High Performance Computing for Scientists and Engineers

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

Module – 04
GPU Computing
Lecture – 30
Introduction to GPGPU and CUDA

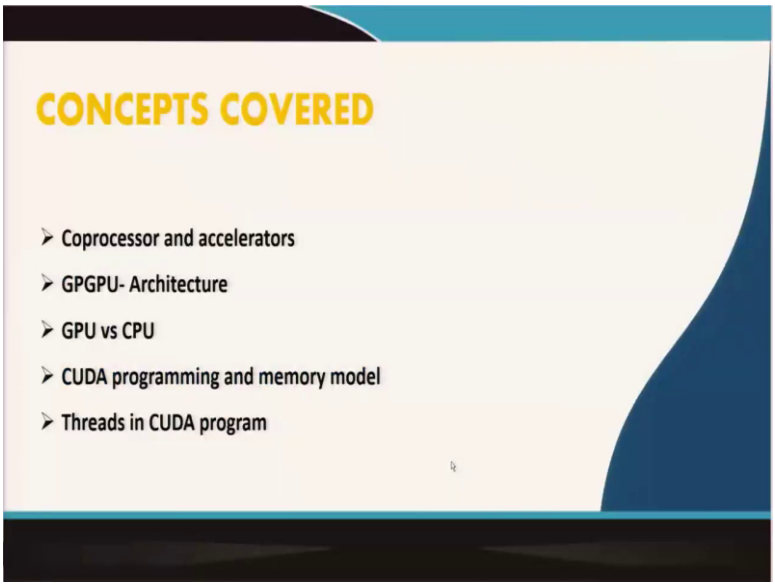
Hello, we are discussing contents in the course High Performance Computing for Scientists and Engineers, in the last few weeks we have discussed the fundamentals of parallel computing and then we looked into open MP and MPI programming using respectively shared memory and distributed memory architecture.

Now we will start the last module of this course which is GPU Computing. GPUs are very different from the architectures we were discussing previously which are multi computer or multiprocessor systems, but GPUs are as the name suggests graphics processing units and now the first question that will arise is, how we can use graphics processing units for doing scientific computing. Therefore, a modified version of GPU which is called General Purpose Graphics Processing Unit or GPGPUs comes into picture.

In this particular lecture and in the subsequent lectures we will look into GPGPUs, their architecture, their functionality, how GPGPUs can be deployed for running scientific computing programs and what are the advantages of GPGPUs in the context of high performance computing, we will look into this and we will also go through one of the very popular API for GPGPU programming which is known as CUDA.

So, this lecture and the subsequent lectures will try to give you some introduction in GPGPUs and CUDA.

(Refer Slide Time: 02:09)

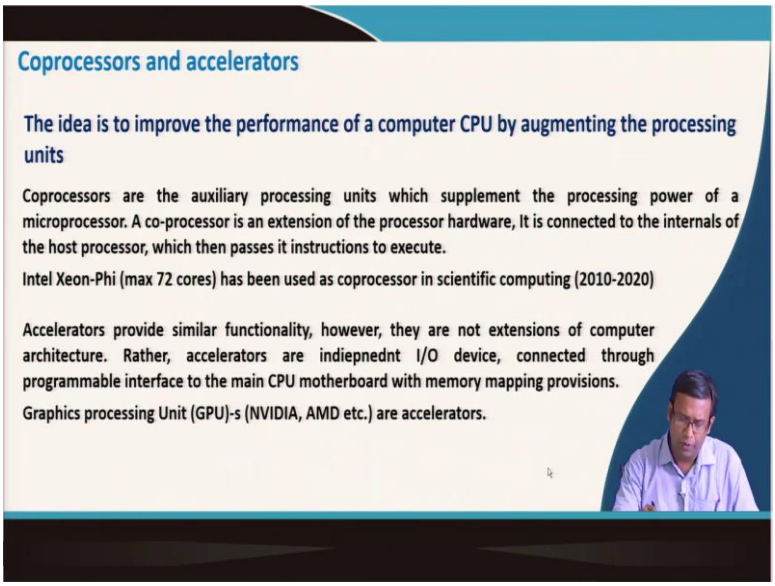


What we look through over these lectures are co-processors and accelerators and this GPGPU is typically an accelerator for optimizing and getting efficient solutions of large computing problems. Look, what are coprocessors and accelerators, then we will look into GPGPU architecture, we will see what is the difference between a GPU and CPU in terms of the hardware as well as in terms of programming and implementations of HPC jobs on them.

Though, we will use this term GPU mostly, but what we are discussing here is the general purposes of GPU or scientific computing use of GPU which are basically GPGPUs. Then we look into CUDA programming and the memory model associated within that and then we will see how threads can be launched in a CUDA program.

So, we will not go into CUDA programming details in this lecture, in the subsequent lectures we will come about CUDA programming details and I will demonstrate to you CUDA codes, but here this is a background and introduction to CUDA programming that I will try to give you.

(Refer Slide Time: 03:31)



So, when we talk about coprocessor or accelerator the idea is that a computer CPU has limited power in terms of computing capacity, it has a particular processor speed, it has its own RAM bandwidth and latency and the caches registers combining everything, there is a limitation beyond which a computer CPU cannot process numbers, it cannot be more faster than this and this though we know that computer speeds are increasing day by day.

However, there are certain constraints like manufacturing of the CPUs putting the right number of transistors which can give high throughput in the CPUs that is the constraint.

The next constraint is the space issue, you cannot have a desktop computer which will take too much foot space. The third and most vital constraint is the thermal issue or the heating issue, if you have many transistors crunching numbers, doing arithmetic logical operations they will generate sufficient heat and there will be a requirement of heat dissipation through that computer which is also a very challenging task.

Therefore, instead of only restricting our views to increase the power of a single CPU, now we are not talking about multi CPU system we are talking about a single CPU system maximum multi core CPU system, but instead of putting efforts in improving the hardware of the CPU the idea is that, if some way more processors or something called accelerators can be augmented to it ,which can allow the CPU to use more processing units.

So, the idea is to improve the performance of a single computer CPU by augmenting processing units in that. The first idea that comes in our mind probably from the discussions we already had in the beginning of this course is something like a vector processor or a processor array.

There is a CPU which offshoots its jobs to multiple smaller CPUs connected with that and a similar idea can be coprocessors, they are auxiliary processing units which supplement the processing power of a microprocessor.

So, there is a processor and coprocessors are attached to them and there can be a number of coprocessors which are attached to them. These also have processing capabilities, but their architecture is simpler and their operation is much less restricted, a CPU has to guide them to operate.

However, it is connected to the internals of the host processor, they are using same memory space, it is only some of the instructions the CPU will offload to the coprocessors and one of the very popular coprocessors was Intel Xeon-Phi which could give maximum of 72 cores along with the Intel Xeon processor.

So, Intel Xeon-Phi is augmented with the Intel Xeon processor and higher speed was obtained and in the previous decade from 2010 to 2020 there are several versions of Intel Xeon-Phi which are implemented and give faster computing. So, these are coprocessors. However, Intel has discontinued Xeon Phis, but there are some core processors still available in the market.

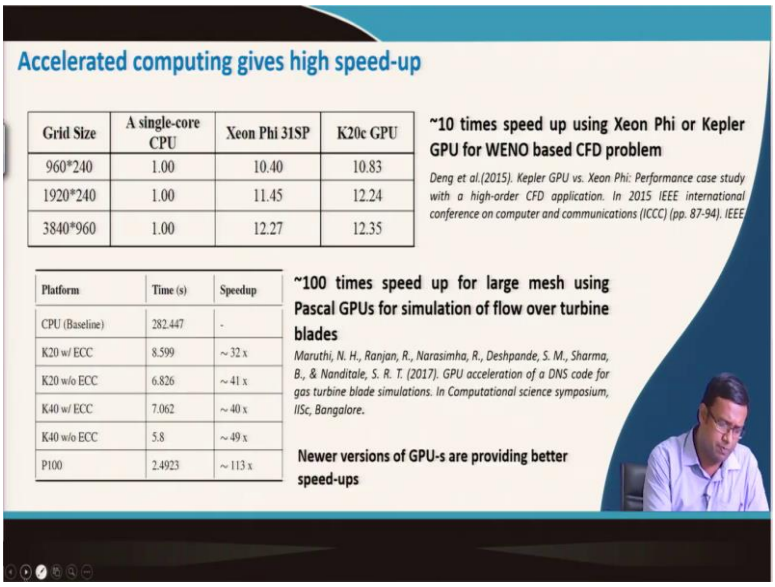
The other idea is using an accelerator, accelerators provide similar functionality as co-processor that there is a CPU, there are multiple computing cores in an accelerator which is connected with the computer CPU, but these accelerator is an accessory to the main computer, it is not part of the computer it is a like a graphics card what we will discuss is an accelerator.

So, it is a card or it is an accessory which is connected through PCI bus connected to the motherboard of the computer, but it is not sharing the same ram or same computers basic architecture if this is connected to the motherboard, but not sharing the same motherboard. It is like an independent IO device and these are also sharing different RAMs. Accelerator's memory and CPUs memory physically are also different memory and therefore, there is a requirement of memory mapping also.

Most popular accelerators are as we are discussing GPUs are graphics processing units and the accelerator market is almost fully occupied by these two companies NVIDIA and AMD.

There are some other companies who manufacture accelerators, but accelerators also provide the same functionality that the CPU can offload some of the jobs to the accelerators and they can process the instructions on some of the data which is mapped from the CPU data to the accelerator's memory space.

(Refer Slide Time: 09:02)



These are some example of accelerated computing we can see that for a WENO based CFD problem Xeon Phi accelerator and the K 20 Kepler GPU their performance is given, where single cores computing time computing speed is scaled as 1, Xeon-Phi gives 10 times faster processing and GPU gives 10.8 times faster processing, as we increase the size of the problem the performance of the accelerator and coprocessor increases, but they are comparable.

So, what we can see is that they in some way can substitute use of multiple processors or multiple computers, they are like accessories. So, if we think of getting 10 time speed up or 12 time speed up, we definitely need more than 12 CPUs if we think of ordinary parallel computing using CPUs, but here we are using only one CPU and then card or co-processor associated with the CPU who is which has multiple cores definitely more than 10 -15 cores.

In Xeon Phi we have discussed that they give around 72 cores and accelerators they give 1000s of cores and the job can be distributed into these cores and we can get faster solutions.

So, they are also following some sort of parallel computing paradigm that is the main job is decomposed into multiple small tasks and this small task goes to the cores of the coprocessor or accelerator. The difference is that the cores or the processing units of the coprocessor or accelerator are not as robust and as complex as ordinary CPUs.

They are very simple and rudimentary , they have less control, they probably also have less complexity in memory management and they can handle very simple instructions. So, a program has to be extremely granular and it has to be broken down into multiple simple concurrent tasks which can go to the processors of the accelerator or coprocessor and we can get faster performance.

Interestingly, this was a 2015 paper, but in last 5 years accelerators have made sufficient progress and if we see even just after 2 years if we see Pascal GPUs performance for simulation of flow over turbine blades is a very real life like problem simulation and the mesh size or the associated problem size is large, it is a 3-dimensional geometry. So, it is solving over a few millions, 10s of millions of points. We can see that at that time the latest version of GPU which is Pascal 100 GPU the speed up was 113 times.

Now, we have built up some idea on parallel computing in the previous discussion. So, getting 113 times speed up in a CPU based system will typically require around 150 to 200 of CPUs even if you have a very efficient parallel architecture and algorithm and that will require a huge computing system at least one rack with multiple blades connected to high speed Ethernet cable etcetera, but the same speed up is obtained just by putting a GPU card into the computer.

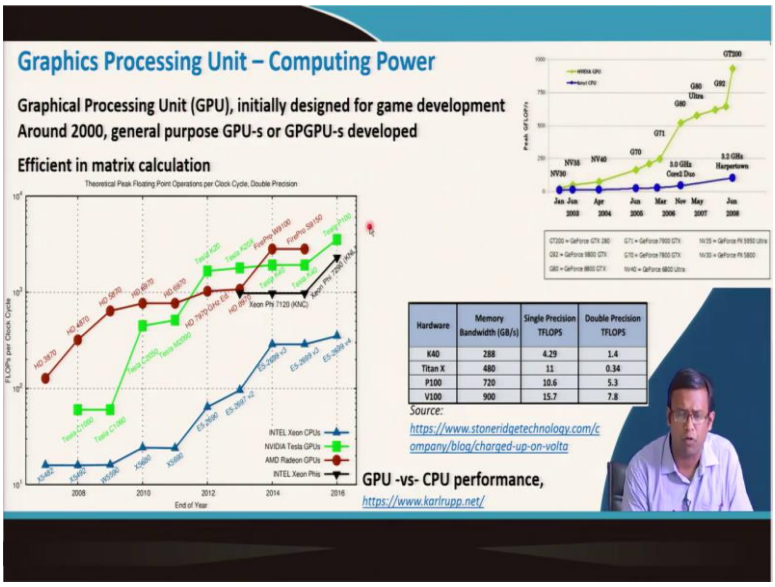
Therefore, accelerators; coprocessors are also showed that promise, but due to several issues including the market policies coprocessors are not that available now, but accelerators like GPUs can be utilized to get same benefit which you are getting from a very large data center or a large parallel computing architecture.

So, over a desktop in a small room we can put the GPU card in the computer and can get similar functionality and therefore, it is very important that we explore GPUs in more detail and we will see that many of the modern parallel computing infrastructures are accelerated using GPUs and many leading software in high performance scientific computing are coming with GPU capability.

So, you have to see, what is in GPU, and how it can provide such a high speed up. Another thing which is to be noted down here is that with different versions of GPU the speedups are different right and from K version to P version there is a huge jump in the performance.

This P version is the Pascal, K is the Kepler version; Pascal is a more recent version of GPUs. So, GPUs are also making certain progress in terms of architecture and hardware which is enabling them to give better speed up with the recent more recent versions...

(Refer Slide Time: 15:07)



Initially GPUs are designed for game development and visualization purposes. The idea as very simple; we think of something like a strategy game where I have the entire computer monitor with me and a different part different activity is happening due to the game development. Therefore, different processing cores are associated with different parts of the monitor on which some of the game's outputs are coming.

So, when graphics card came the idea was that you distribute the monitor into multi multiple pixels computer display into multiple pixels and attach different processing units with them. Around 2000 the vendors and the scientists working on GPUs identified that this GPU functionality can be extended to some other activities apart from only gaming or computer graphics.

It is also seen that as different processors can be associated with different locations or different activities, we can somehow consider a matrix and associate different cores of GPUs to different

rows of the matrix or different elements of the matrix and can get an efficient matrix calculation.

In fact, the jump or leave from ordinary GPUs are graphics solely working on graphics to GPGPUs came through when GPUs were enabled to matrix multiplication.

Well, so another interesting thing which is observed is that, if we consider the peak power of GPUs and compare them with CPUs, over here starting from 2003 when GPUs are more being utilized as GPGPU over years it is seen that compared to CPUs GPUs are made substantial progress in terms of the peak performance in gigaflop per second.

There is a comparison between Intel CPU and Intel GPU and we can see in 2008 a single GPU card can give almost say 6 to 7 times faster speed compared to 3.2 gigahertz CPU.

So, GPU can give better performance which was observed till 2008 and the reason was simple that if you want to improve performance of CPU the amount of hardware modification you have to do and the challenges posed by that especially from the space or footprint issues as well as from the thermal cooling issues makes it more difficult to put more transistors in the CPU and increase its speed.

But in GPU as the computing cores do very simple work, they have less control ,they can do simple arithmetic calculations, you really do not need to deploy more transistors for controlling the core sub GPU and therefore, the transistors which can be put into the can be mostly used for computing purpose and therefore, GPUs can give better performance.

However, the simplicity in the design of GPU makes it more difficult in terms of programming and we will look into that later. This was the story of 2008, now if we see after 2008 what happened. CPU performance improved to certain extent, but GPU performance kept on improving at a probably at a faster rate and now we can see there is almost 10 times faster speed which is obtained in a Tesla P 100 GPU which you have shown here in 2016 that was launched in 2016 compared to an advanced version of E5 Intel CPUs.

There are other GPUs from AMD and there are coprocessors Xeon Phis the accelerators are at one particular range and the CPU performance it is at a much lower range. As time progressed there are better GPUs, there are higher versions of GPU, CPUs have also made some progress. We can see that the Flops per Clock Cycle in CPUs have increased also, but in GPU the increase

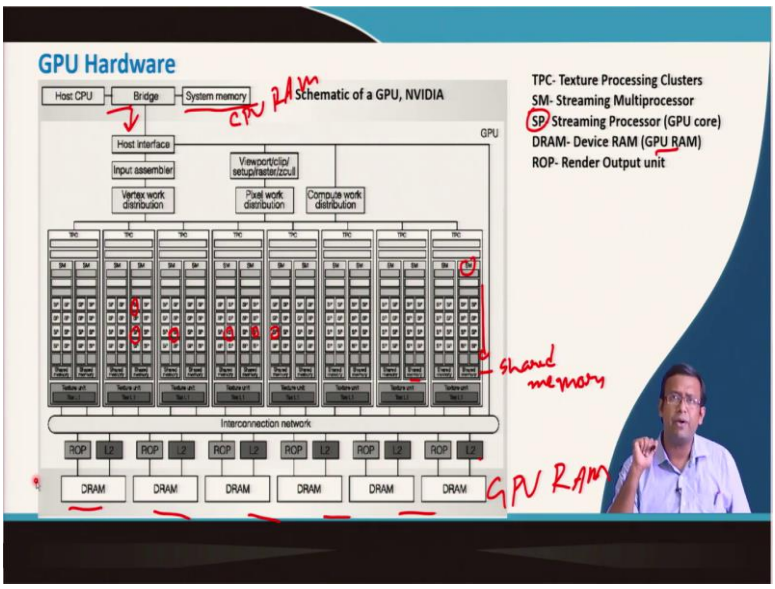
is more. So, the GPU computing power is increasing at a higher rate in time compared to the CPUs. So, all the accelerators or coprocessors they show a greater promise in giving us faster solutions.

We can also see that if we will just look into the GPUs and these are the GPUs coming from NVIDIA only. So, the different versions for example, K 40 that version was available in 2014 and P 100 was available in 2016, in 2018 Volta a newer architecture of GPUs were available. If we see the performance, their performance increased rapidly in terms of the memory bandwidth, in terms of single precision floating point operation and more rapidly in terms of double precision floating point operation, theoretically a single Volta GPU card can give 7.8 Teraflop speed.

The improvement in double precision floating point calculation double precision calculations we required in scientific computing solutions for there for less numerical error and better accuracy of the solution.

So, improvement in double precision performance poses GPUs as the best candidates for doing HPC in today's world. So, that is how performance wise GPUs acceleration gives us better speed up and the and the betterment is increasing day by day.

(Refer Slide Time: 22:18)



Now, what is in a GPU? That becomes a new next question. Well. So, this is a schematic of a GPU and this is schematic of an NVIDIA GPU, NVIDIA is one of the GPU vendors. I told you

that there are two leading GPU vendors NVIDIA and AMD, this is from NVIDIA. So, GPUs are accessories of the in the main computer.

So, the entire GPU system is connected via bridge to the host CPU. So, GPUs are not computer GPUs, they are accessories of a computer. So, these GPUs are to be connected to the main computer to work. So, there is a host CPU the GPU device is connected to it. So, it is identified as a device by the computer.

This is preliminary design for graphics purposes. So, you can see vertex work distribution, pixel work distribution, there are distribution for looking into vertex of the graphics and there are work distribution for looking into pixels inside each of the graphic cells and then because we do computation there is a work distribution scheduler for compute work.

Now, inside the GPU hardware there are texture processing clusters, texture processing again it is as the name suggests this comes from the graphic set domain. These are preliminary developed for graphics purpose; however, we are using it for scientific computing purposes, some of the names they are carrying the legacy of the graphics application capabilities.

So, inside the texture processing clusters there are multiple streaming multiprocessors, inside each streaming multiprocessor there are a number of streaming processors and these streaming processors are the SPs or streaming processors. So, these are the streaming processors these are the cores of the GPU.

You get a number of streaming processors combining them and all the streaming processors can be used concurrently or some of them can be used by a scheduler may be all of them cannot be used concurrently, but a large number of them can be used concurrently and a large number of concurrent instructions can be processed.

Now, these instructions are coming from the host CPU to the GPU. Then we can see that there is a device RAM, GPU has a separate RAM, the CPU sorry. The CPU that comes with its own RAM this system RAM is CPU RAM. The GPU comes with its own RAM this is GPU RAM. So, there is a device RAM or GPU RAM and there are rendered output units which in the name suggests these are some sort of control units, the name suggests that this comes from the idea of graphics design.

There is an L2 cache, but you can see that this L2 cache is quite far from the streaming processors, there is an interconnection network through which the cache is attached. So, the latency is typically high for GPUs because the cache is not directly sitting near the GPU cores or near the streaming processors.

If you come close to the streaming processors there is something called shared memory, which is close to the streaming processors. So, the entire streaming multiprocessor shared one common shared memory, but this memory should have less bandwidth.

So, from a memory perspective because there is an interconnect network which connects from RAM to the GPU cores and there are L2 caches which are quite away from the processing units or streaming processors from memory perspective memory latency is probably an important issue in GPU computing and we will see truly that it is there. Rather this shared memory thing which is not present in CPU architecture they are present here in this GPUs.

So, what we understand here is that architecture wise GPU is quite different from CPU and when we will do GPU programming, we have to keep in our mind different types of memory hierarchy in GPUs. We look into it in detail when we will look at CUDA programming. The memory hierarchy is in GPU. A programmer has to be aware about hardware while doing GPU programming.

In CPU it is actually simple we know that there is a RAM, there will be some data flow from RAM to cache, hardware aware program is good, if you can take care of how cache lines will be read, what are cache hits, cache misses, this is good in CPU programming, but even if we do not that we will get reasonably good performance we can get much better performance if we know that well, but in GPU if we are not quite aware about the hardware we can end up in getting very bad performance.

(Refer Slide Time: 28:43)

Tesla Product	Tesla K40	Tesla M40	Tesla P100	Tesla V100
GPU	GK180 (Kepler)	GM200 (Maxwell)	GP100 (Pascal)	GV100 (Volta)
SMs	15	24	56	80
TPCs	15	24	28	40
FP32 Cores / SM	192	128	64	64
FP32 Cores / GPU	2880	3072	3584	5120
FP64 Cores / SM	64	4	32	32
FP64 Cores / GPU	960	96	1792	2560
Tensor Cores / SM	NA	NA	NA	8
Tensor Cores / GPU	NA	NA	NA	640
GPU Boost Clock	810/875 MHz	1114 MHz	1480 MHz	1530 MHz
Peak FP32 TFLOPS[1]	5	6.8	10.6	15.7
Peak FP64 TFLOPS[1]	1.7	.21	5.3	7.8
Peak Tensor TFLOPS[1]	NA	NA	NA	125
Texture Units	240	192	224	320
Memory Interface	384-bit GDDR5	384-bit GDDR5	4096-bit HBM2	4096-bit HBM2
Memory Size	Up to 12 GB	Up to 24 GB	16 GB	16 GB
L2 Cache Size	1536 KB	3072 KB	4096 KB	6144 KB
Shared Memory Size / SM	16 KB/32 KB/48 KB	96 KB	64 KB	Configurable up to 96 KB
Register File Size / SM	256 KB	256 KB	256 KB	256KB
Register File Size / GPU	3840 KB	6144 KB	14336 KB	20480 KB
TDP	235 Watts	250 Watts	300 Watts	300 Watts
Transistors	7.1 billion	8 billion	15.3 billion	21.1 billion
GPU Die Size	551 mm²	601 mm²	610 mm²	815 mm²
Manufacturing Process	28 nm	28 nm	16 nm FinFET+	12 nm FFN

[1] Peak TFLOPS rates are based on GPU Boost Clock

So, some concept of hardware is essential while looking into the GPUs and we will look into the specifications of the GPU hardware. So, there are 4 GPUs which are considered, all these 4 are coming from NVIDIA.

This is NVIDIAs V100 GPUs white paper which we are discussing, Tesla V100 this is the latest version of NVIDIA GPU, the previous version was Tesla P 100, M 40 Tesla K 40 and the names are named after different scientists Kepler, Maxwell, Pascal, Volta. We can see that if we go to the latest version V100 it has 80 streaming multiprocessors and 40 texture processing clusters. So, each TPC has 2 streaming multiprocessors.

Then, each streaming multiprocessor has 64 single precision or 32-bit cores and the total number of cores therefore, becomes 64* 80 =5120 single processor cores. There are 32, so we can see there are FP 32 cores against each two FP 32 cores there is one FP 64 core. So, against two single precision cores there is one double precision core and double precision are what we are interested in for scientific computing.

So, there are 32 double precision cores in all 80 streaming multiprocessors that gives us 2560 cores available in each GPU. We can see that there are texture units the Peak Teraflop speed in double precision is 7.8 and in single precision is 15.7 and we can see that the number of cores is in single precision is just double of the double precision cores for V 100 therefore, the speeds are different. If we go back to K 40 the single precision cores were more at almost 3

times then the double precision scored and therefore, the double precision speed was one - third in K 40.

So, the recent architectures are better in terms of double precision scientific computing. Shared memory is the memory which is sitting on the GPU chip is up to 96 kilobyte and it has also increased it is started from 16 kilobyte in K 40 it is up to 96 kilobytes here and the L1 cache can be configured in the shared memory part only. So, L1 cache has also increased in Tesla V 100 therefore, the computing speed will be increased because latency will be small here.

What is important is that, the newer GPUs have large register files, register size is near 20 megabytes. So, there is a large register files and therefore, large number of registers are available for each of the cores each streaming multiprocessor has 256 kilobytes register and these registers can be values like loop unrolling type of things can be beneficial while doing GPUs and there are certain other uses of this registers also, we will discuss it later.

L2 cache size is 6 megabyte and the GPU RAM it was initially was up to 12 gigabytes in K 40 in V 100 one can get 16 gigabytes. So, this is also important that GPU RAM is 16 gigabytes; that means, it can work on a large memory, but it cannot work on very large memory.

So, if you have an extremely large application which requires a 100 gigabyte RAM size, a single GPU is probably not sufficient or you have to do something while mapping the CPU memory to GPU memory, well. So, we will continue with more detail in the GPU architecture and continue with these discussions.