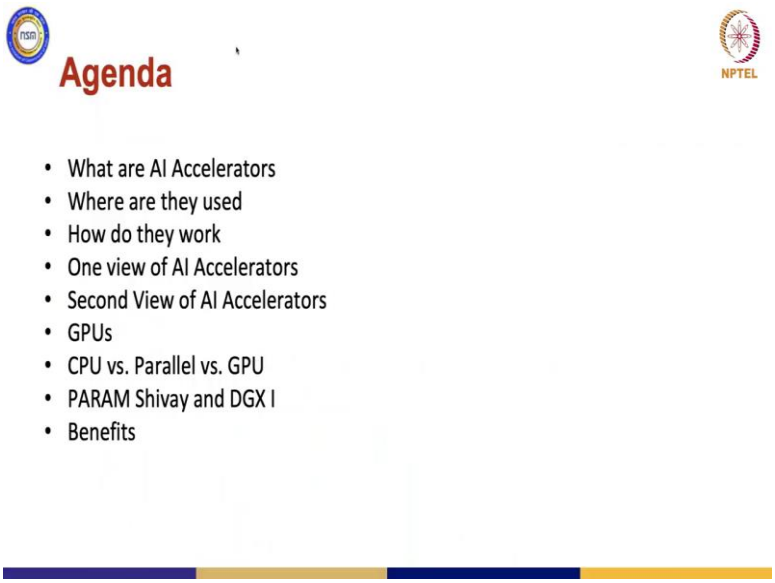
Applied Accelerated AI
Prof. Satyadhyan Chickerur
School of Computer Science and Engineering
Indian Institute of Technology, Palakkad

Introduction to AI Accelerators
Lecture - 03
GPUs Part 1

Good evening, everyone. Welcome to the 2nd day session on Applied Accelerated Artificial Intelligence workshop. So, today we would be discussing about AI Accelerators and to be very specific we will try to discuss in terms of what we are talking about is GPUs today right.

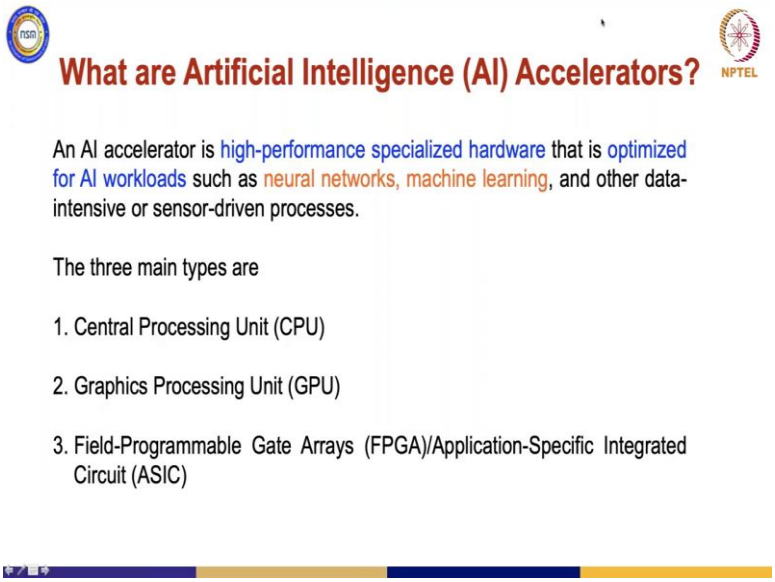
(Refer Slide Time: 00:40)



So, the agenda of this particular session of the day would be like this. We will first start with what are AI accelerators, where are they used, how do they work, one view of the AI accelerators, second view of the AI accelerators. Then what are these GPUs, how do you actually think of writing a program in terms of just running it on a CPU as against on multi core or a parallel program versus a program which you write it on a GPU.

Then we will try to see the architecture or the setup of PARAM Shivay which is a cluster which also has GPUs and DGX I. And we will do some demos on these particular systems ok, very brief small demos to start with and then we will end up with what would be the benefits of actually working on such a hardware right, ok.

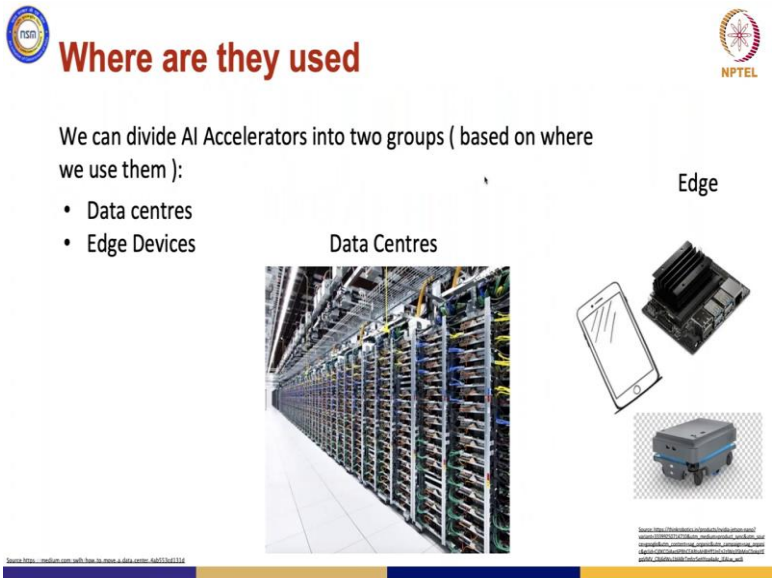
(Refer Slide Time: 01:50)



If you see a AI accelerator is a high performance specialized hardware that is specifically meant for AI workloads which in turn would be neural networks, machine learning programs, and intensive programs which are basically sensor driven, which involve certain processes, which also would link all of this right. So, you can work with a program which has neural networks, which has machine learning right which is linked to inputs from some sensor devices.

And if you classify it, there are three main types of artificial intelligence accelerators for that matter. You have a CPU; you have a graphics processing unit or a GPU and then you have Field Programmable Gate Arrays or ASICs as well. So, these are three actual areas or devices basically, wherein you can work with for your AI workloads.

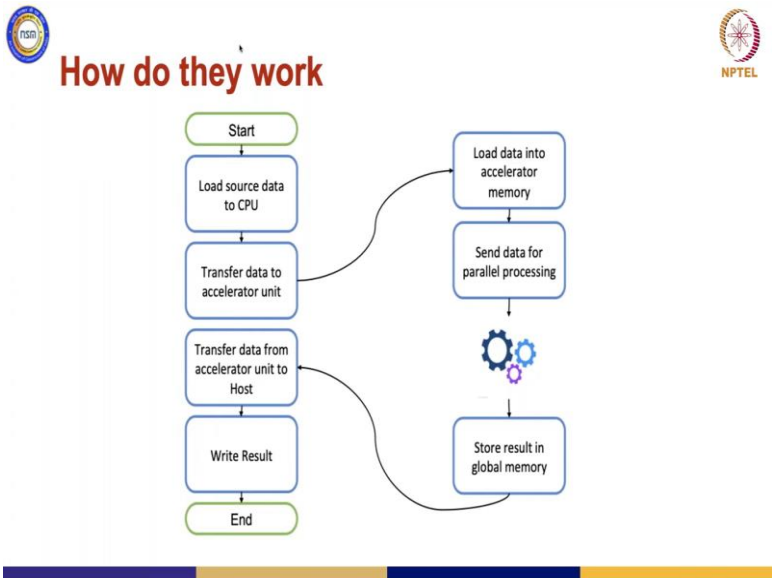
(Refer Slide Time: 03:11)



And if you see that way from the classification point of view, you can divide these AI accelerators into two groups based on where actually you can use them right. So, you have these AI accelerators at data centers and you have these AI accelerators which are on your edge devices.

Or the devices where you actually do inferencing, which are not very computationally strong devices right. The devices which are there on the edge. So, this is how the classification basically looks like, when you talk in terms of where they are placed and how you are going to use them.

(Refer Slide Time: 04:05)

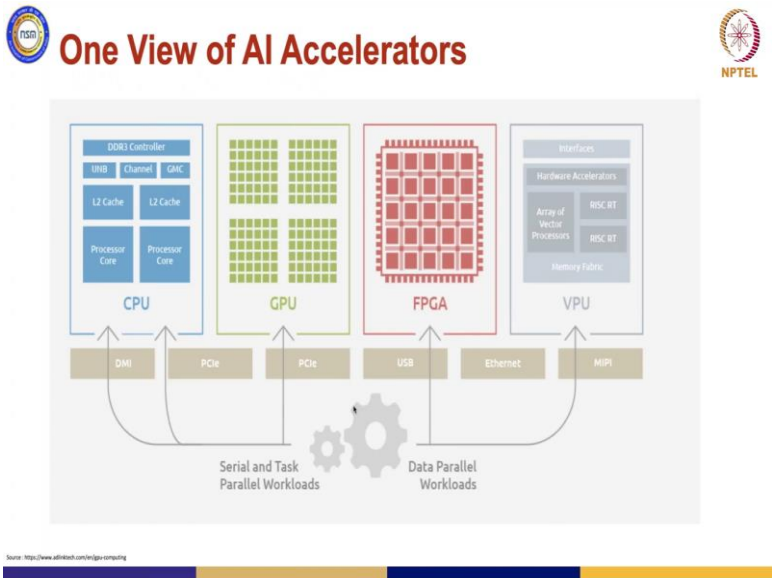


Now, technically speaking, any of these accelerators right are supposed to be working in a coprocessor mode. So, if you have not worked with it this is how you actually try to develop a program which could run on accelerator. You have got two types of programs right. The programs which are there which are run on CPU, then there is some portion right of your data the whole program which you need to actually transfer to the accelerator and the accelerator does the processing for you.

You get the result again and then this result is again transferred to your host. Host is nothing but your CPU for that matter. So, CPU is a host, your GPU is a device this is the terminology which generally people use in these type of systems right. So, host is your CPU and the GPU is your device. So, when you start writing such programs, what effectively happens is not everything is to be offloaded to a GPU or accelerator for that matter.

The portion which is compute intensive, which is actually required to be run by the GPU will be required to be offloaded to the GPU. And then it basically is executed and then return back to the host, which actually does all the other types of work right. So, this is how they are used by people when you start developing your programs and you will start writing your programs right.

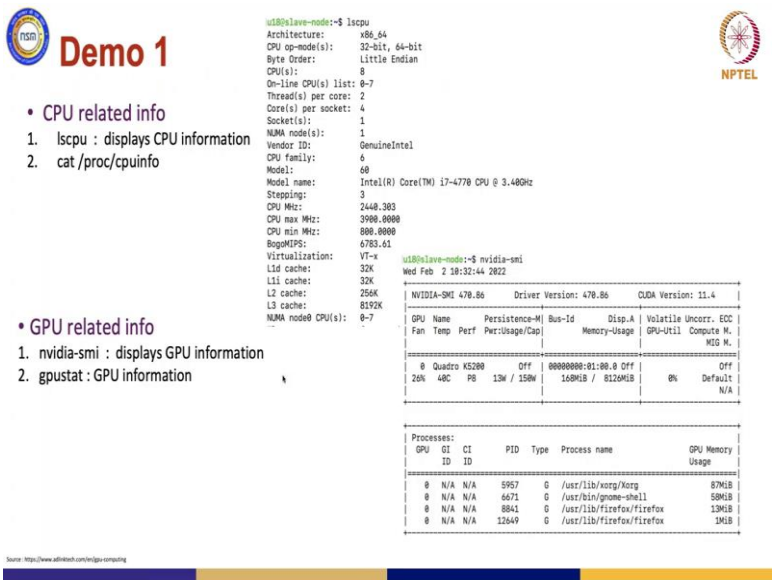
(Refer Slide Time: 05:52)



Now, one view of the AI accelerators ok is something like this. You have a CPU, you have a GPU you have FPGA and you have VPUs and all there are various types of accelerator. So, from one view of the AI accelerator, what I meant was you actually have a serial and a task parallel workload ok and then you have data parallel workloads. So, when you have a data parallel workload, it is very good for people to run it on a FPGA and a VPU.

Whereas, if it is a serial and task parallel workload, these type of workloads are effectively run through the CPU or a GPU and this all is in a coprocessor mode. So, you have PCI express through which your CPU and the GPU actually communicates. So, I hope it is clear that different ways of executing compute intensive workloads by different types of accelerators right. So, this is what is a gist of how you actually can split your workload among these various categories of accelerators.

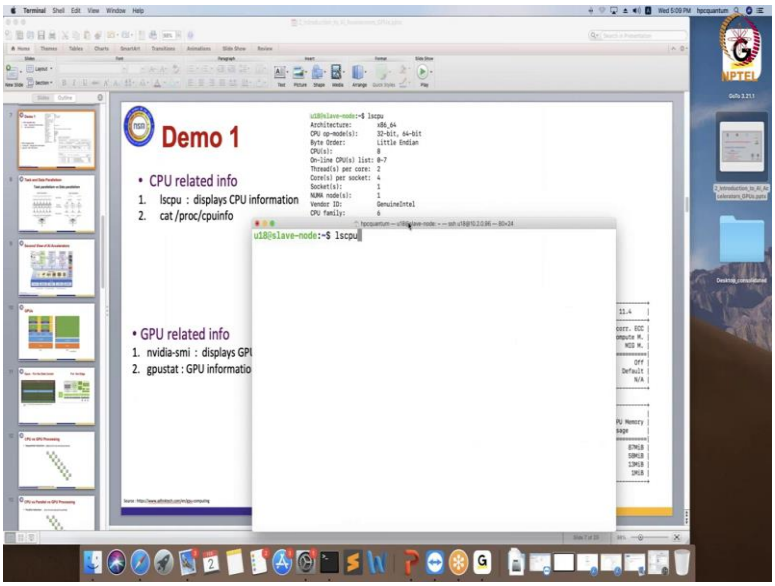
(Refer Slide Time: 07:15)



Now, let us try to do a demo and see if we are trying to work on a CPU ok. It might be a multi core CPU and then you can work on different types of GPUs. So, let us try to understand and do hands-on by these two commands right which are written here. CPU related information if you want to gather, there is something called as lscpu which displays the CPU information and then you have this cat/proc/cpuinfo, these two are going to give you the information about the CPU configuration of your system, the snapshots are attached.

Similarly, when you see the nvidia-smi command it displays the GPU information and you have this gpustat which gives a better information also a bit of it. So, let us try to do this hands-on and see at three different places ok.

(Refer Slide Time: 08:33)

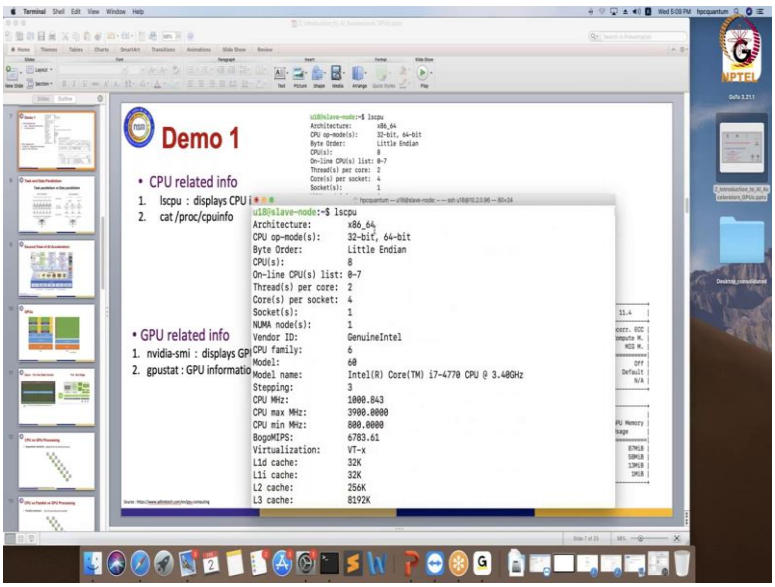


So, I will just go to those places and see ok. So, now, I hope this particular command prompt is visible to everyone.

Student: Yes, it is.

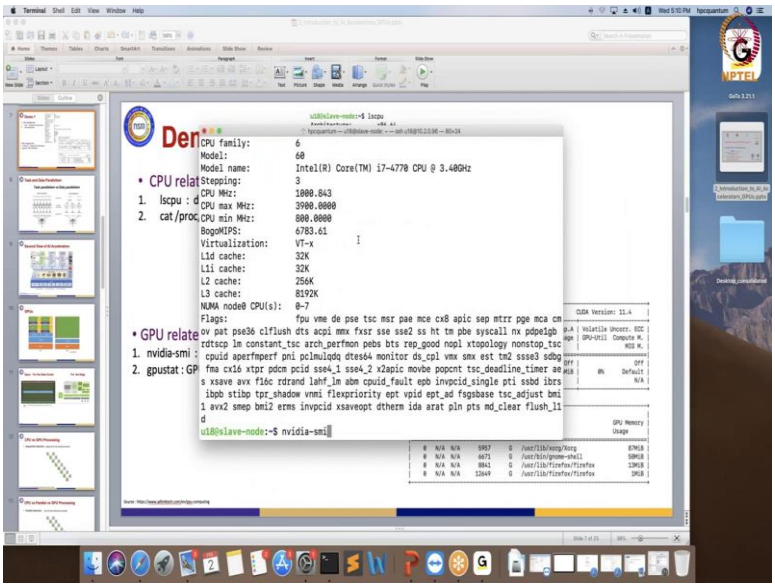
So now, this is a GPU system right, this just a workstation with a GPU ok. So, we will try to do two things now, first we will try to understand what type of CPU this has got, ok.

(Refer Slide Time: 09:07)



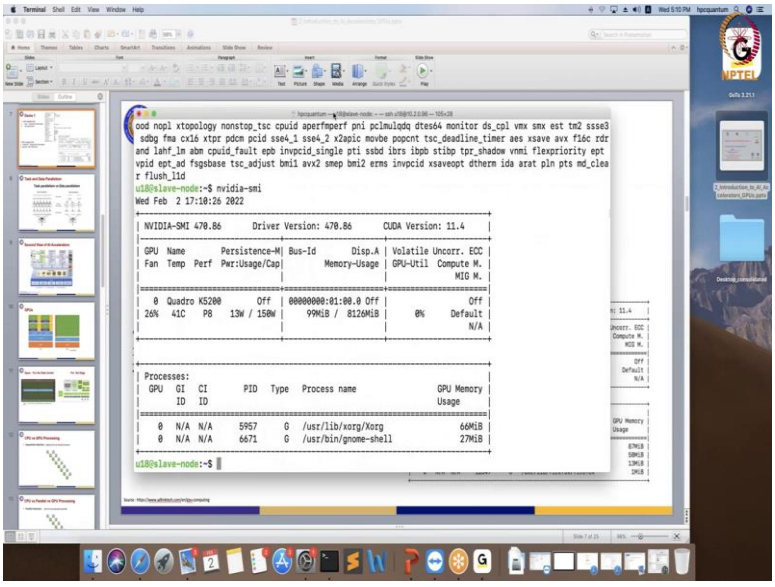
So, if you see this lscpu, the architecture is informed, we have what type of operational modes this CPU can work on, what type of Endian-ism is there ok, how many CPU cores are there in this ok.

(Refer Slide Time: 09:34)



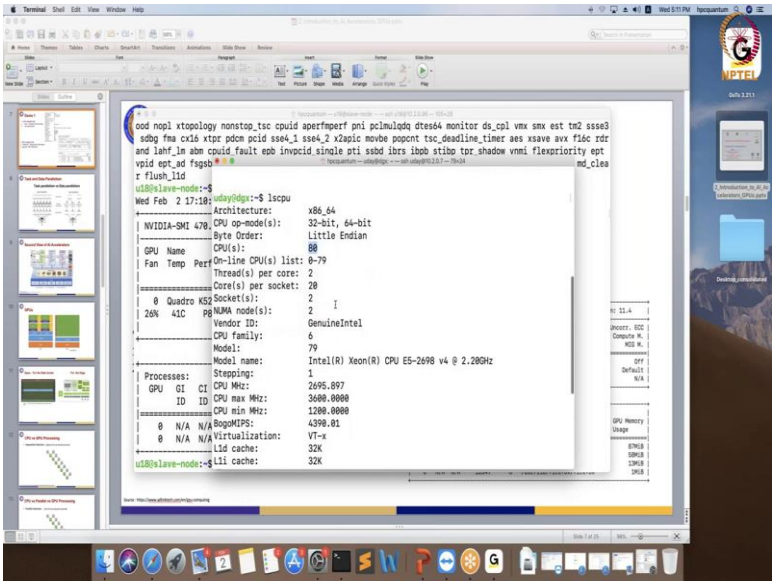
And then what is the megahertz speed of a CPU, what is the maximum speed, what is the minimum speed ok and the model and how many threads per core. So, hyper threading concept is there and how many cores per socket. So, this is a general thing about this particular system right. Now, let us try to understand what type of GPU this has got.

(Refer Slide Time: 10:03)



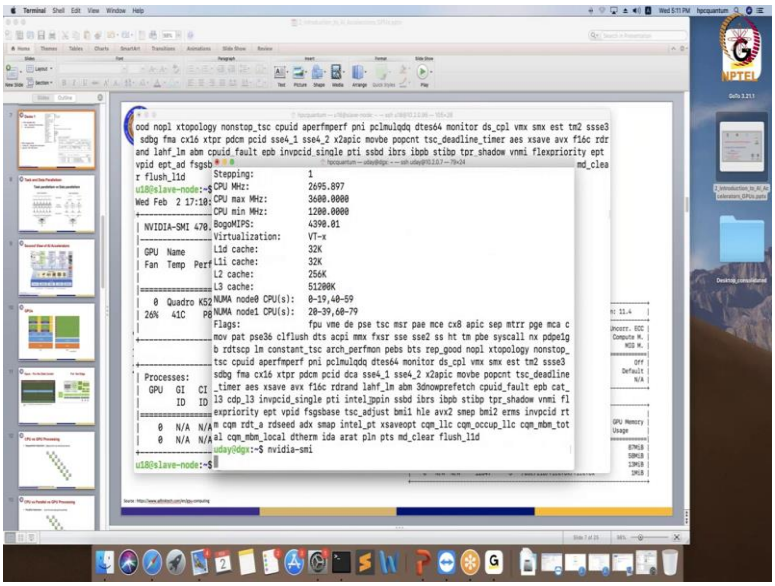
Now, you see this; this particular system has a Quadro K5200 GPU. So, this tells about the GPU number ok, the name and all of this like fan speed, then compute node, the index ok all of this. So, this you can go through in detail when that afterwards, but you get the information about this particular GPU. Now, let us try to go to another place now which is RDGX.

(Refer Slide Time: 10:52)



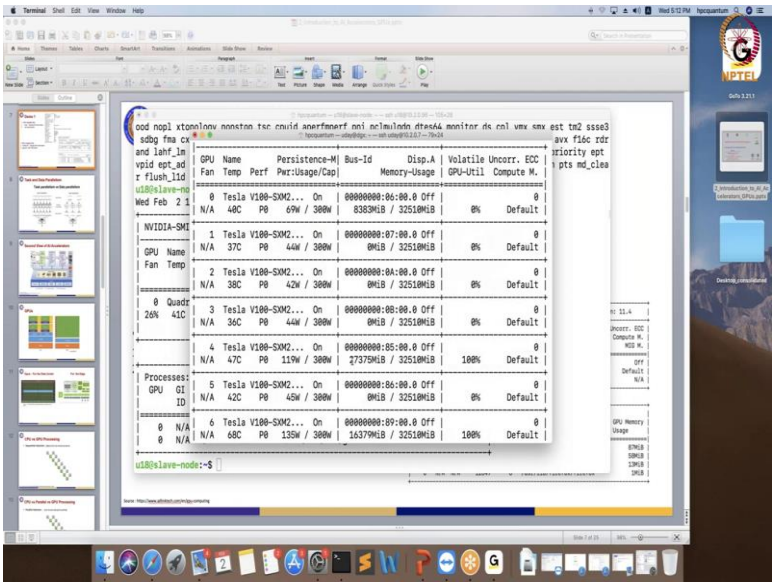
So, let us try to do the same thing again.

(Refer Slide Time: 11:13)

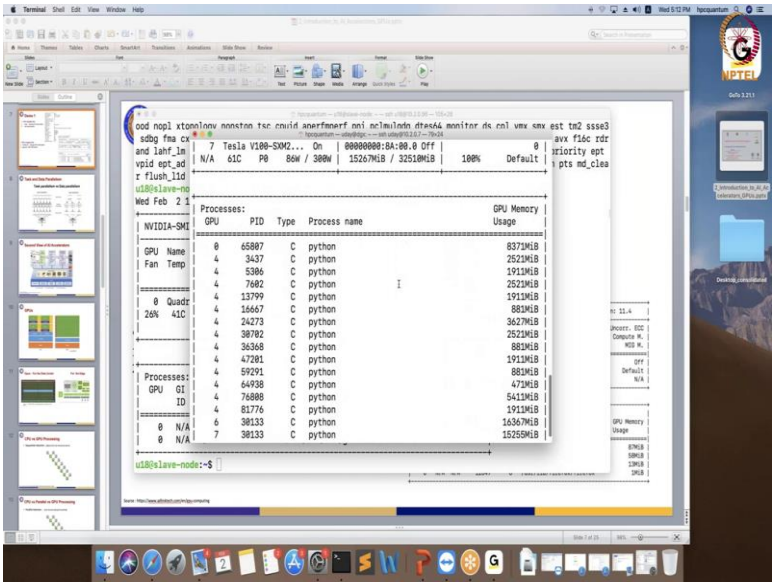


So, if you go to the DGX. So, this DGX has got 80 cores, 80 CPU cores that is what it means right. And you have 20 cores per socket, then you have on node 0 CPU, what are the numbers 0 to 19, 40 to 59 and node 1 CPUs 20 to 39 and 60 to 79. So, this is a DGX server right, DGX I. So, let us try to understand the GPU information about it ok.

(Refer Slide Time: 11:41)



(Refer Slide Time: 11:50)

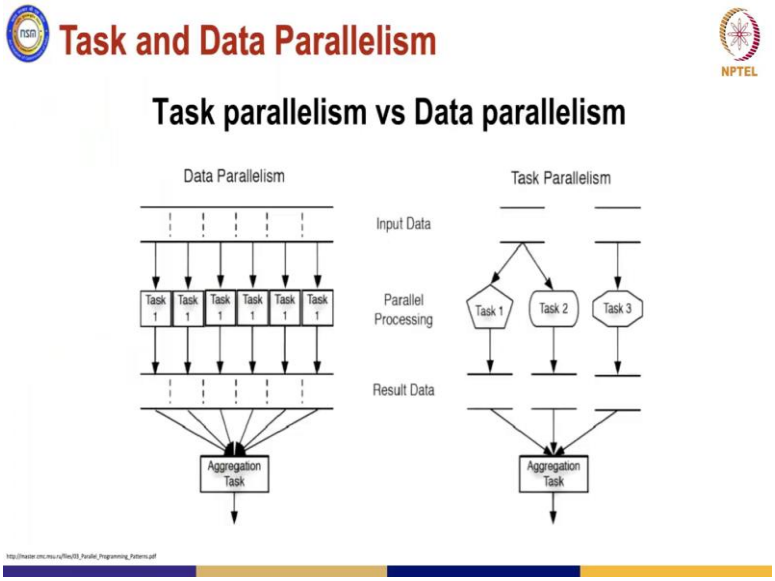


So, if you see this, this is a DGX and if you see the GPUs, you have got GPU 0, GPU 1, GPU 2, 3, 4 and up till 7 and these are all Tesla V100s ok. And you have got the

memory, the utilization of that memory on that particular GPU ok. So, this is how the information about your total GPU system ok you can gather using nvidia-smi.

And then for each of these GPUs which are the processes which are running ok, and how many what is the memory which that particular GPU is using right, all of that information could be gathered from these two commands right. So, lscpu and nvidia-smi.

(Refer Slide Time: 12:55)



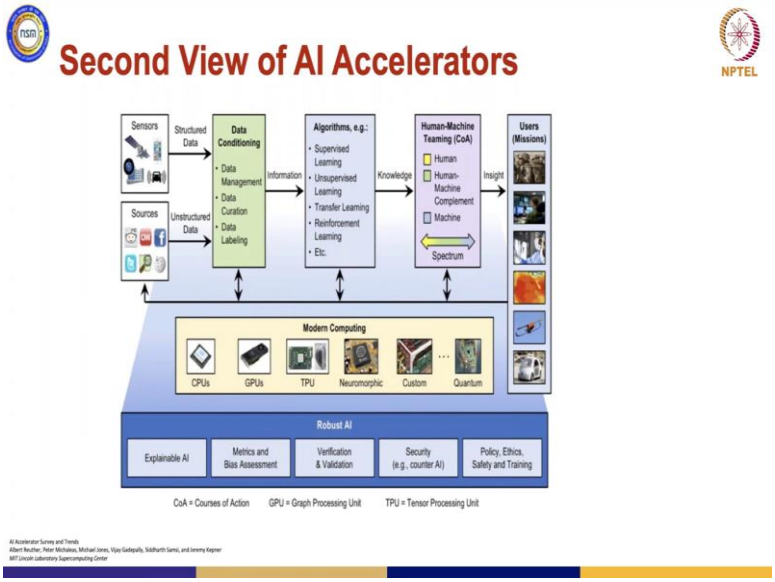
So, now let us go to the next slide which talks about, which talks about the parallelism which I had just told right, I had told about the task parallelism as well as the data parallelism. So, let us try to understand it in brief as to when you talk of data parallelism the data actually is split into various blocks and given to various tasks. So, there is a data decomposition which actually happens too much.

And then this is how like this block of data goes to this task, this block of data goes to this task, this block of data from third block goes to the third task right. And something like this and then you have this aggregation of all of these results and then do this. This data parallelism when you talk task parallelism let us say you have a small section of data.

Then you actually split ok, that particular data which can be unique for each of the tasks and then it is not necessary that here the tasks will use the same data or these tasks will do the same work ok. So, basically it is the decomposition at what level, decomposition

at the task level, decomposition at the data level right. So, here these are different tasks, you get the same data or you get different data for the same type of task right. So, this is how actually is a basic difference between the task and data parallelism.

(Refer Slide Time: 14:44)



And from the second view of AI accelerators, what I meant was you can use these accelerators, if you see here it starts with CPU, GPU, TPU and then all the way up till quantum accelerators. So, the idea here is that all of these different levels of complexities which are available in these accelerators could be used as different levels of your programming applications right.

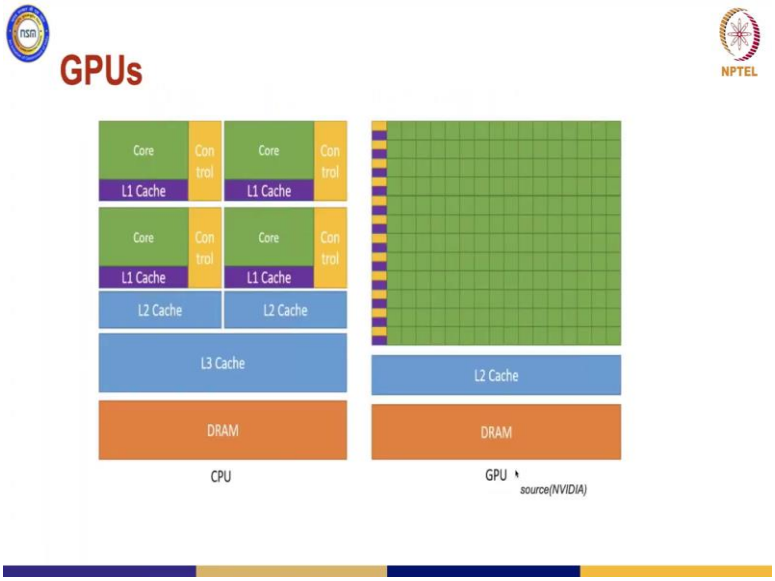
So, you actually can use with these your sensors, you can do data conditioning, you can develop in your algorithms, you can do actually human in loop and all of that human machine interaction at various levels, various complexities. The same hardware could be used right and then you can have application development for various users, then you talk about explainable AI, you can talk about matrices, you can talk about verification, validation, policy, ethics safety training.

So, it actually is a very huge gamut of applications, course of action require requirements and then the application or the area where you are working in and the hardware. So, these are all linked right and this is what I thought like we would cover it as a second view of AI accelerator right. Two views of AI accelerator, one is from the view of how you are going to do computations only computational workload and the second thing is

how can it be used for other things right, like inferencing or data conditioning or something processing the structured data or unstructured data.

Of course, they are all considered as data workloads, but the idea was at different places you want different type of accelerators and different type of usage for those accelerators right. So, that was the intention of this specific slide ok.

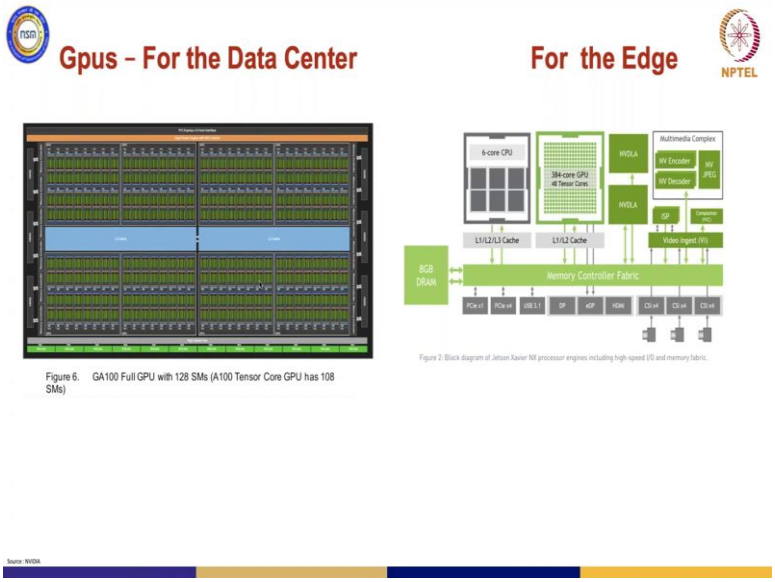
(Refer Slide Time: 16:57)



So, now let us try to understand the difference between the CPU and the GPU, which is something like this. It is basically having multiple cores, the CPU and you have L1 cache for each of the core, then you have controls unit for each of these cores. And then you have got L2 cache, L3 cache and DRAM. For GPU you have got these huge massively parallel processing cores with you and each of these ok, have got their own control and cache and then you have L2 cache you have the DRAM, the way in which these are used for massively parallel applications ok.

The way in which they are programmed are bit different, then yeah that is how basically it is done. Not going into the details of them, but this is how the difference lies in the CPU architecture and the GPU architecture.

(Refer Slide Time: 18:16)

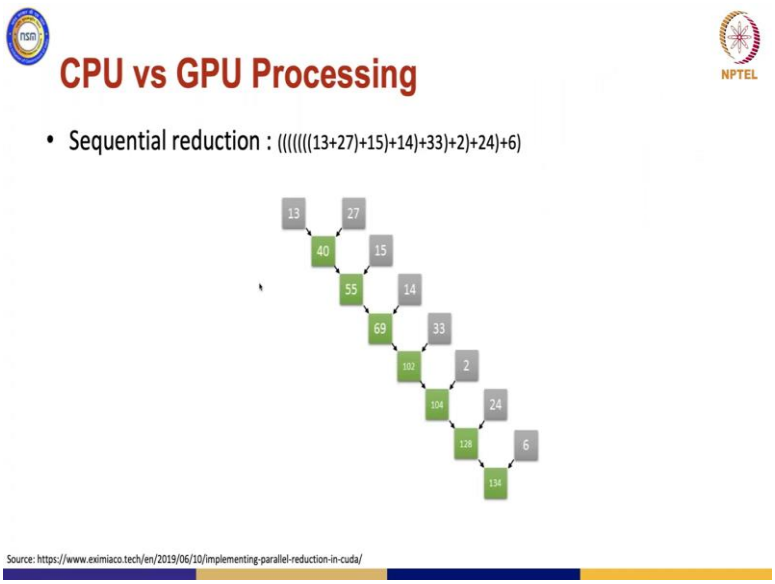


So, massively parallel programming is supported by GPUs and this is an example which I thought, this slide shows if there is a GPU right for the data center and then the same type or the similar family of GPUs ok for the edge. So, the idea was the difference is in this design of how many cores you are working with right, how is the connectivity between the CPU and the GPU stuff, because this is totally a embedded type of a Jetson Xavier NX processor engine which require which also has a high-speed IO and memory fabric.

So, this is for the edge devices, wherein when we would be doing inferencing we will try to show you how to use such devices for doing your inferencing portion of your problem which you are trying to solve, right. And this is the GPU for data center you have B100s, you have A100s right and so many of them which are to be used by data centers for training your models right, coming up with lot of different type of approaches for models.

Then you have basically so many other things which you can do on these GPUs, which are basically for the data center right and you have this unveiling stuff and all of that. So, this is the difference in the architecture and the design of these accelerators at the data center level and at the edge level ok.

(Refer Slide Time: 20:00)

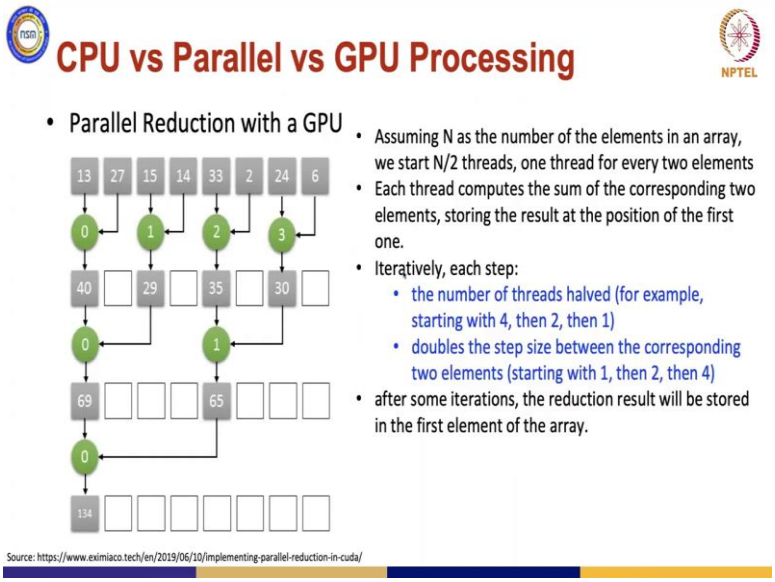


So, now let us try to understand a very basic thing of something which is called as sequential reduction. For example, you have to add these numbers together. So, let us say how does a sequential processing happen for it, it is something like you go on adding 13 plus 27. Then you get that and then you add 15 to it, then to 14 then 33, 2, 24 and 6. So, this is a way in which a sequential processing happens.

So, if you see this particular type of reduction operation, you will see that you cannot do anything in parallel, it cannot be done parallelly right, because this particular thing depends on the sum of this and so on and so forth. So, this is basically a way in which a CPU actually works for you, when you come down to a parallel reduction ok.

So, the idea is something like this, you can do this particular addition ok at one particular on one particular core or one particular CPU or a processing element and then you can do this on a different processing element and so on and so forth. And then you can go on summing it up, right. So, this is a way of parallel reduction.

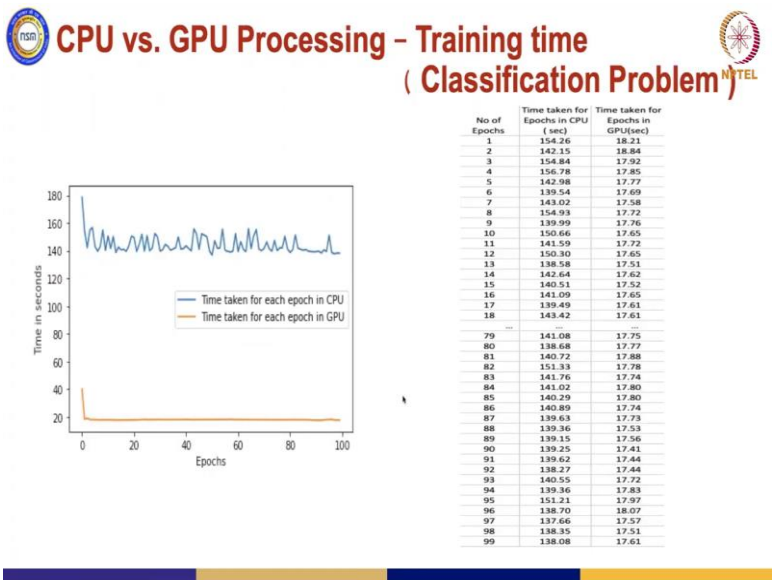
(Refer Slide Time: 21:22)



Now, how do you do this parallel reduction using a GPU? So, the idea is we are trying to assume that you have got let us say N number of elements. So, you basically start with N/2 threads. So, you use one thread for every two elements, each thread computes the sum of the corresponding elements which you are using it for and then you store the result ok, in one of the threads.

So, iteratively at each step the number of thread is going to be half ok and the step size right, between the corresponding elements will go on doubling and ultimately you get the sum as a sum of a array of all of this and then you get the result. So, this is how actually you actually move from a sequential type of execution thinking to parallel and then to the concept of GPUs with many threads ok.

(Refer Slide Time: 22:39)

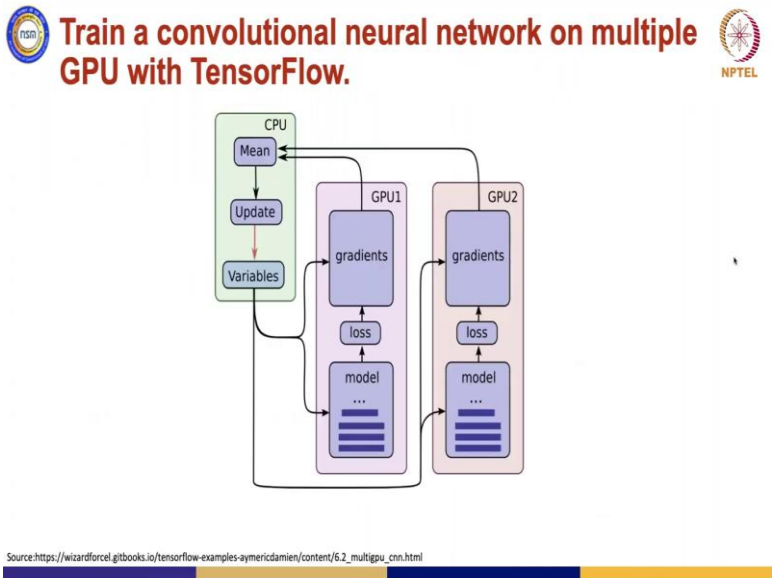


No of Epochs	Time taken for Epochs in CPU (sec)	Time taken for Epochs in GPU(sec)
1	154.26	18.21
2	142.15	18.84
3	154.84	17.92
4	156.78	17.85
5	142.98	17.77
6	139.54	17.69
7	143.02	17.58
8	154.93	17.72
9	139.99	17.76
10	150.66	17.65
11	141.59	17.72
12	150.30	17.65
13	138.58	17.51
14	142.64	17.62
15	140.51	17.52
16	141.09	17.65
17	139.49	17.61
18	143.42	17.61
...	...	...
79	141.08	17.75
80	138.68	17.77
81	140.72	17.88
82	151.33	17.78
83	141.76	17.74
84	141.02	17.80
85	140.29	17.80
86	140.89	17.74
87	139.63	17.73
88	139.36	17.53
89	139.15	17.56
90	139.25	17.41
91	139.62	17.44
92	138.27	17.44
93	140.55	17.72
94	139.36	17.83
95	151.21	17.97
96	138.70	18.07
97	137.66	17.57
98	138.35	17.51
99	138.08	17.61

So now, this is actually a snapshot which shows about the difference in the processing time for a small classification problem ok, which we ran on the GPU as against the CPU. So, here if you see for epochs vary ranging from 1 to 99 ok, you see the time taken for each epoch, on a CPU as against on a GPU right. So, you can see that you have got for if there is only 1 epoch to be run you get about 154.26 seconds time to get executed on a CPU as against 18.21 seconds on the GPU, right.

This was just to show you the time difference right between something which you run on a CPU and something which you run on a GPU ok, all the way up till hundred epochs. So, the idea was to just show you the difference, then show you the difference and then you can assume that just for a very small classification problem right of some diabetic retinopathy detection, just we tried doing it for some application ok. And this was just idea to be given so that the time difference for execution it on a CPU and a GPU ok.

(Refer Slide Time: 24:19)



So now, if you are trying to train a convolutional neural network on multiple GPUs with TensorFlow right, this is how it is going to happen. You have your CPU right and CPU has to be the master and then you have these coprocessors which are your GPUs. So, actually what is happening, when you try to develop a convolutional neural network or for that matter neural network you will have gradients, losses, weights and all of this right. And then you will have your own model right which basically would be running on a GPU.

So, you have these variables, you have to calculate the mean, the updation of all of this and then it has to be again fed back to your model for you know improving its accuracy and so on and so forth. So, the idea is this is how it is going to be linked. So, CPU does certain specific portion which is not actually effectively to be done by a GPU, because GPU is compute hungry type of a processing element right.

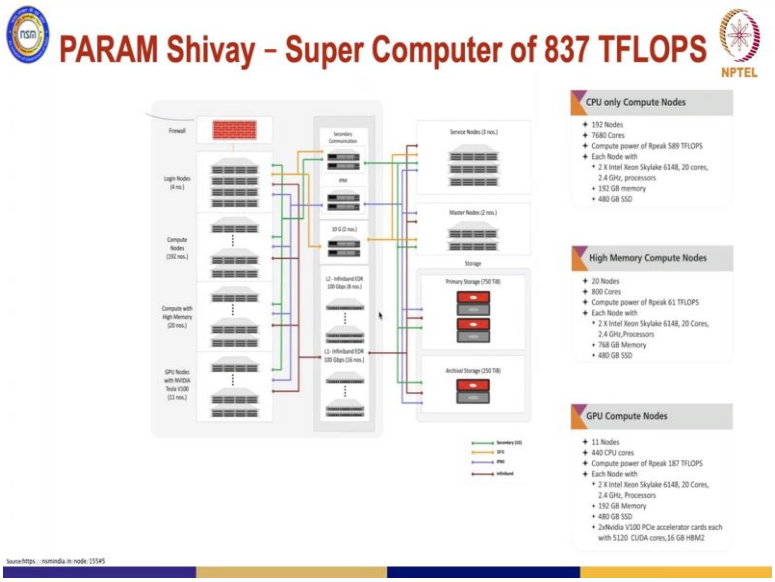
You have to give it lot of computations, you cannot depend on this data intensive or io intensive applications to be effectively run on GPU. So, that is one thing which you should keep in mind, that not all applications are suitable suited to be running on a GPU, they have to be massively parallel, massively parallel requirement should be there for that type of application.

Then only it actually gives you the benefit and the advantage of trying to solve a problem on a GPU. So, this is how it is going to actually be used when you are trying to go ahead

in these classes, because you will be working on various types of models, how do you accelerate them and all of that. So, this is the basic idea of what is going to come.

This is example of a multi GPU type of a setup, you can have a single GPU type of a setup and then you can have a cluster GPU with clusters right, that type of a setup. So, let us try to see in the days to come as to how we can use this type of a setup also for solving certain problems ok.

(Refer Slide Time: 26:43)



So, this is the PARAM Shivay cluster at IIT Varanasi, this gives you about 837 teraflops, if you see the configuration of this system right you have got about 4 login nodes, you have 192 compute nodes and then you have GPU nodes with NVIDIA Tesla V100. So, the same V100 which I showed you on the DGX.

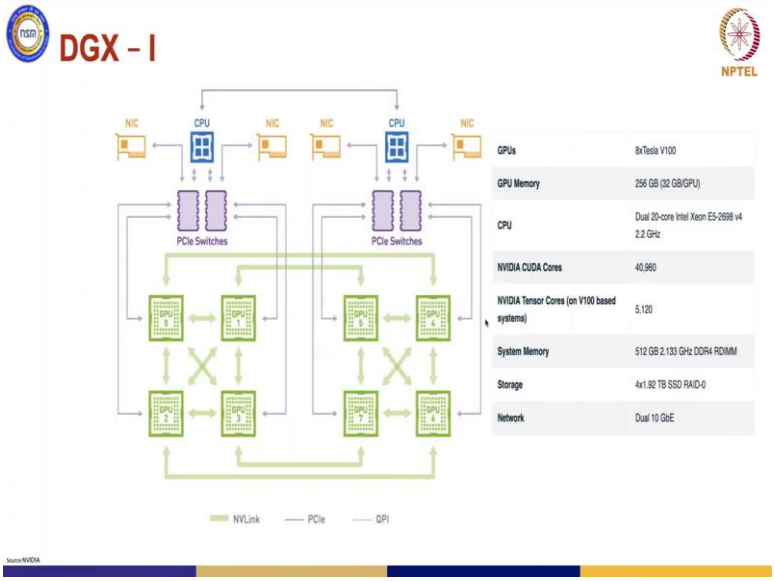
So, DGX has got a very specific type of a setup which I will show you in the next slide, but the idea here is here also we are using V100s, there are 11 numbers and then you have actually InfiniBand switch and then the communication, then you have primary storage of about 750 terabytes and 250 terabytes of archival storage.

Then you have got high memory compute nodes right. So, there are about 20 nodes, 800 cores all of this information is available on the NSM website, the reference is also given. So, if you see the GPU compute node specifications here, you have got about eleven

nodes with 440 CPU cores and then you have got about 2 x V100 PCI accelerated cards, each with 5120 CUDA cores right.

So, this is basically with each node which you are talking of. So, this is the total design of your PARAM Shivay cluster which you have and then this is the DGX I server actually.

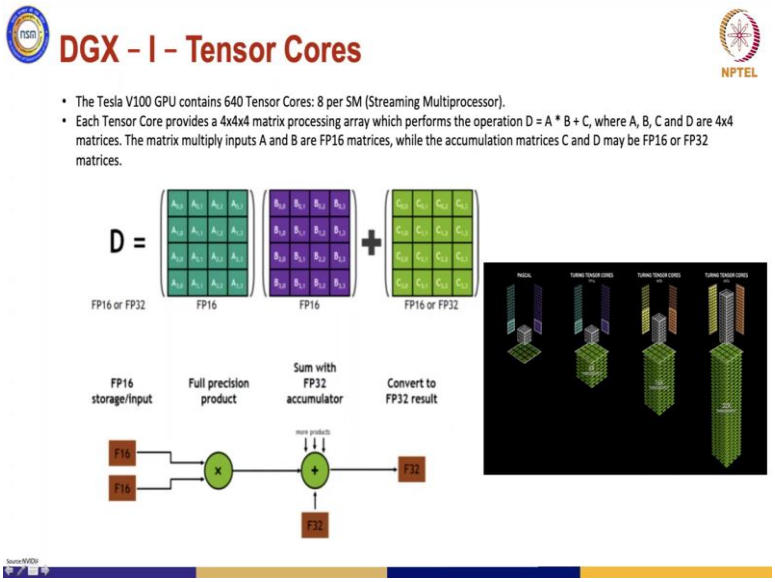
(Refer Slide Time: 28:35)



So, if you see this architecture, this is something which is having a NVLink right. So, NVLink is a proprietary bus link ok which is specifically developed by NVIDIA and it uses its NVLink in its servers right. So, if you see here you have got about 8 GPU cards right and then you have PCIe switches, then you have your CPU cores, then you have NIC.

So, if you see this DGX 1 server, you have got 8 tesla V100s, you have got 256 GB of memory, you have got dual 20-core Intel Xeon CPU processors. And then you have got about 40960 CUDA cores, you have got 5120 tensor cores. You have these system memory and the network right. So, this is the architecture of your DGX I.

(Refer Slide Time: 29:55)

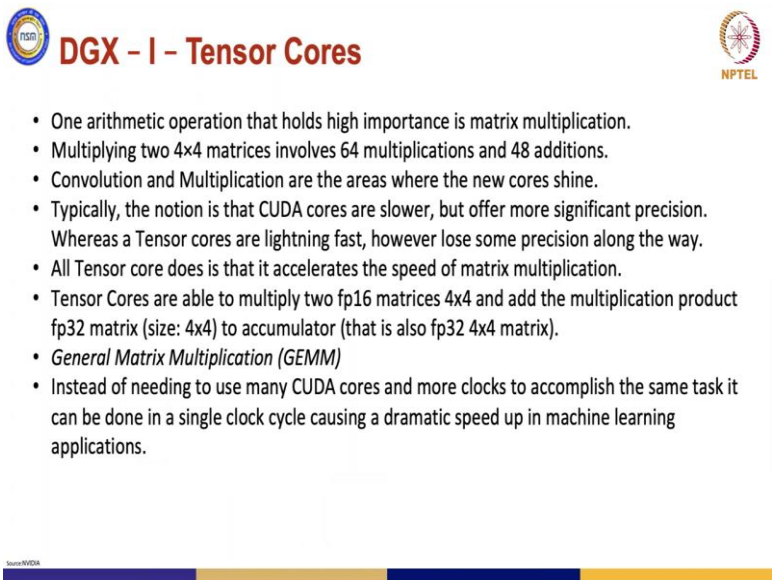


Now, let us try to go and see what is one of the very important features for these V100 GPUs. One of the very important things which is there is a tensor core. So, there is an animation which is also running on the slide, which tells you about how basically these tensor cores can process right; floating point operations.

So, FP16 or FP32 or INT4 and all so on and so forth. So, the idea here is you perform a operation of the type A * B + C, which is equal to D. So, you are trying to actually do a matrix processing ok, using arrays and then it is something like accumulation which you are trying to do in the end right.

So, this is a generic representation of this and then you can work with any type of values, it can be FP16 or FP32 and you can get the result in one of the various things ok.

(Refer Slide Time: 31:31)

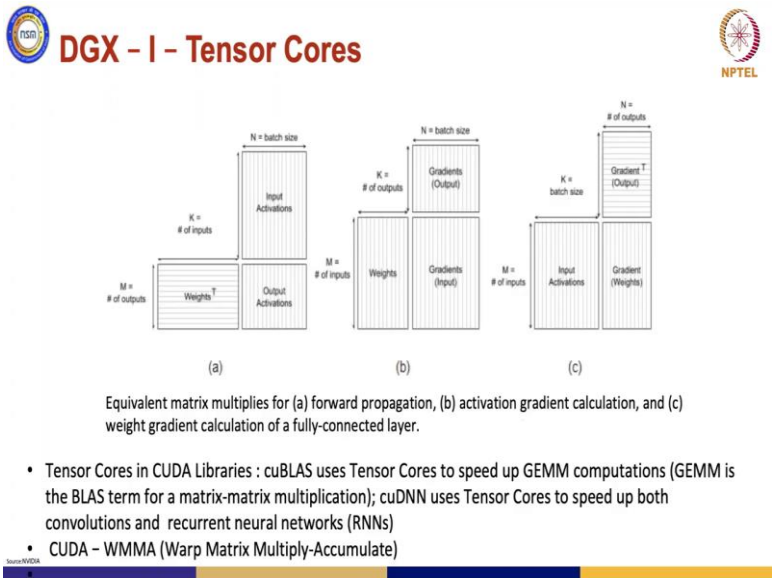


So, now why is a tensor core very important? So, if you see one arithmetic operation that holds very high importance is a matrix multiplication. So, you tend to solve lot of your problems using matrix multiplications and if you specifically see matrix multiplication of let us say a 4 x 4 matrix, it involves about 64 multiplications and 48 additions.

So, this is where this tensor core is going to be of help to us. So, the whole GPU is made up of two types of cores, one is the CUDA core and another one is the tensor core. So, notionally if you see CUDA cores would be bit slower, but they will give you lot of significant precision. Whereas, a tensor core right is very fast and along the compensating for this speed right you lose some precision ok. So, all tensor core basically does is that it accelerates the speed of your matrix multiplication.

So, tensor cores are able to multiply two fp16 matrices 4 x 4 and add the multiplicative product right and then to the accumulator. So, this is General Matrix Multiplication or GEMM. One of the things is instead of needing to use many CUDA cores and more clocks to accomplish task let us say, it can be done in a single clock cycle with lot of dramatic speedup for applications which involve machine learning this that everything ok on these tensor cores.

(Refer Slide Time: 33:46)

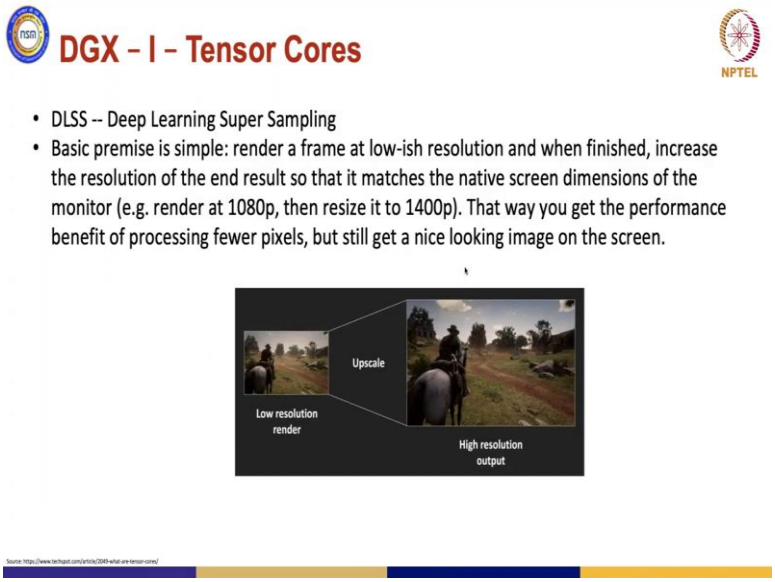


So, this is how we can think of actually trying to create ok easy to understand representations of whatever we would be using in our deep learning or machine learning type of model development. You can show it as an equivalent matrix multiplications for forward propagation or your activation gradient calculation and then weight gradient calculation, when you are trying to do it for a fully connected layer in the end.

So, you can see here that these are the matrices right, size of the matrix and then you have these output activations input activation weights and all of that put in a manner which could be easily right executed by tensor cores. So, if you see that way how basically people could use tensor cores is that you can use tensor cores in CUDA libraries, you have this cuBLAS which uses tensor core to speed up the GEMM computations.

And then cuDNN also uses tensor cores to speed up both the convolutions and the recurrent neural networks or RNNs. So, this is how you could specifically start using tensor cores for your own application development right. And then there is something which is called as CUDA Warp Matrix Multiply Accumulate or WMMA, this will help you to actually use tensor cores very fast.

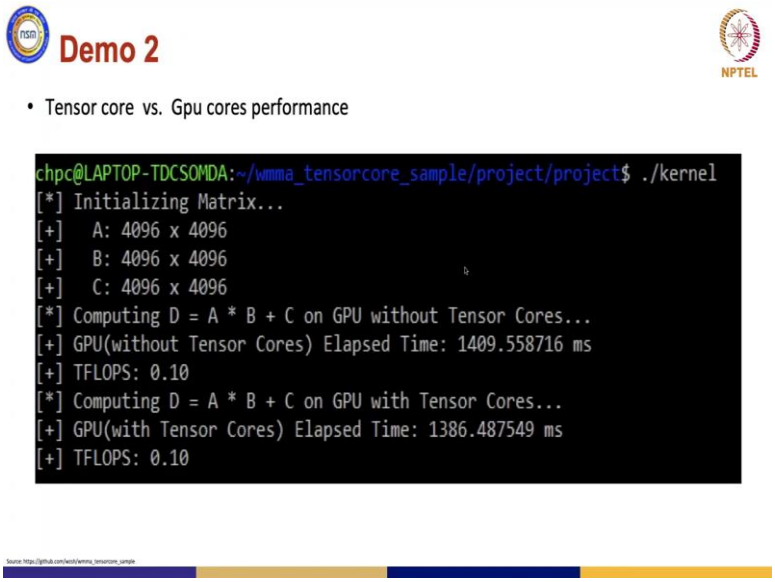
(Refer Slide Time: 35:57)



And then how these tensor cores could be used for various other applications right. So, here there is something which is a application of a sampling thing right. So, you have something called as a deep learning super sampler. So, what effectively people have tried to do is that you render a frame at a very low resolution right and then once it is finished right you increase the resolution so that matches with your screen dimensions of the monitor.

That way you get the performance benefit of processing fewer pixels, but still get a very nice looking image on the screen right. So, this is actually how tensor cores are being used ok. So, now there is a demo which we thought we will show you of matrix operation right on a hardware which has tensor cores and I will like to show it to you and this is how the result will look like.

(Refer Slide Time: 37:50)

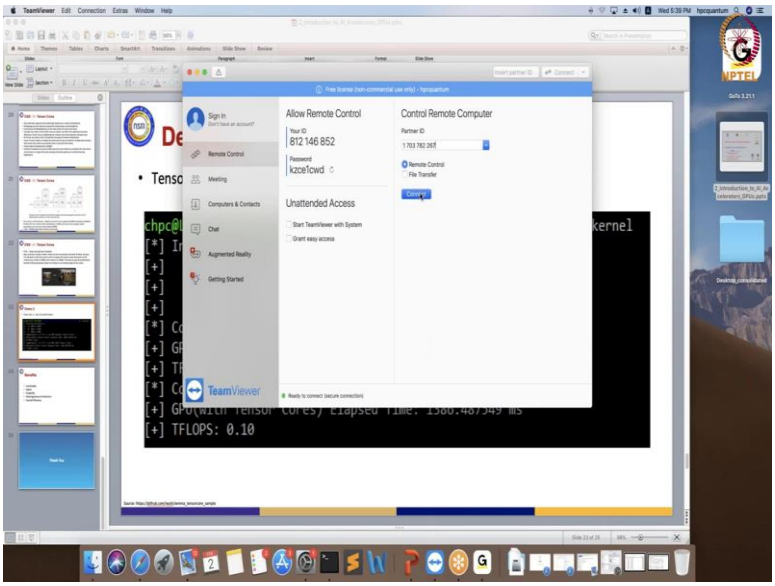


If you see this we are trying to use three matrices, 4096 x 4096 each then you do the computation which is the same thing which we were thinking of D is equal to A * B + C.

(Refer Slide Time: 38:18)

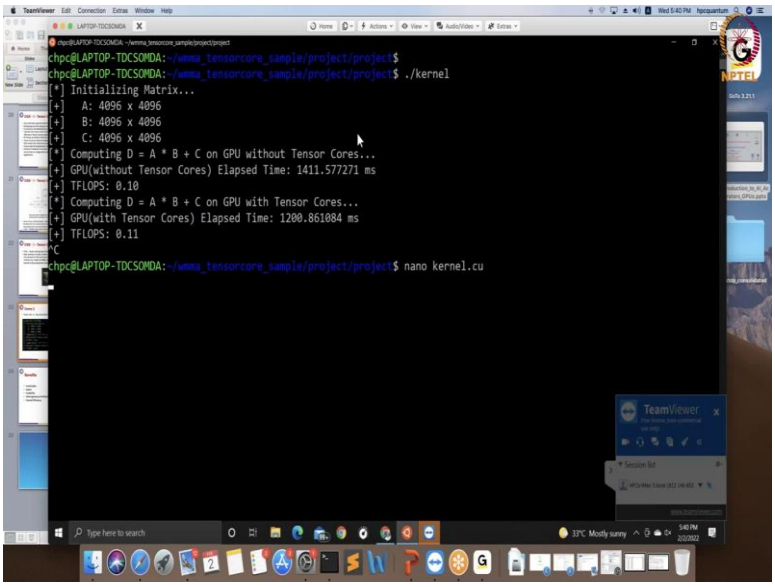
And we have done that on GPU without tensor cores and with tensor cores. So, let me just take you to that this thing and then I will just execute and discuss that program as well in brief.

(Refer Slide Time: 38:43)

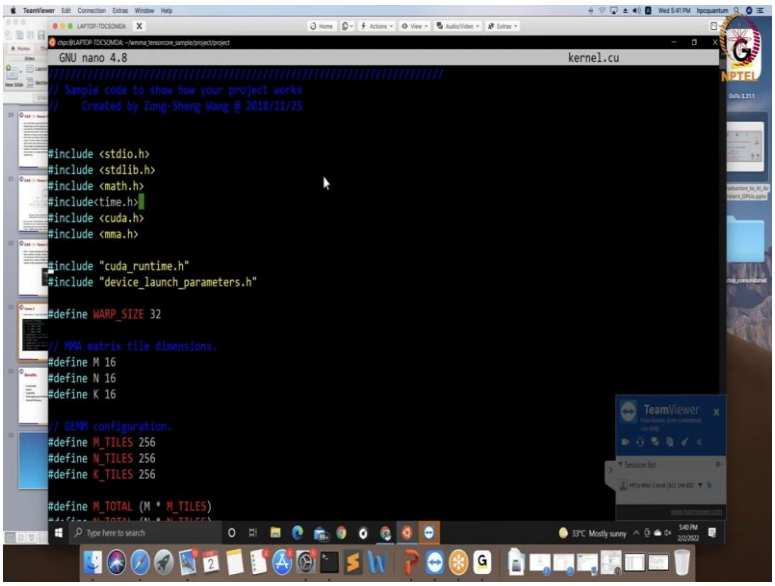


So, let me just, 1 minute, give me a sec yeah ok; so.

(Refer Slide Time: 39:28)



(Refer Slide Time: 40:34)



Student: Can you increase the font size?

1 minute sir, I will do that.

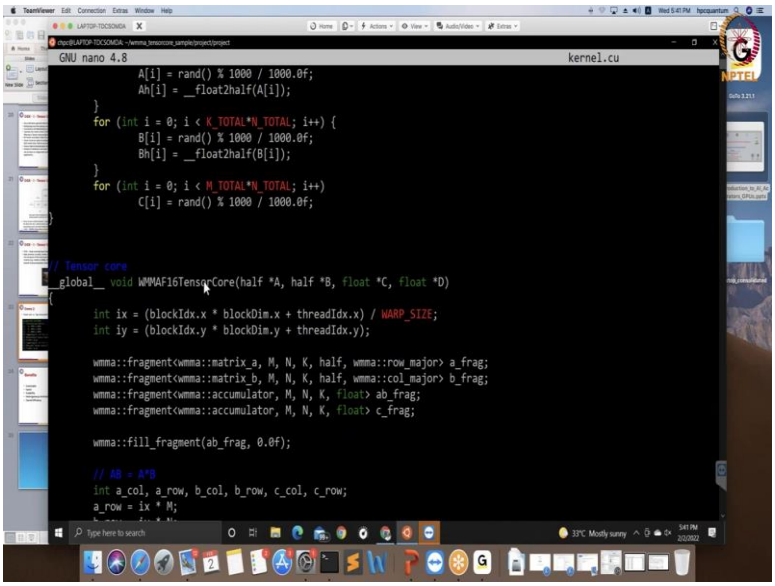
Student: Yes, this is good, this is good.

Ok, sure so.

Student: You can explain what you have fired and what is happening.

Yes, I will explain the program as well. The idea is that we are trying to compute ok, D = A * B + C with tensor core and without tensor cores right. So, let me just show the program once, there are three programs actually. So, this is a sample program which talks of trying to use something which is called as WMMA. So, this actually means warp matrix multiply and accumulate. So, the idea here is you are trying to use these matrix tile dimensions.

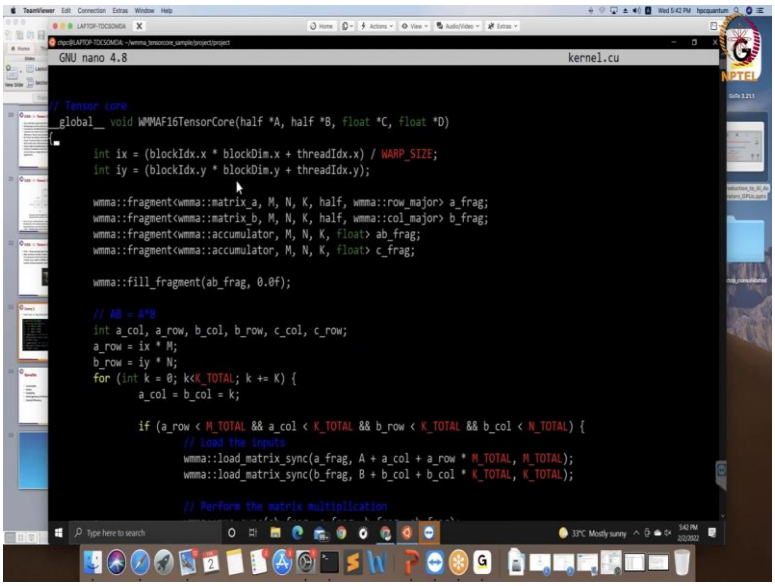
(Refer Slide Time: 41:14)



So, this is actually WMMA initialization, then you work with tensor cores using WMMAF16TensorCore type of thing instead of just CUDA cores. I will show you a GPU program also, there is a GPU program also which does not use this tensor core right.

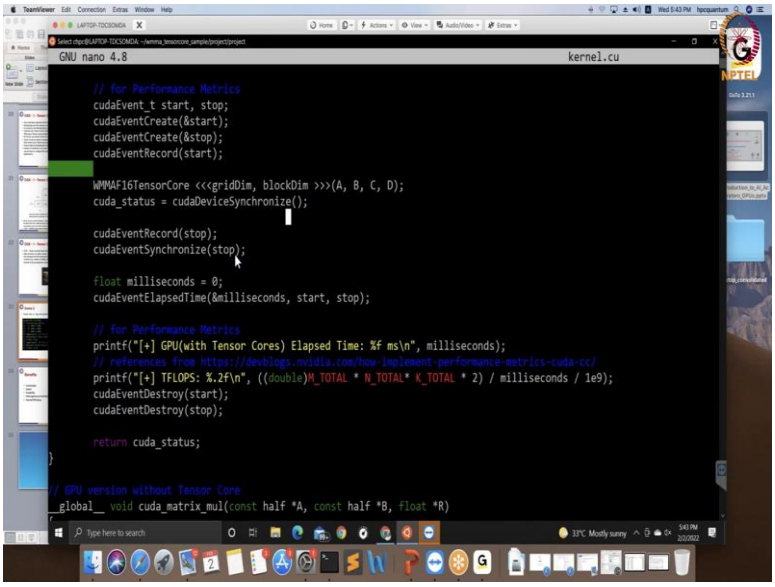
Of course, you use your blockId, block dimension threadId which gives you all of these right the same thing, but here you are trying to work with WARPs right and that is why it is called as WARP MMA and here it is basically trying to understand a very basic same thing which you try to do, but you are using the CUDA tensor thing here right. So, which I just now showed you.

(Refer Slide Time: 42:15)



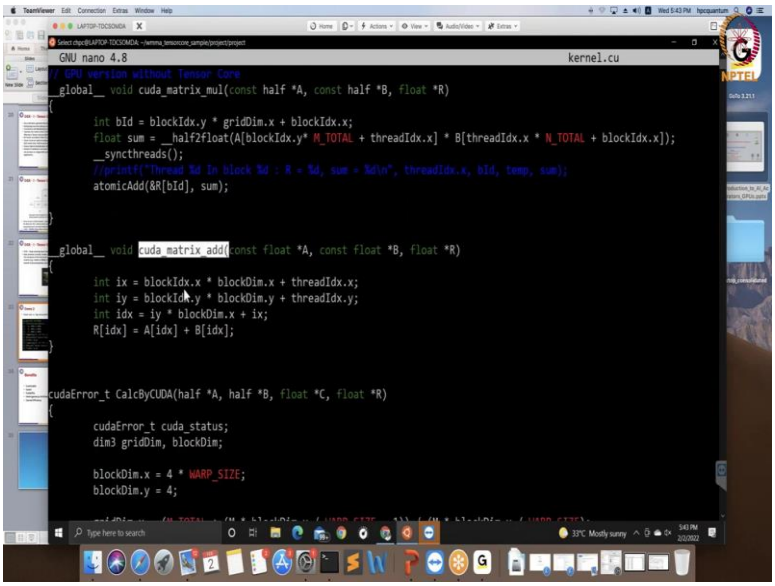
So, I will show you a GPU program also, I am not going into the details, but it is a CUDA program. So, if you can understand CUDA programming you will understand. But here the idea is to have this particular thing used ok, which is a additional thing which has been now added into your CUDA thing ok.

(Refer Slide Time: 42:40)



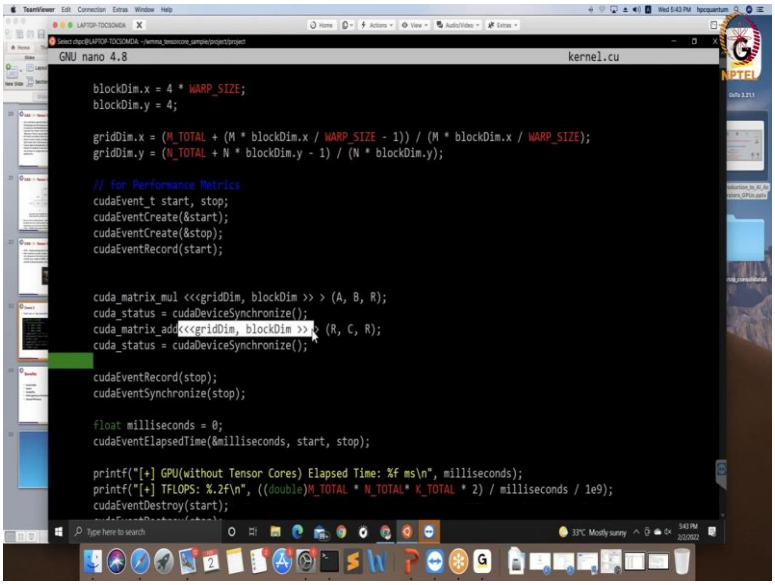
So, this everywhere uses this tensor core and then you can run it using this same grid dimension, block dimension type of stuff ok and then you synchronize it and then you do things.

(Refer Slide Time: 42:58)



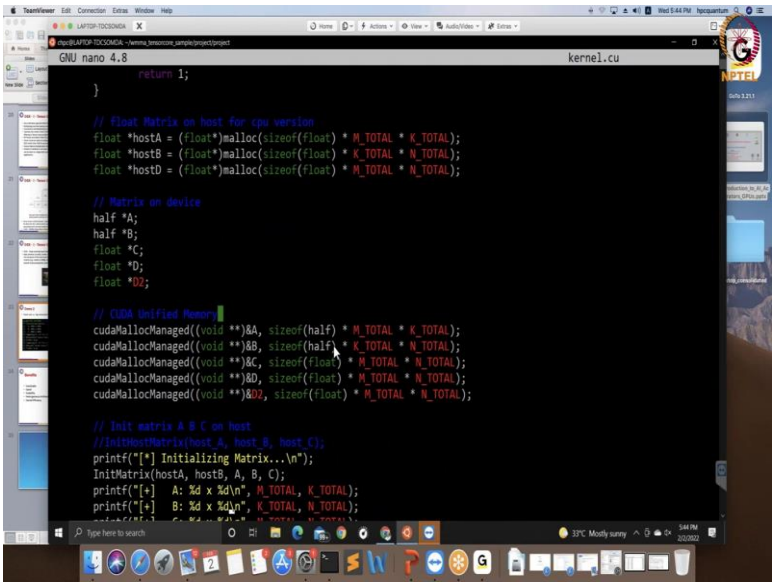
So, this is how it is and then this is a GPU program without the tensor core, wherein you just do CUDA matrix addition, use the same block Idx block dimension.

(Refer Slide Time: 43:16)



And then come up with this, then you do the same thing and then you run this using the grid dimension, block dimension type of thing and then this is again a very basic rudimentary CPU version, which talks of doing the same thing and then trying to do the multiplication and then stuffs like that; so, yeah.

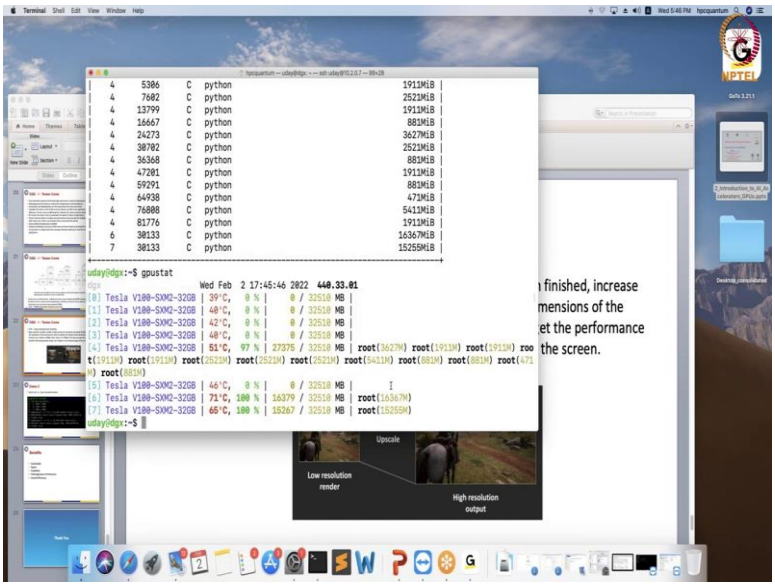
(Refer Slide Time: 43:32)



So, this is just trying to also use the unified memory and yeah, so that is it. So, I thought like this is just a idea of how basically people could use tensor cores and how they can use CUDA cores for that matter and this was just the idea of doing things. And then this was the idea of trying to tell the thing that people can start programming using the tensor cores and then we can actually see certain other things on the DGX.

And we can do all of those things as we go ahead in the days to come, we will try to actually execute lot of programs. And then and then from the benefits point of view right, sustainability is one thing, the speed is another thing, scalability, heterogeneous architecture you could utilize, then the overall efficiency right improves. So, these are certain specific benefits of trying to go in for GPU programming and then, yes. So, we can actually see some more things in; so, ok.

(Refer Slide Time: 45:10)



So, this is a thing which I had left actually so I thought I will share that with you. So, if you see the gpustat, gpustat again gives you what which GPU are going to use right, Tesla V100, 32 GB of memory, then the temperature at which it is working and then how much percentage it is being utilized with. What are the various programs which are running, how much of it is being utilized, how much memory is being utilized by which program.

So, this is a way of trying to actually see it in a different format instead of trying to use nvidia-smi. So, these are certain things which you can go on trying and then do things.