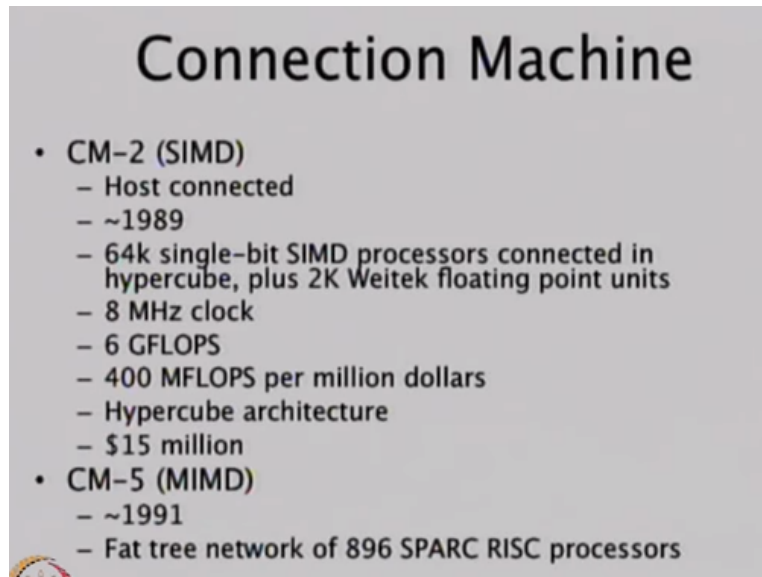
**Cray**

- Late 70s
- Small # vector processors
- \$9 million
- 80 MHz clock
- Later (Early 80s)
  - 105 to 117 MHz clock
  - 800 megaflops for 4-processor machine
  - \$15-20 million

And CRAY which is started in late seventies I started making vector processors was selling a 9 million dollars and only 2 select few they would not sell to anybody. And it ran at 80 megahertz clock at the time okay and they in by the time eighties came around they had 210 approximately megahertz clock and it was giving around 800 megaflops for a 4 processors machine okay.

These are vector processors they can do much more from their day processors point of view not today's processors point. And so I went up to about 15 million dollars so for 15 million dollars you could buy something that could run at 800 megaflops okay.

**(Refer Slide Time: 13:00)**



## Connection Machine

- **CM-2 (SIMD)**
  - Host connected
  - ~1989
  - 64k single-bit SIMD processors connected in hypercube, plus 2K Weitek floating point units
  - 8 MHz clock
  - 6 GFLOPS
  - 400 MFLOPS per million dollars
  - Hypercube architecture
  - \$15 million
- **CM-5 (MIMD)**
  - ~1991
  - Fat tree network of 896 SPARC RISC processors

And connection machine which had there is a company that was found around connection machine came out of MIT and they had sequence of machines not all of them are actually sold. So here you see CM2 and CM3 so they had number CM1, 2, 3, 4, 5. 2 and 5 were the once they were sold and so the time frame now we are talking about is 1989 for a CM2 and approximately for early nineties CM5 okay.

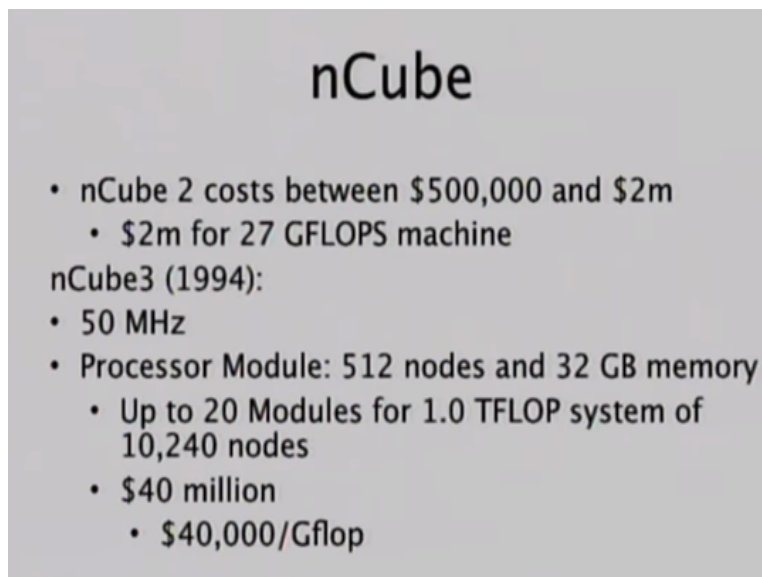
CM2 was one of the early proponents of the SIMD technology meaning that it was trying to become less expensive by getting rid of all this different controls taking the entire machine and having it run the same instruction at a given block. So it had 64,000 SIMD processors okay it went into craze where very low granular large powerful processors small numbers of them. And CM2 was large number of very small processors 1 bit processors which if you have to add them all of them adding 1 bit at a time.

Because it was SIMD also it also could take only a small clock a slower clock at the time but because they also had large number of processors they were able to go beyond what crazy numbers were so they are talking about 6 gigaflops at the time of this and they broke the gigaflop

barrier. And the cost was a 15 million dollars so if you just talk in terms of how many flops for how many dollars comes to approximately 400 mega flops for every millions dollars.

And CM5 which used factory and CM2 use the hypercube network okay the one that like structure with log and steps up to log and steps. CM5 used the tree structure and was made of at the time commodity processors the sun SPARC processors and it went to not large number of small processors

**(Refer Slide Time: 15:37)**



nCube

- nCube 2 costs between \$500,000 and \$2m
  - \$2m for 27 GFLOPS machine

nCube3 (1994):

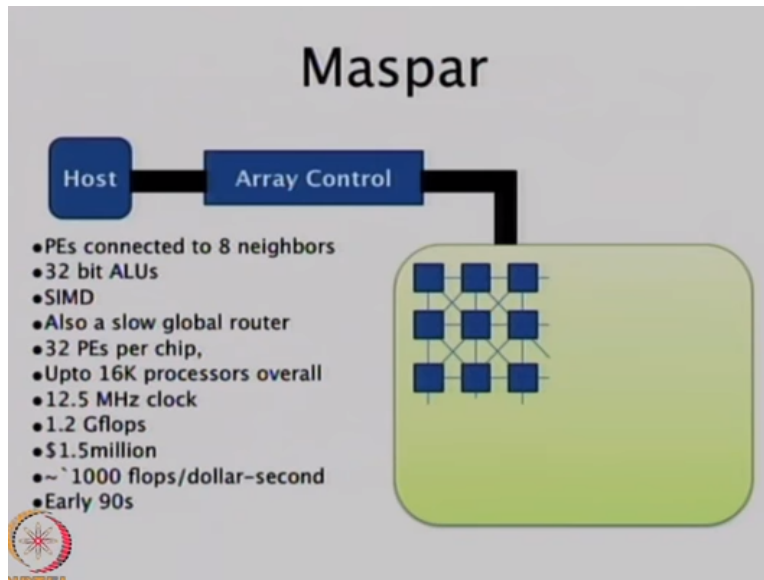
- 50 MHz
- Processor Module: 512 nodes and 32 GB memory
  - Up to 20 Modules for 1.0 TFLOP system of 10,240 nodes
- \$40 million
  - \$40,000/Gflop

But again small number of large processors in parallel around the same time was starting the server machine called N cube which has the name suggest and connect how as a hyper cube precisely. And N cube which was again 90 second was available at the time for just small number of million 1 million to 2 million dollars and again this came just at the CM5 time. So at that time talking about gigaflops the CM2 already had broken the gigaflop again.

An N cube 3 which came at in mid-nineties frame they turned down the clock little bit ahhh and they were able to get to 40,000 dollars per gigaflops. So costs are coming down per flop okay and they were also first to break the teraflop barrier using ahhh 10,000 processors.

**(Refer Slide Time: 16:35)**





Somewhat more on the probably at the time commodity sized machine which is lots of people bought these well not so people does not mean you can me for our homes. But large research labs for their labs okay and some universities also had and this machine was called MASPAR which is using SIMD processors connected using the mesh 2D grid with toroidal connection at the end.

And it also had connection diagonal so you could be only talk to you left right or top bottom levels but also north east and south east and it also fact it had instructions that is it send this south east. So everybody was sending their data south east they did have large number of processors 16,000 overall and these were all not one bit processors okay. Actually they had the whole 32 bit logic unit but it turned of flops and Gigaflops was at 1 Giga flop for 1 million dollars approximately.

And then Cray which is probably so we started with Cray and we are kind of ending with Cray because it is only of those companies that still is doing business okay and queue is gone connection machine is gone power is gone but Cray kept evolving kept changing what they were doing they throw away their entire architecture and went with these days more commoditized design and so by end of the century post 1995..

They had started selling up to this was their highest end machine at that time which cost 40 million dollars for 32 processor machine with large amount of memory at that time 8 gigabyte of

shared memory across the 23 processors. Again it is at the gigaflop range not the teraflop range so 57.6 was the official peak the gigaflop numbers yes it is reasonably high numbers and again Cray until this time until about 1995 or 95 was still selling to just governments it was not selling to would not sell to china.

For example it did sell to India but it would post to engineers so that you do not use it for non-authorized usage.

**(Refer Slide Time: 19:40)**

## Roadrunner (2008)

- \$133 million
- Multi-stage InfiniBand interconnect
  - Infiniband: 2-level fat-tree, each leaf switch has 180 down links and 96 up links (18 such CUs), 12 up links from each CU connected each of the 2nd level switches
- cluster
- 122400 cores
  - 6912 dual-core Opterons
  - 12960 power XCell eDP: 116640 cores
- peak **1.7 PetaFlops**

Now we jump a little bit 2008 Cray was not assigned there were vector processors meaning that processor would run many things not necessarily vector of length for but length 4 kind of become standard later is a vector with much longer length . So there is some SIMD flavor to it because it was doing something on that vector but it was not officially single.

Vector is not necessarily collection of SIMD you have a vector of data in which you are doing something you are running one instruction on that vector of data and that instruction is may adding data together so that is not SIMD. This was road runner was the top so there is this stuff 500 high performance machines which is based on some scientific computation bench marks that is published at the end of every year in November at supercomputing.

And 2008 the runner was road runner 2008 winner was road runner which was now going into petaflops I think so we are not just making teraflop the audio is running but peta flop area first machine to do so. And 122,000 crores so there is lots of course and these are again now we are talking as this you will see that 2008, 2009, 2010 all the top machine are going to be using commodity hardware .

You buy things from Intel or AMD or Nvidia and you just put them together 133 millions so not just you see has not necessarily come down at the high end these are the highest end machines but what you can get for that cost is been raising at the rate at which the things are getting faster. So it uses this infiniband interconnect we talked about multistage networks so it has several stages of interconnect some is fat tree some is just basic BUS and you just local processors you can talk too much faster and somewhat further away processors to which you have to spend some more time talking to and so on and hence so forth.

So there is a significant bit of middleware layer software layer that is required to make sure that can in fact may send data or arrange the data be sent only local most of the time that two processors that need to communicate or to processes that need to communicate together in some area where they need to talk to each other fast and so on hence so forth. So there is lot of hidden cost in that 133 million which is not all hardware cost okay.

But from this point onwards basically see the clusters off the shelf nodes by this CPU nodes from Intel you buy some network typically there would be some designing of interconnect because one network for the entire tons of thousands of processors not of nodes now is not going to do the job. So there is some hardware cost as well as the design cost spent in the interconnect but actual compute engines are just off the shelf.

Yes and no right so it is not just buying infinity bands or gigabyte it just not just connecting them because they are actually designing faster interconnects okay. one of Google's patent is on how to make these fast interconnects for there because Google only buys processors of the shelf but they have their own interconnect that makes them talk faster of course it is very expensive because it designed only for that purpose and very few people can buy it.

But if that were also come out of ties then yes one day you could just build super computer in your kitchen in top number no it is not param it is this computer at TATA at Pune. I do not recall what its position is its is not eight it is closer to 25 range but not in the top 20.

**(Refer Slide Time: 25:19)**

## Tianhe-1 (2010)

- 3584 blades with two nodes/blade:
  - Node: TWO Xeon X5670 CPUs + ONE Nvidia M2050 GPU
  - 7168 GPUs, and 14,336 (6-core ) CPUs
- 138 cabinets
  - 112 compute, 12 storage, 6 network, 8 I/O]
- Cabinet = 32 blades + 16-port switch board
  - Proprietary high-speed interconnect: Arch = 160 Gbps
    - Twice the bandwidth of InfiniBand
  - 262 Terabytes total memory
  - 2 Petabytes disk (*Lustre clustered file system*)
- \$88 million to build, \$20M/year operating cost
- **Peak 2.5 petaFLOPS**

And this is the now currently doing champion at 2.5 data flops this is less costly. And one of the reason it is less costly is that using these GPU's it is gone up from breaking the peta flop barrier to reaching the peta flops cost to build on only 88 million and they also coat their running cost it is also much less as it turns out than the other one's just 20 million and most of it is power and cooling power.

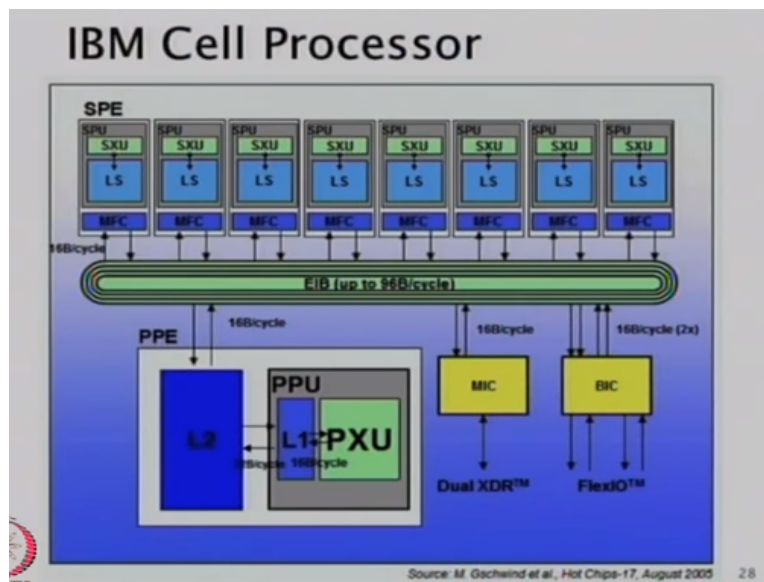
They also use a proprietary high speed internet they give a name which I have not written on it can go up to 160 gigabytes per second. Which is if you think about it not that much more than what you get off the shelf it is just twice infinity by infinity and which you can buy from the market at not that later price because lots of people are now are using high performance contribution.

It has to be counted in the end right but there are lots of latency hiding techniques the people use and we will talk about them later on in the course also you basically use pipeline techniques

okay. I think we are at 30 minutes we are going to take a short break I am not quite done with this section yet.

So we will spend few minutes on the section and then move on to a program because we are little late we are little hope that we would start that a little earlier to finish your first assignment and for that I need to get the programming first so we will do programming and then come back to talking about the models of computation and design techniques okay all right. let us continue we the machines that I was going to talk about actually.

**(Refer Slide Time: 27:58)**



I have talked about the couple things that remain in this section was some special purpose procession units that people are building into their machines. So I told you that Tiahana 1 is using these Nvidia GPU's but I did not tell you that roadrunner was using cell processor from IBM which is not question yes Sony play station itself but road runner also used there.

So cells are not just for gamine although they were designed for it originally for some reason that part of it is the design of the architecture itself but major part of it is just software support that needed to go with it cell processor is not quiet taken up it is a reasonable architecture it has good performance numbers wise but it is not that well used in the high performance computation country very quickly what the layout is it has number of SIMD units.

Those blocks are you see that are called as P U the top row of 8 blocks are simply units each okay. There are 8 SIMD engines so it what does it make it it makes a SIMD machine with parts each SIMD is different set of instructions but a given SIMD is given instructions on a number of execution units at the same time okay. In addition to it 8 SIMD engines it also have 1 generic power processors okay.

To run operating system to run CPU like things although it is in designed to simply be the machine on your desktop it plugs in to an another machine meaning that you probably another operating system but this CPU will also run an operating system and you can run the Linux on it example on it has it is a heavy duty CPU right. So it does everything that any other CPU would do it has a full set of instruction that you would find it in any CPU.

What else it adds is its interconnect between these power PC's and the interconnect which is has a intelligence it is not just connecting it is bus like so you basically say you send it to see SPU number 3 and you put the data on the bus. It also has intelligence and I will tell you just in a bit what that intelligence is let us look at the top rows. The blocks has something called LS4 so this a memory that is shared but any other SPU's does not even know that there is this result.

It turns out that in this design SPU's can it is P yes there are execution engines within SPU and they all shared this local store. So the instruction set of the SPU's only has load and store from this memory it does not have any notion of virtual memory it does not have any notion of e caches and levels of caches and so on and so forth. So with the little bit of intelligence between the local store and the SPU's or SXE what they call SXU or SPU's.

And a bit more about the logic incised this bus they basically simulate a virtual memory like situation. They do not have real translation tables but what happens is that there is a virtual address and if that address maps to local area it is going to get fetch from local store. Otherwise what is going to happen is that EIB is going to issue a request to the main memory it is going to get the data into the local store and then from the local store it is going to get service to the execution unit.

It works very much like a cache EIV is connected to the only the Power PC is able to do that right it is going to create the DNA engine in there right so the power PC is going to suck this up and the DNA engine will get the data from where to copy to where okay and then that DNA engine which is inbuilt at the EIB level is going to take care of transferring the data okay. The two things that DNA engines is connected to all right.

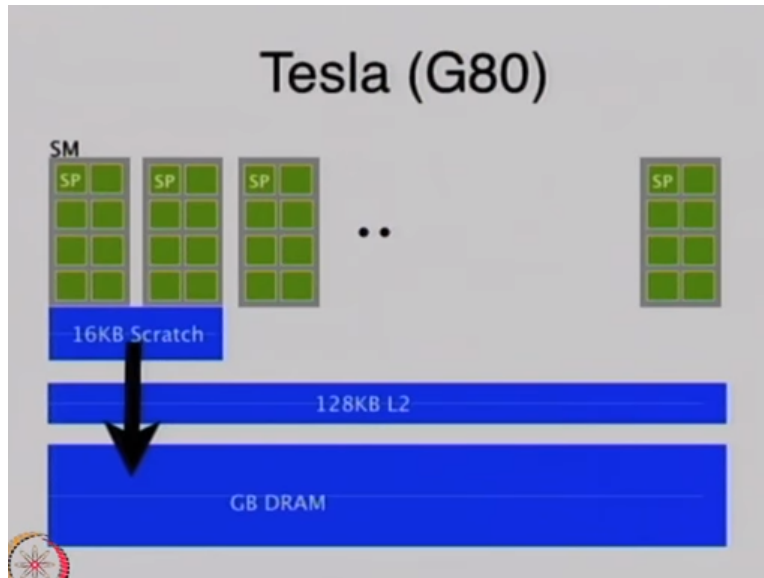
MIC is the memory interface control okay that is where it is going to get the data from okay BIC is connected to all the other things that it many to connect to network bus or jus it is an output input output interface. So the main thing that you may want to take from here is that it has lots of processes which work in the MMD fashion inside each processor there are other sets of processors is actually a multi processors which work in CMB fashion and they have a local store only which so as far as memory operations and circumstances it directly go to the local store and come from the local store okay.

It depends on your contacts right but you are actually talking about course they would not count the individual execution units they count the how may CMD engines they have. It is basically running the OS and the design intent is that not everything will fit the CMD more. If you do not fit CMD more then you run that part of you code on this power PC okay and the rest gets run over there.

It is a cash L2 is connected to the global memory local storage does not actually communicate with L2 that data that goes to there is through that MIC interface L2 is not caching for them with some intelligence somebody is taking care of figuring out where to get it from and where to put it as far as the SPU engines execution units are concerned they do not know anything about it they just know they have to read something from this address and they have to write something to this address from the local store okay.

And somebody else is going to block them reading that address because that address does not have the data supposed to have.

**(Refer Slide Time: 37:40)**



And this is GPU alright this is the tesla architecture which came out in 2007 with from NVidia named G80 but you see here is a sequence of a essence okay SR a speed of 1 SM working in a CMD fashion and different SM's are working in an memory fashion listen the overall control design okay. SP is the execution unit so it is again it is hard to call it a single processor because does not have the entire control there is a shared control and as there are so many execution units once CMD in SM.

SP's are working together in a CMD fashion so SM will have one control for the entire set of SP's in this design 2 SM's together shared a local memory which they called shared memory but it also was able to directly fetch data from the main memory which you see here is the DRAM ok the Giga bytes a couple of Giga bytes. As a matter of fact in the G80 there was a L2 cache but as this picture shows it was being by passed not actually being used it was being used in graphics.

So this was not for just computation it also for graphics and L2 cache it was designed for graphics where to cache was designed in that context. But they were not quite able to make it properly in the SIMD in the GPU computation context and so they just bypassed it they are not using it. So locality is the property of your program of your code okay your code still probably still have the locality but there is no cache here meaning that the CPU decides or the SP is decides whether to rate from the flash or to read from the DRAM okay.



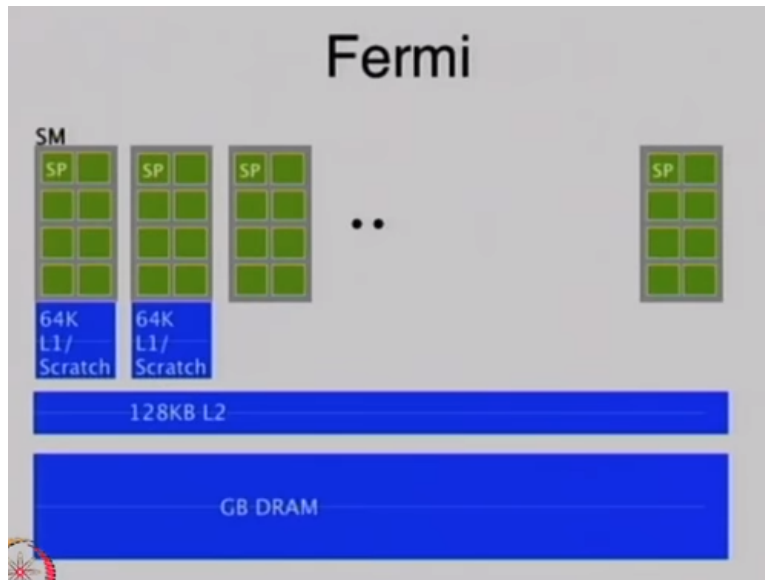
So they are separate instruction in the instruction set or reading from the scratch pad and reading from the DRAM. So the cache as to be managed in software okay and there is lot of research on whether scratch pad memory is better than cache because the cache pack memory does not have all of the paraphernalia that you need for cache right. For caching you need to have associative lookups so that you can figure out whether or not it is in cache you have to keep all of that those tables are around so that you can know what is in cache what is not play what is active.

There is cache coherence information that needs to be maintained all of that it is not done at all for scratch pad. So for the same area and power profile you can get a much bigger scratch pack as a result of which you can cache more although it is not a cache but you can store locally much more data as a result of which if you little store about how to store data it will be faster than a oblivious cache which does not know what kind of access patent you have is using some recent used policy hard coded.

And so there is always tension between which is better well of course some way turning here also otherwise who is going to resource allocation is going to launch processes who is going to decide. So there is an OS running here and CPU as its own OS this is a proprietary OS so you really do not know what they does but there are protocols established what they are CPU as to tell this OS for its to configure its local state okay.

So CPU sense it here is the program so sense it is there is a data address for DMA something from the CPU main memory to the GPU's main memory which is also often called the global memory in the context of CUDA. GPU's memory is different this is not the CPU's main menu and there will be a PCI express connection from the bridge did I not have I will may be show you the diagram later of the architecture inside a PC where you also see what is connected to what.

**(Refer Slide Time: 44:30)**



It has been removed in this next architecture it does make somewhat difficult to go and that was one of the criticism of it and so the architecture that came out the last year 2010 which was code named Fermi as a very similar design still has SP's okay more of them now although I have not multiplied them in the diagram. And they are SM's holding this SP's the change is that are included are that each SM's get own local storage scratchpad.

So it not shared by 2 SM's and this scratch pad instead of 16 kilobytes is 64 kilobytes and it can be configures as a cache. So if you do not want to deal with managing your memory locality then you just let it be run as a cache what is more it can be partitioned into 32 kilobytes of cache and 32 kilobytes of scratch pad for example so there are 2 or 3 configuration you can use. So you can get maybe a best of both words you can have an entire scratch pad I am sure that you can have entire cache I think some scratch pad that is necessary.

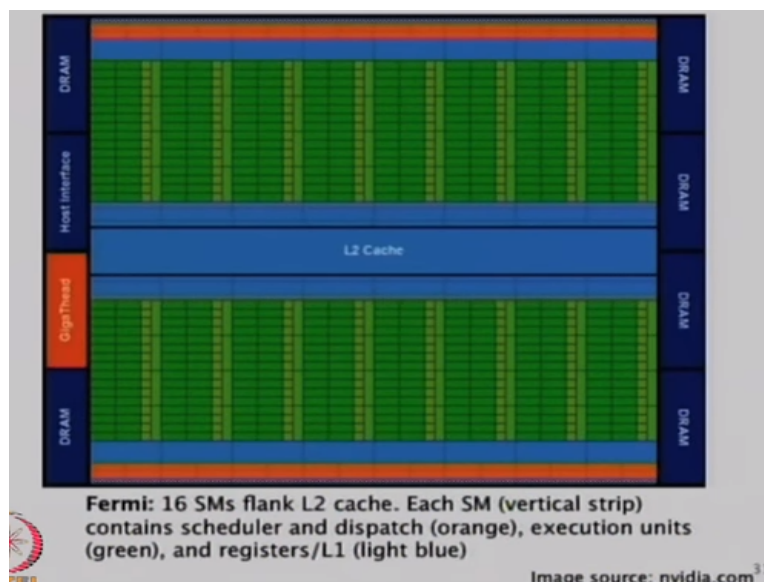
But you can look up the details of the architecture later on and then the L2 is sharing it is L2 and the DRAM together the global memory think of that the shared memory shared across the memory units okay and this scratch pad is still shared but still only across the CMD processor one of the memory processors so all the CMD units. Precisely right and the it is to smart it is 128 kilobyte cache is old time if you look at Intel caches result megabytes.

So it is a small cache but then there are these trade-offs to be made right whether they could have it was a designed decision to rather spend that in the L1 right there is so much wattage to be allowed to consume there is so much area you are allowed to consume and you have got to trade off whether to put that in L1 or put that in L2 whether to give more SP's or to give more memory those are the design decision that were made by NVidia.

So what happens is that you still address what has not changed is the instruction set architecture where you are saying this address is going to scratchpad this address is going to the main memory the global memory. The global memory if it is in the cash section will not actually go to the global memory this scratch goes to the local memory. It is a completely implementation problem meaning that there is not design that is involved in why they were not able to do it.

The real explanation is that it was not designed originally to do computation and it was designed to do graphics and they were not simply not able to manage the right timings and the right generic cache coherence protocol that were needed to get it done okay. For me on the other hand right from day 1 was being designed for computation along with graphics so they make sure that all those the new at the short comings were and when and how to fix them it is just that they either did not have the time or the money or the motivation to do that at that time okay.

**(Refer Slide Time: 49:37)**



This is this shows that in the framework or the time line we had been looking at so far actually it does not quite you can buy the Fermi card I think it is in the range of 1500 to 2000 dollars okay. I am talking about the top of the line yeah you can get the GPU's which will not have so this is the full GPU's and what I will explain this diagram in just minute and you do get stripped down version of it with the fewer SM's essentially with less global memory and you can get it for 400 dollars okay.

Reasonably good fast GPU but this GPU you can get this now for a couple of 1000 dollars as a peak performance in teraflops okay. So million dollars per mega flops versus about a 1000 dollars for the teraflops that is the state of the so let us look at this diagram what you see here as in fact it is I think says there at the bottom the SM's each vertical column of 2 green boxes of 2 columns of green that you see together that is 16 SP's 16 SIMD units make up 1 SM okay.

They will be running on SIMD and then there are 16 of these SM's spread on the machines okay. and they are spread around this central area which is the L2 cache and a inside the green blocks the green SP's and individual SIMD engines you see those controls units being shown in pale yellow and the shade memory at the bottom being shown in blue okay and then there is some control units to can control the launching of these execution of these or what equivalent to processes is from the CPU domain.

They have one other very important characteristics which is also the part of the design is that it implements threads in hardware. So that means is that it is going to maintain context for thousands of threads at the same time okay. So you do not pay the price for switching context from thread to thread for that means you can break your program down into really small units each being a thread.

And all of these threads are simply swarmed on to the SM's only as 16 things can do at given clock right actually it is 32 because it is two things but so it has only 32 things it can do every clock but you can give it a 1000 threads which together can hide latency one thread goes to the memory and it has to wait for hundreds of clocks before the data can come back from memory other threads can keep executed okay.

And because there is no context switch penalty it is as if your entire program is running all the time okay without any stop and that is one reason you actually see an excellent real performance as long as you have a program that does lots of computation which can happen in parallel depends on the other than GPU or not I mean you cannot break and out slow okay. So this is about that and it seems that we have also reached the other end of other half an hour so we will again stop this.