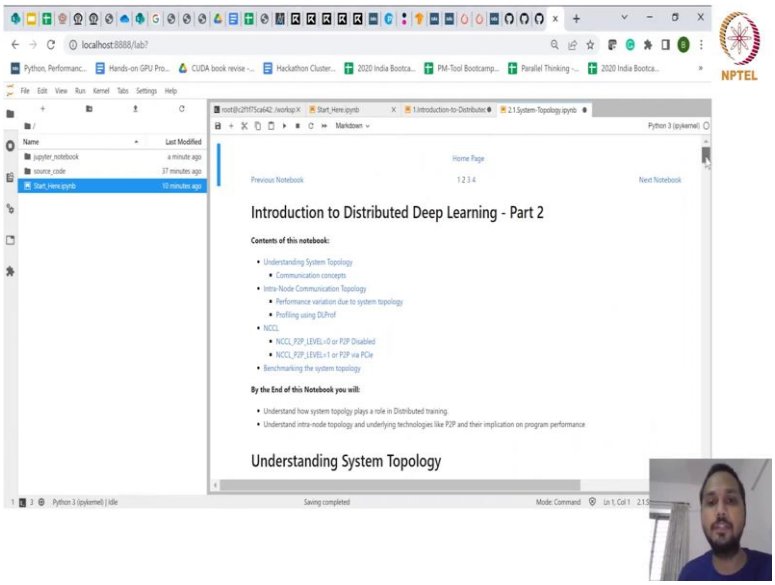
Applied Accelerated Artificial Intelligence
Prof. Bharatkumar Sharma
Department of Computer Science and Engineering
Indian Institute of Technology, Palakkad

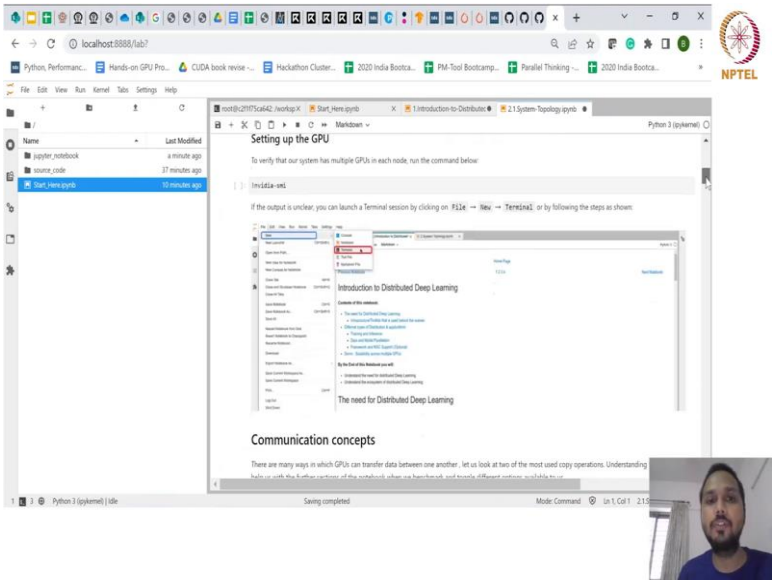
Lecture - 37
Fundamentals of Distributed AI Computing Session 2 Part 2

(Refer Slide Time: 00:17)



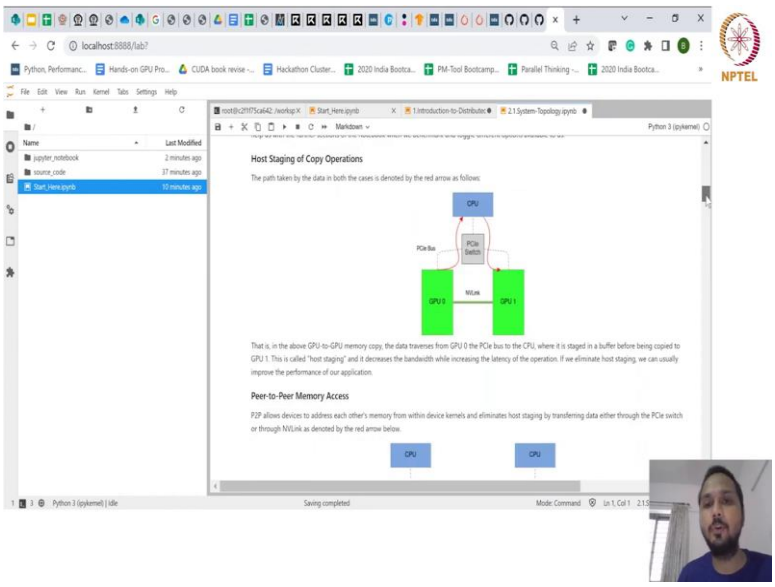
So let us look at the part which we had discussed last time, which is the system topology part and the impact of it right.

(Refer Slide Time: 00:29)



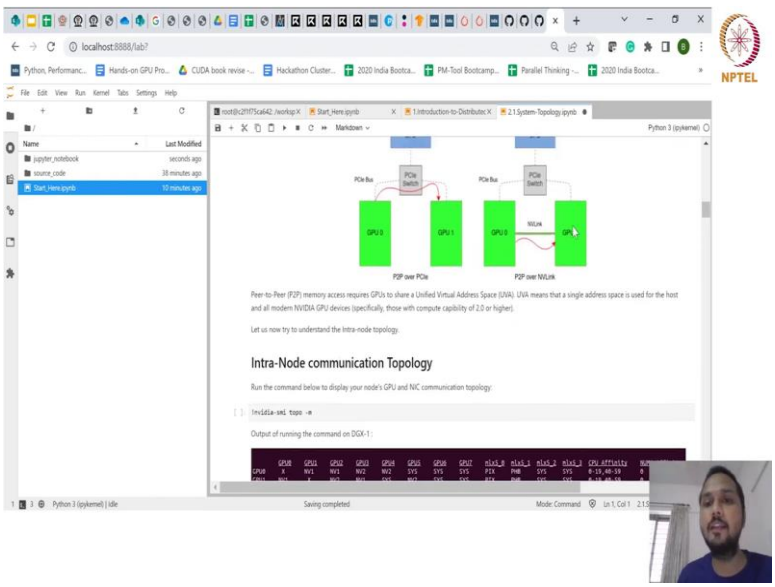
So, I already talked to you about the GPU V 9 NVIDIA. So, and we saw there are 8.

(Refer Slide Time: 00:32)



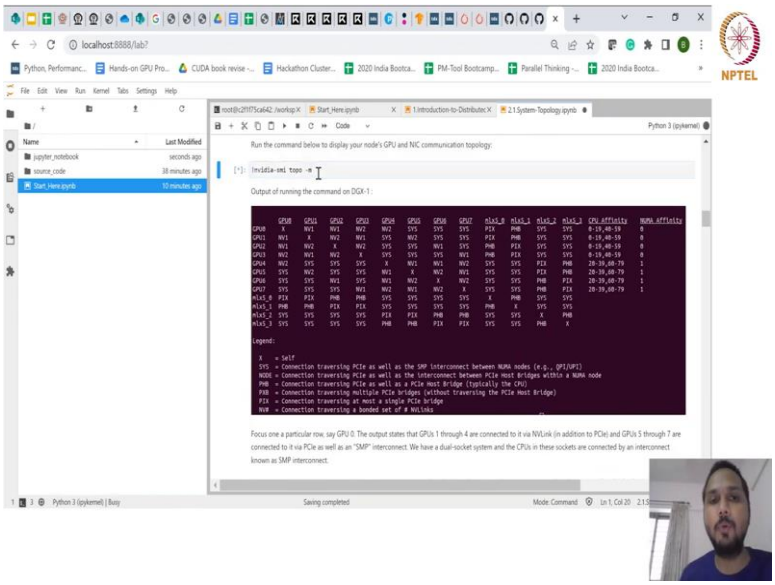
I talked to you about the concept of peer to peer transfer and in case I do not have it what happens. So, in case GPU 0 and GPU 1 do not use memory links like NVLink and they are not directly connected in the system, in that case if there is a host staging which happens which means from GPU it goes to the CPU and then from CPU it goes back to the GPU 1, that is going to reduce your throughput and increase the latency reducing into reducing your overall scaling.

(Refer Slide Time: 01:06)



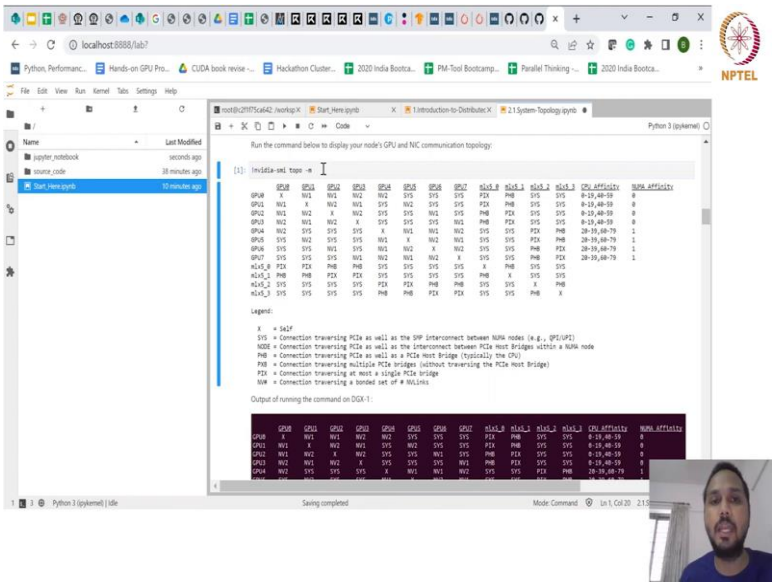
In the peer to peer transfer environment, you should be able to directly talk between the 2 GPU and you do not need to do a hop from the CPU.

(Refer Slide Time: 01:15)



Now, the question that you might have is how do I know this. Now, already GPU provides you tool to do that.

(Refer Slide Time: 01:21)

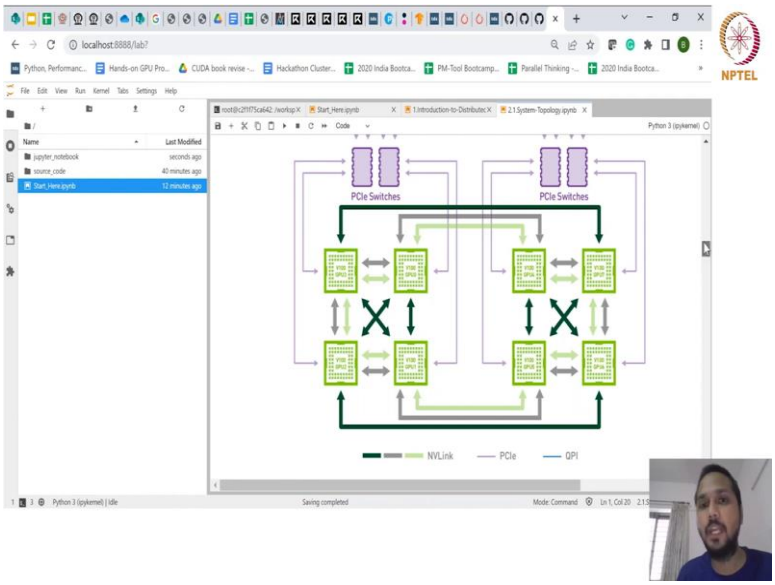


So, here I am running a command called as nvidia-smi topo -m, topo stands for topology, m stands for metrics. So, I am telling it to tell me more details about my system and how those GPUs are connected to each other. So, in the system where I am currently as you

can see our GPU 0 is connected to GPU 1 via NV1, while GPU 0 is connected to GPU 3 via NV2, right.

So, and you can see here what are the definitions of NV. NV is the connection traversing a bonded set of number of NVLinks, right. So, how many NVLinks you are actually traversing it through.

(Refer Slide Time: 02:05)

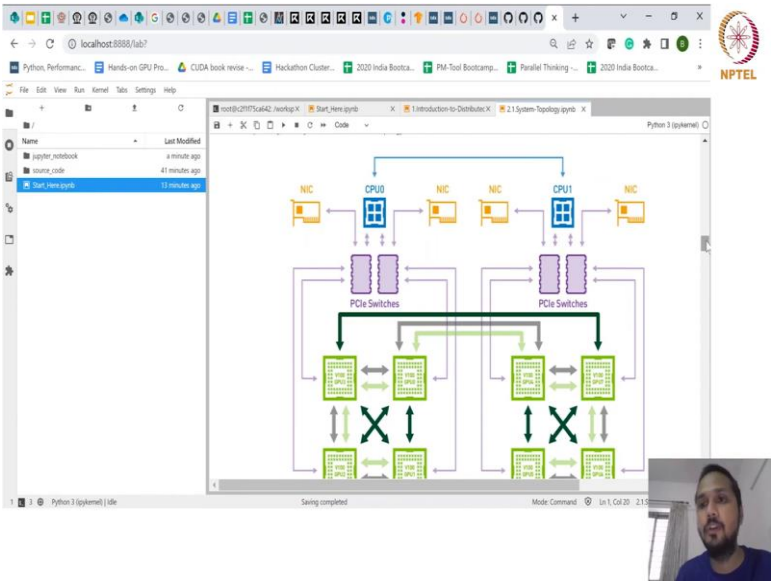


To give you an example of the topology which we showed last time in the DGX, if you can see here GPU 0 and GPU 1 is connected via one NVLink. So, hence NV1, but if I look at GPU 0 and GPU 3 they are connected to each other via 2 NVLinks which means if one NVLink is giving me a speed of 25 Gbps between GPU 0 and GPU 1, at the same time between GPU 0 and GPU 3 I have two NVLinks practically giving me 50 Gbps.

So, it is increasing my transverse between these two GPUs, while when I go from GPU 0 to GPU 5 I do not even have a NVLink. So, I am going to take a path of four staging right and that is what is kind of replicated here. From GPU 0 to GPU 1 and 2 I have one NVLink, but when I go to GPU 3 I have two NVLinks attached, which is very good because I basically get more speed, the more number of links of NVLink the better it is for me.

But if I look at GPU 5 I said that there is no NVLink connection and we are going via system, what is the definition of system? The system is I have to traverse via PCIe as well as between the NUMA nodes.

(Refer Slide Time: 03:34)

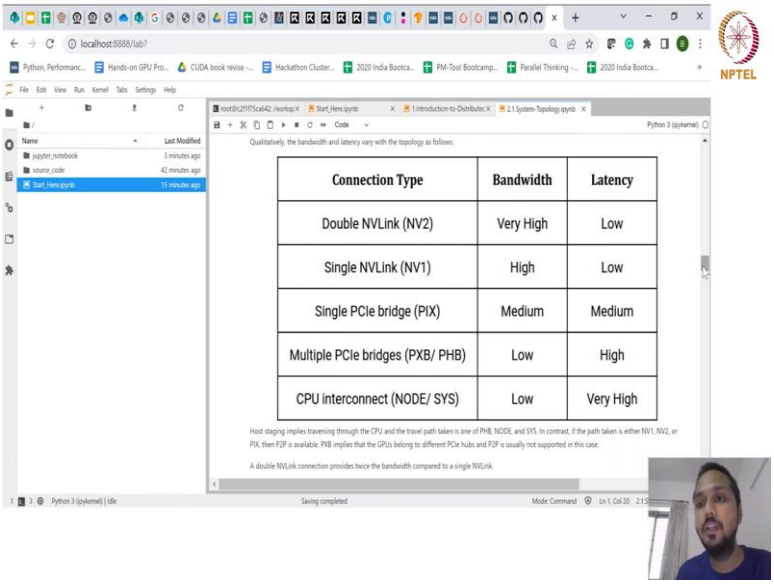


Here there are two numa nodes, CPU 0 and CPU 1 and I have to traverse from GPU 0 to CPU and then transfer to the next numa node, which is CPU 1 and then come back to GPU 5. So, the interconnect between 0 and 5 is kind of weakened here right and for different systems you will see a different topology and based on this topology you will see different scaling numbers when you use multiple GPUs on the system.

And if you want to go across again you would see that there are InfiniBand or Mellanox switches which are there. So, GPU 0 and Mellanox switch is connected via PIX and PHB, while the Mellanox switch 5 2 is going again via system, which means if GPU 0 wanted to transfer it data to another node which is not shown here, to another node.

And if it was using this NIC card again it will transfer to the CPU from CPU 0, it will come to CPU 1 and then it will go via NIC to the another machine, which is again not a very good scenario. Now, this kind of gives you the overall topology of your system.

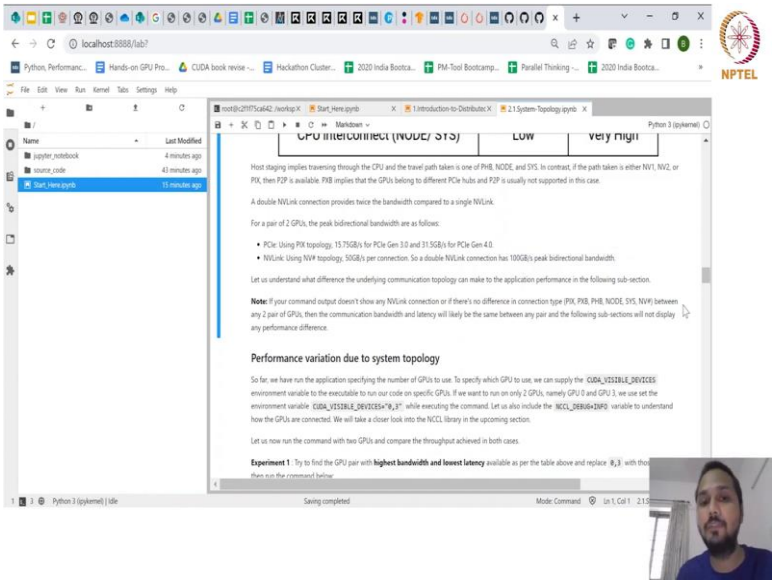
(Refer Slide Time: 04:59)



So, whichever system you run it on you would be able to see the topology for your own system. So, as I said you can basically think of it like this, that you can have multiple connections and each and every connection will have a particular thing like NVLink 2 is going to have very high bandwidth and low latency, NVLink 1 will have a very high bandwidth higher than the PCIe, but it will be lower than NV2 right. But the latency will be still low.

So, the; so based on what bridge or what topology is there on your system you will end up seeing a different chart of bandwidth and latency. Idea is to have higher bandwidth and lower latency to have better scaling results ok.

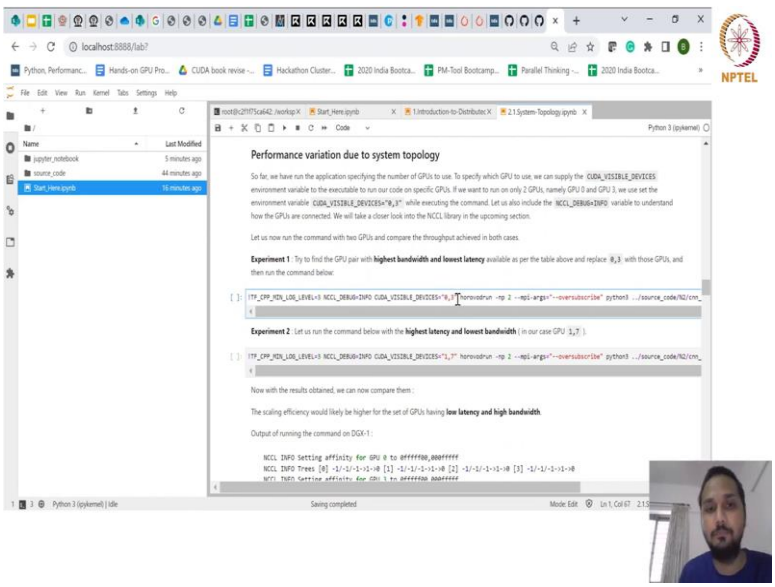
(Refer Slide Time: 05:40)



So, you can see here some of the numbers which are there, like for PCIe it gives you maximum 15 Gbps on a PCIe Gen 3 right and while the NVLink you can go up to 50 Gbps right. If you have a double NVLink you can go up to 100 Gbps of bi directional bandwidth.

So, you can see the differences when you have a system with NVLink and without the NVLink. Now, how does it matter is something that we are going to see here, in this particular section when we are talking about performance variations.

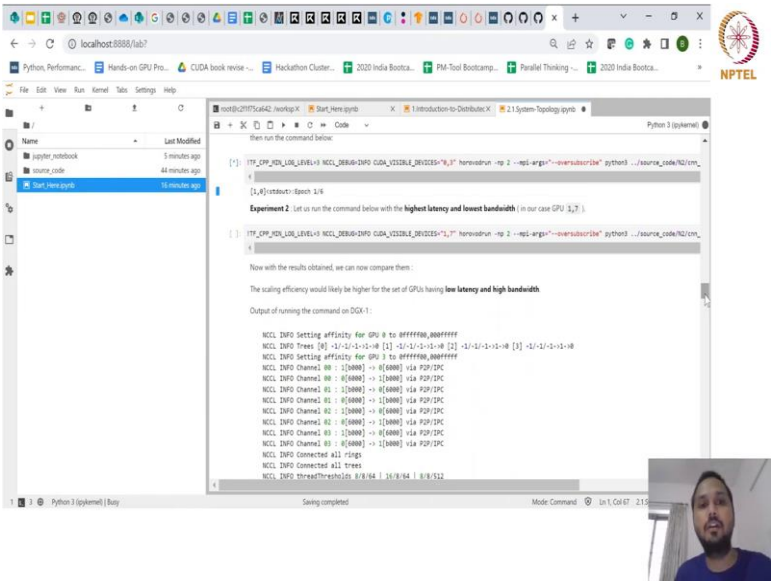
(Refer Slide Time: 06:12)



So, again we are going to do the same thing previously, but I am going to enable more information about the communication using NCCL. So, you can see here I am setting the NCCL flag debug so that I can get more information about the way I am transferring the data behind the scenes using NCCL.

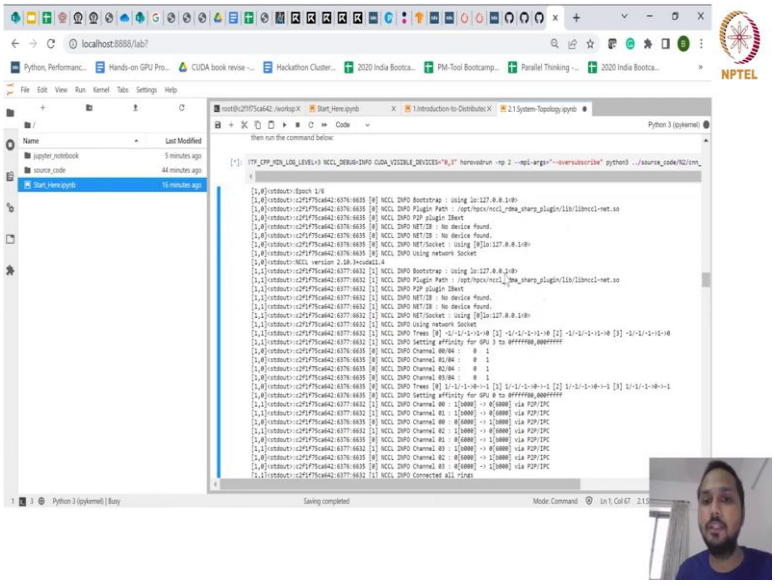
So, even though I am using TensorFlow and I am using Horovod as the framework, I am still going to utilize NCCL as I told behind the scenes, but this flag will help me in identifying when I do communication between 0 and 3 in a multi GPU environment, am I using NVLink or am I not using NVLink and what kind of a numbers I can expect.

(Refer Slide Time: 07:05)



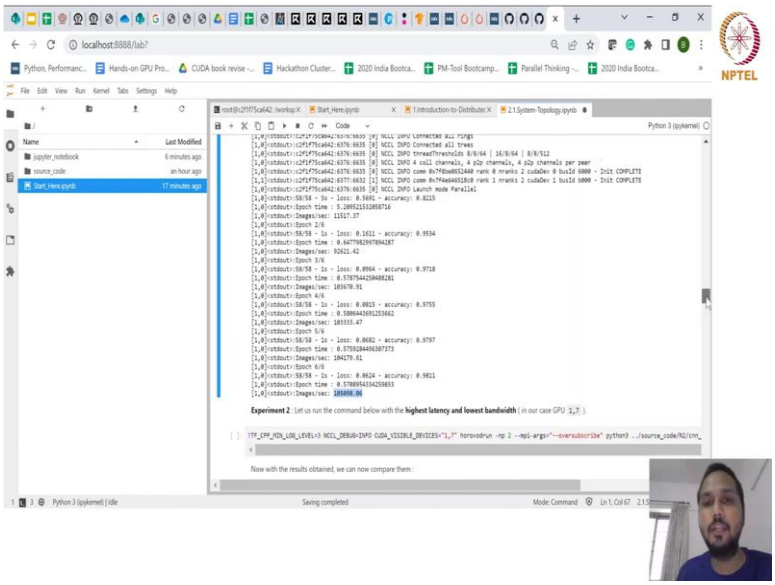
So, just give it a couple of seconds and we should be running.

(Refer Slide Time: 07:14)



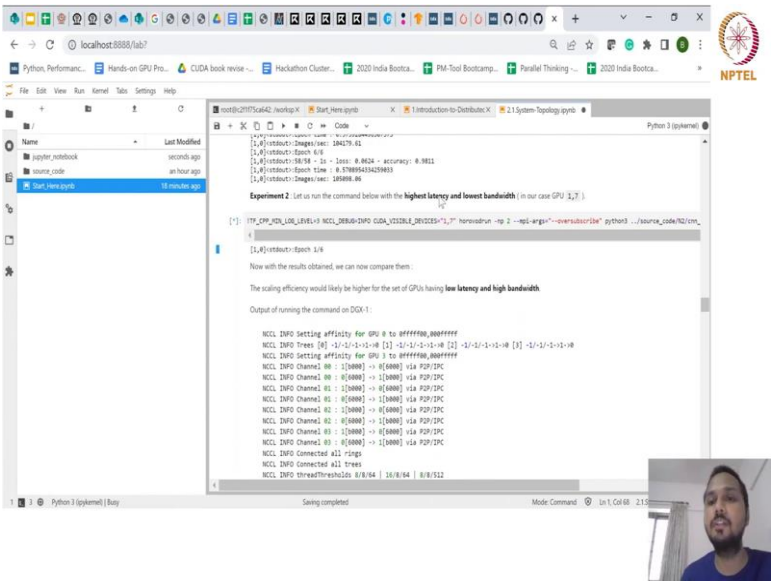
So, you can see here it has started giving me more information. The first thing it is telling me is that I have started using NCCL what is NCCL and all we will talk about it in a bit. But NCCL is something which I am using and it gives me information, like it is telling me that for NCCL between 1 to 0, between GPU 1 to GPU 0 which is the 0 and 3 GPU I am using peer to peer communication. I am using peer to peer communication right, it states it when you are actually enabling the flag of the debug info for NCCL.

(Refer Slide Time: 07:55)



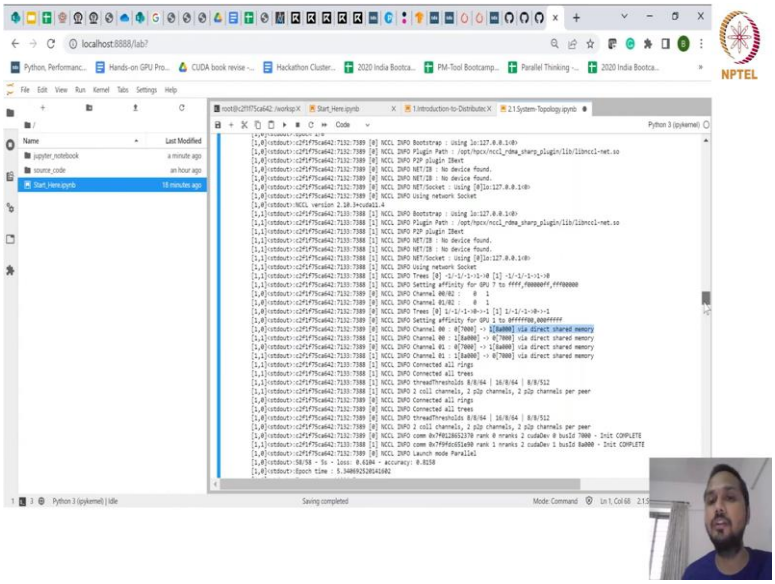
And by the end of it, you will get certain number of images per second, here it is around 150098 and images per second. So, again we are running a fashion MNIST only on 2 GPUs here. The same thing now what we are doing is instead of doing it via 0 and 3, I am going to use GPU 1 and 7. And if you remember earlier 1 and 7, if you see 1 and 7, here they are not connected directly via any kind of NVLink right.

(Refer Slide Time: 08:34)



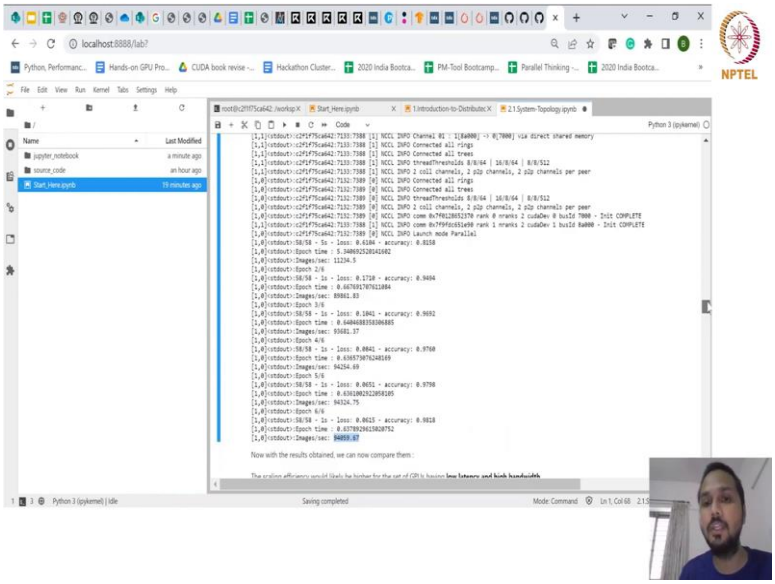
So, let us see what does NCCL tell us when it is trying to communicate via the between GPU 1 and GPU 7 for the same exercise which we did earlier when we got 150000 right, some number.

(Refer Slide Time: 08:50)



So, instead of showing that between 0 and 1 I am using peer to peer, now as you can see here it is stating that I am using something called as the shared memory and I am not going via peer to peer. So, it is stating that I do not have any direct link between these two GPUs and hence I am going to revert to using certain sort of shared, where I have to transfer the data say via host staging if required.

(Refer Slide Time: 09:16)

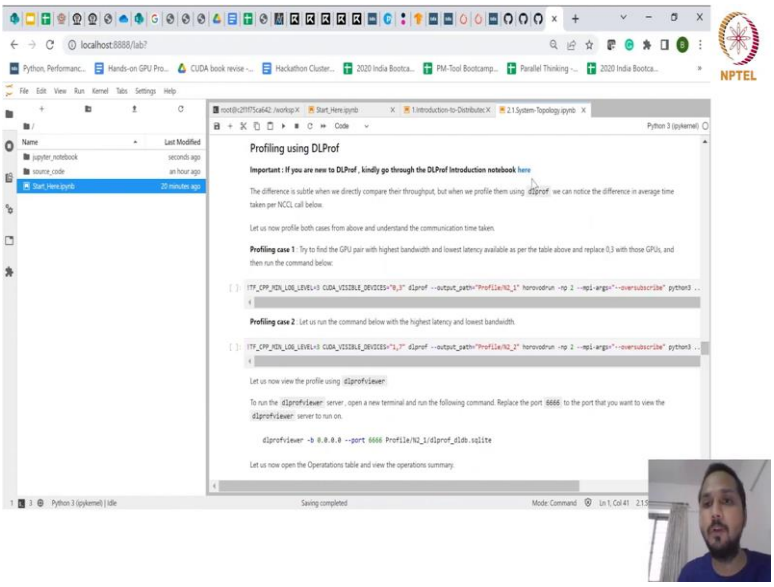


And you can see here that my images per second has reduced from 100000 something to 94000. So, by having NVLink versus not having NVLink, you can see here that the

number of images per second that I could process has in fact reduced, right. So, this is what we can see. So, again repeating we enabled a particular flag.

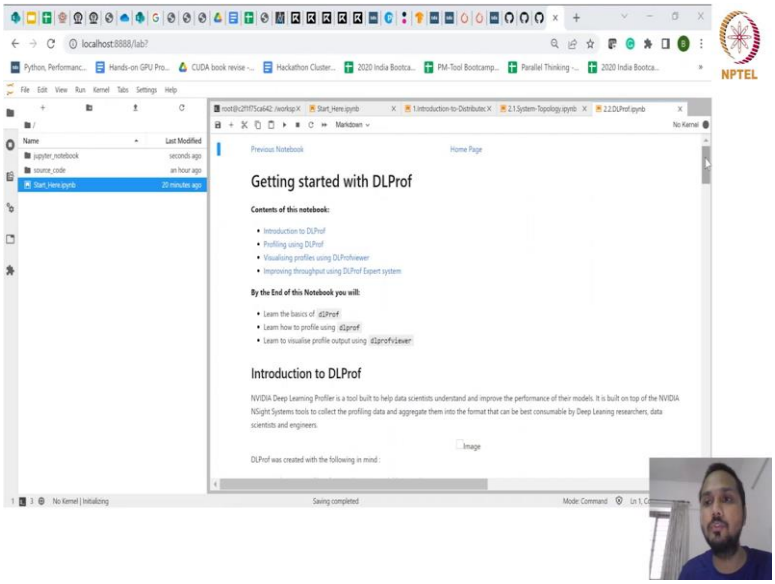
So, I am running using Horovod, I am running on 2 GPUs, but I decided to use GPUs where I knew there is a NVLink connection versus there is no NVLink connection, I am running it for fashion MNIST and I enabled a particular flag called as NCCL debug info so that I can see what NCCL library behind the scene is choosing, when I start communicating across the two GPUs and the impact of it as well, right.

(Refer Slide Time: 10:13)



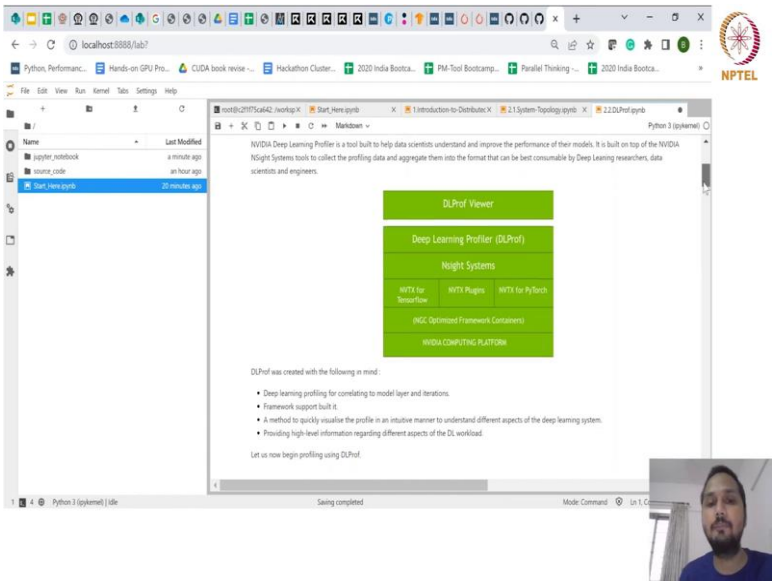
Now, you might again start talking like ok this is good, but I will also introduce you to tools which you can use for actually finding out more details of how well your model was actually running on a GPU. So, NVIDIA has the tool which is again a free tool which you can use which is called a DLProf Deep Learning Profiler.

(Refer Slide Time: 10:41)



Now, DLProf is the one which will give you more information about when you run your model across the GPUs, how when you are utilizing the GPU.

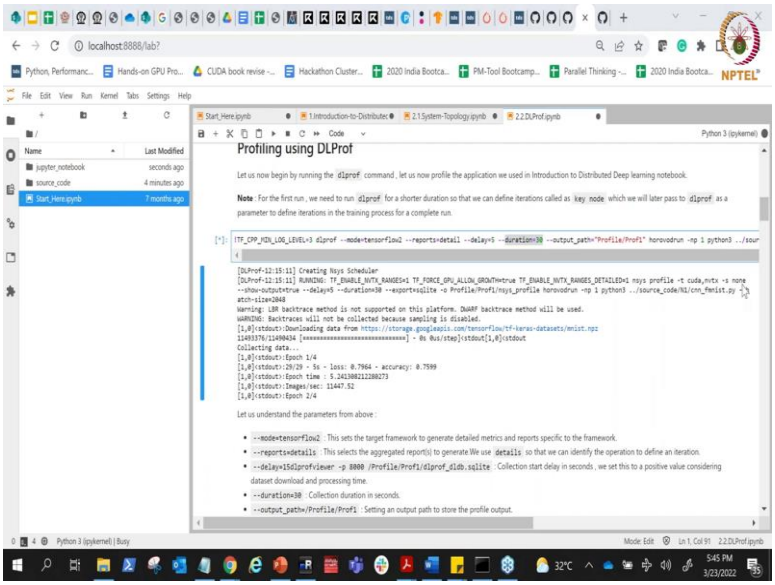
(Refer Slide Time: 10:48)



So, NVIDIA deep learning profiler is the tool which is built specifically for data scientists to understand and improve the performance of their model. So, you can see here it is built obviously over the NVIDIA computing platform, it runs in a container environment and it can run on different NVTX plugins for TensorFlow PyTorch and all

and it basically is built over the NSight system itself and we are going to see a demo of it.

(Refer Slide Time: 11:17)



So, we ran these session earlier, but let us try to run it once more. In this particular case you can see here I am using a tool called a dlprof. And that dlprof I am giving it certain parameters like I am telling it that I am going to run the dlprof the profiling and I want to gather the TensorFlow statistics for distributing the work and we will go back to the theory in a bit.

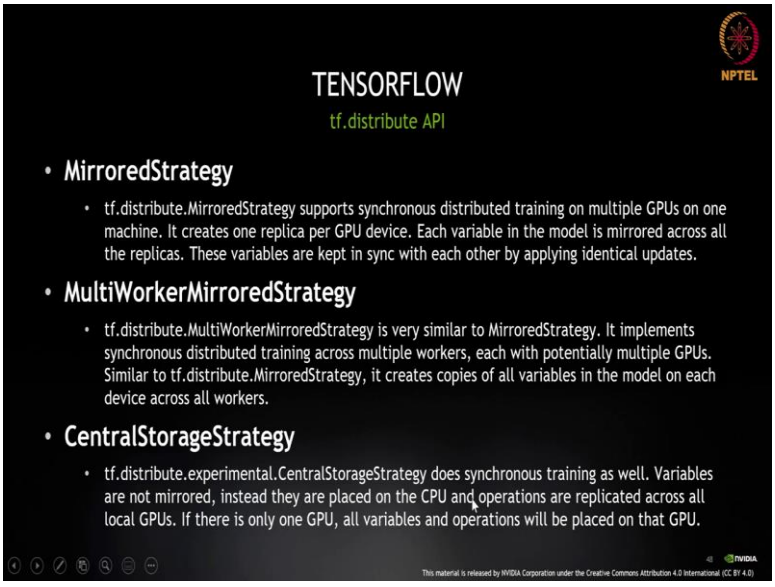
(Refer Slide Time: 11:40)

(Refer Slide Time: 11:46)

Like TensorFlow has its own way of distributing the work. The TensorFlow uses an API called as distribute API. And in this distribute API you can choose different things and their different strategies, like you can choose synchronous communication, asynchronous communication you can also choose the platform it can run on multiple types of distribution.

Like you can run it on a pure CPU nodes across or you can also do it on Google TPU also or you can do it on the GPU.

(Refer Slide Time: 12:20)



You have to define the strategy for say a TensorFlow version. In the TensorFlow if you choose mirrored strategy for distributing the work, it will basically indicate that you are going to use synchronous distributor training on multiple GPUs, but on a single machine which means you are going to run or distribute the work across multiple GPUs, but within the same machine it will not go across different nodes.

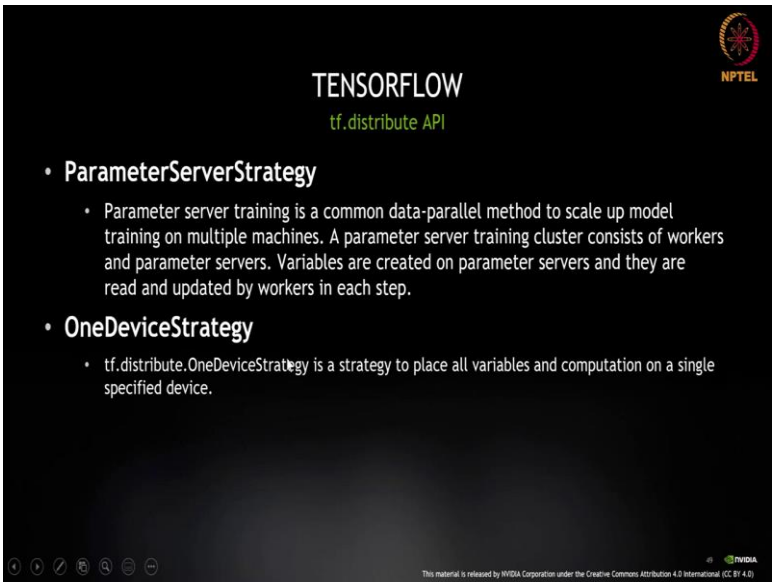
What it does? It actually creates one replica per GPU. So, we can see here it is kind of doing data parallelism and each variable in the model is mirrored right and they are kept in sync, because this is asynchronous programming environment. Multi worker mirror strategy is the same very similar to mirrored strategy above, but you have multiple workers each potentially having multiple GPUs.

So, it is almost similar to the mirrored strategy and it only works within the same node, but you are going to have more number of workers targeting multiple GPUs. You also have central storage which again does synchronous training, but there is a difference. In this case all the variables are not mirrored across different GPUs, instead they are all kept on the CPU.

So, the CPU will have the variables and whenever GPU go through the update the go through the forward propagation and back propagation and they calculate the gradient they basically communicate and update the CPU version of it which is then taken by the other GPUs as well.

So, it was slightly different environment. Where are you keeping your final state of the variable for communication is different in mirrored strategy and central storage.

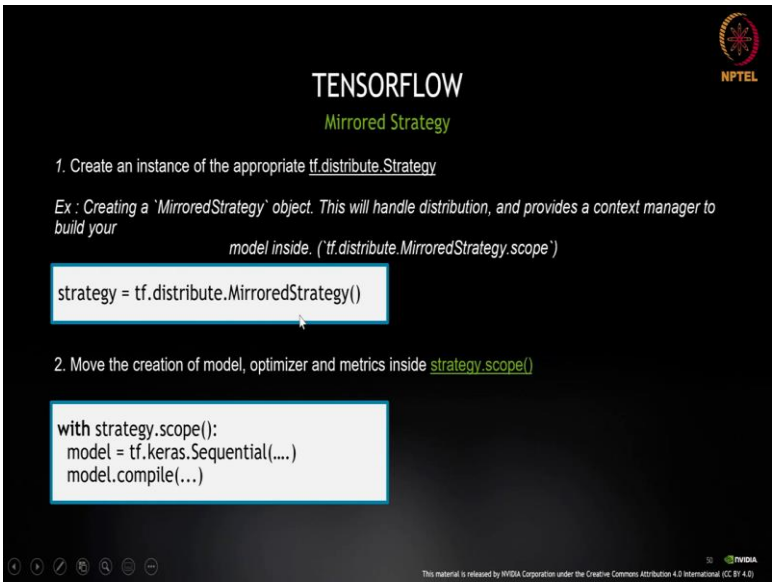
(Refer Slide Time: 14:09)



You can also take parameter server strategy which we talked sometime back right. Parameter server is a common data parallel method to scale up the model training, it is basically working on the concept of a synchronous program, where a parameter server training cluster. It consists of multiple workers and parameter servers.

The variables are actually kept on a parameter server and they are read and updated by each and every worker. So, each and every worker does not need to talk to each other, they just talk to the parameter server as well and you can also define other kinds of strategy like, one device strategy also right.

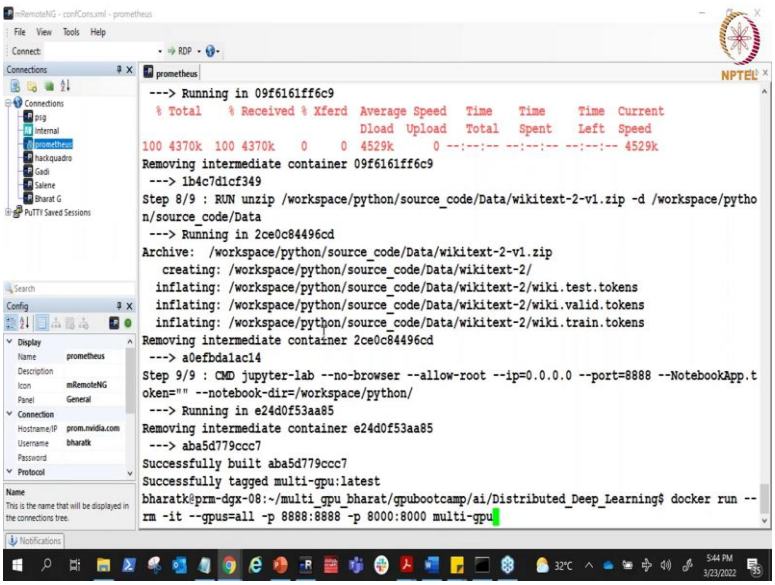
(Refer Slide Time: 14:45)



So, if you were to use TensorFlow mirrored strategy in that particular case you would have tf.distribute.MirroredStrategy.scope. So, that you are saying tf.distribute.MirroredStrategy(). And you create the model, whenever you are creating the model the model creation happens within the scope of that strategy.

So, you can see here you are saying with strategy.scope and inside that you are defining your model. So, the model is defined inside the scope of the strategy that you have defined. So, that is how the work of the TensorFlow works.

(Refer Slide Time: 15:26)

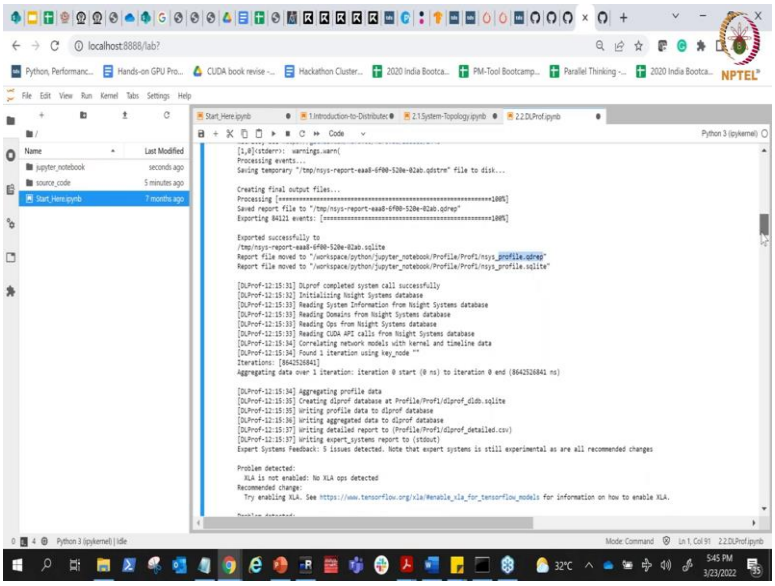


So, luckily my machine is kind of up and let me just try to do the docker run off so that my machine brings up again hopefully. So, if it is running the Jupyter notebook and container if my network is stable I should be able to bring you all back to the same time here yes.

So, we are in the DLProf lab only and I was taking you through the process of creating this report. So, as I said just to come back we have the dlprof, which is the profiler running I am telling it to use the TensorFlow and I am telling it to collect it for certain duration right.

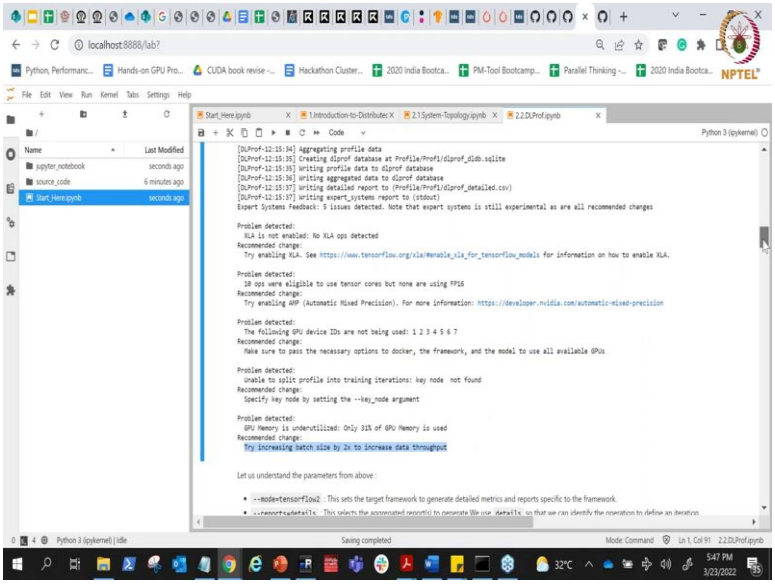
And this will give me more details about my overall thing. So, in the end once you have finished the model training, in the end if you see it will dump certain outputs.

(Refer Slide Time: 16:37)



It is saying that it has created the profiling output and it is present in a particular location. The location is kind of listed here which is nsys _profile.sqlite database and it also stores it in our database called as qdrep file right.

(Refer Slide Time: 16:54)



Not just that, in fact, in the end it will throw out after you have trained your model, it will throw certain issues if you read the issues, the issues are kind of highlighted. What it is saying is I have detected 5 issues and note that the experts system is still experimental and you these are hints given to you.

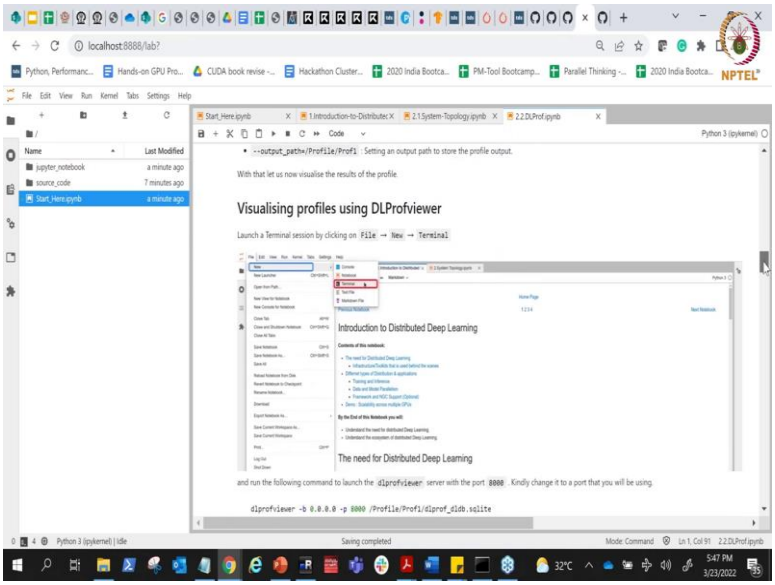
And you can check whether it is doing certain things or not, is it doing a XLA or not and you have done a session where we were talking about XLA, but let me show you the last session there was a particular lecture on AMP, whether automatic mixed precision you can use or not.

So, we can see here that it is telling that you are eligible, 10 operations were eligible for tensor core and it can use FP16 using automatic mixed precision, but we have not enabled it. So, you can improve the accuracy, you sorry you can improve the performance or the overall throughput by enabling mixed precision, try doing that it is one of the problems.

Another thing it has listed is that we observed that there are 8 GPUs on your system and you utilize only one of them which is GPU ID 0, while 1 to 7 are not being utilized. So, ideally you should look at distributing your work right. And there are many more things the another problem that it is highlighting is with the current batch size the GPU is underutilized. It is only utilizing 31 percent of the GPU memory.

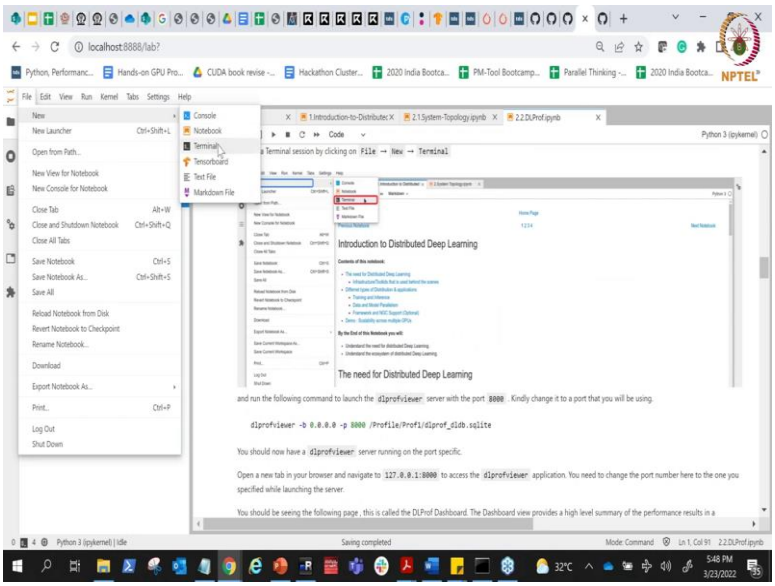
So, you can actually increase the throughput of your system by two times by changing the batch size. So, you can see here by running it via profilers, you can actually get much more details of your system right, of your overall thing and now that is what is very very critical to understand right. So, the profiler can give you more details without you having to know about it.

(Refer Slide Time: 18:53)



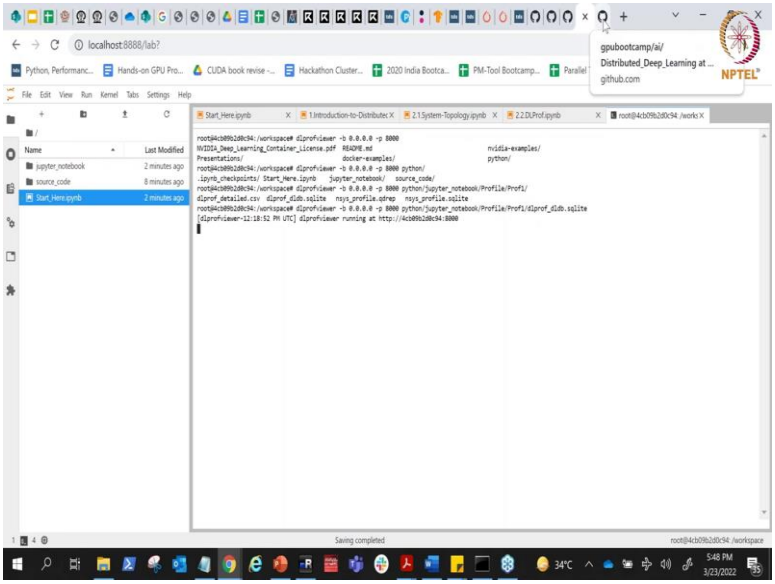
We can also visualize it in form of much more detail.

(Refer Slide Time: 19:01)



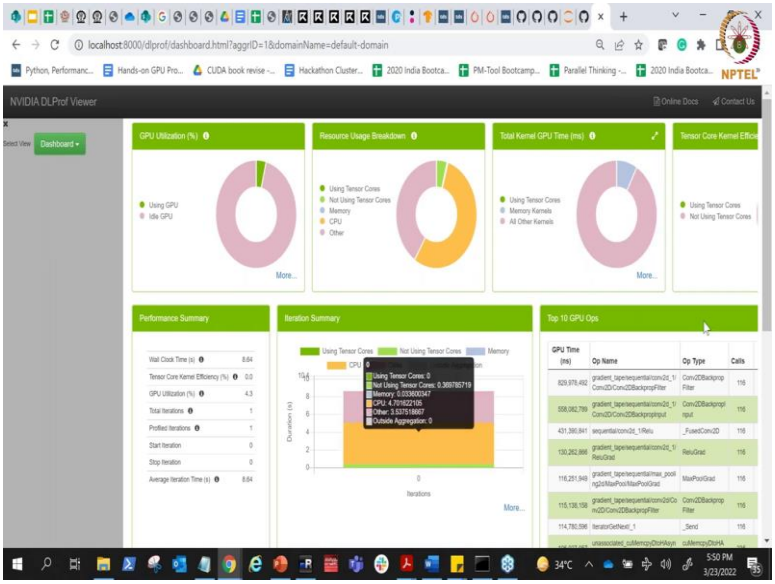
So, what I am going to do is that I am going to open New, Terminal file; New, Terminal.

(Refer Slide Time: 19:04)



And I am going to run a particular command as a server so that I can visualize the details. I am going to say dlprofviewer -b 0.0.0.0 -p 8000. So, I am hoping it at a port 8000, so that I can visualize that port 8000 and then I am just looking at python notebook profile of one and I am going to visualize the profile, the profile input which was created just now.

(Refer Slide Time: 19:58)



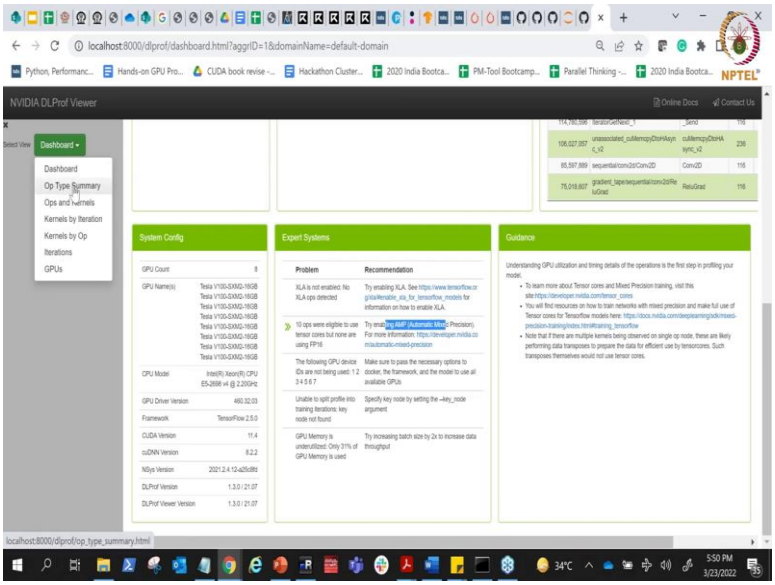
So, let me just run it and I am going to open now the another terminal and visualize my output using not just command line, but also via the via a visualization tool. So, you can

see here the dlprof whatever it showed on the command line, you can actually see it inside the visualization tool, it is like a client server model.

So, you can see here when I ran my system it is saying my GPU is only utilized actually for this much of time in my overall simulation. You can see here it is telling me that I am actually running it only, the GPU is only utilized for this much of time rest all it is idle it is not doing anything and it is also telling me that I am whether I am using tensor cores or I am not using tensor cores.

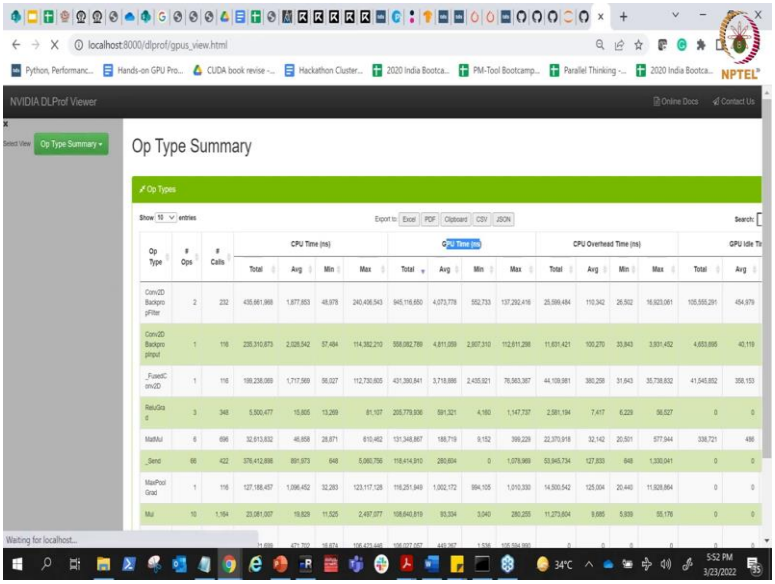
How much time I am using the CPU and how much time I am not doing anything or I am just waiting. So, you can see here that actually I am not doing a very good job at it, the total Kernel time spent on the tensor core is almost none. So, I am not using any kind of a tensor core operation right and also it does you can see it in different forms, in terms of iterations and all like for every iteration how much duration was spent and much more than that.

(Refer Slide Time: 21:17)



It also highlights the part of how to use different features, like how to use say automatic mixed precision and it also has linked to the definition of what how you can enable it or what is the definition of automatic mixed precision. So, you can enable it and click on it and get more information about automatic mixed precision. This is like one of the view which is the dashboard view.

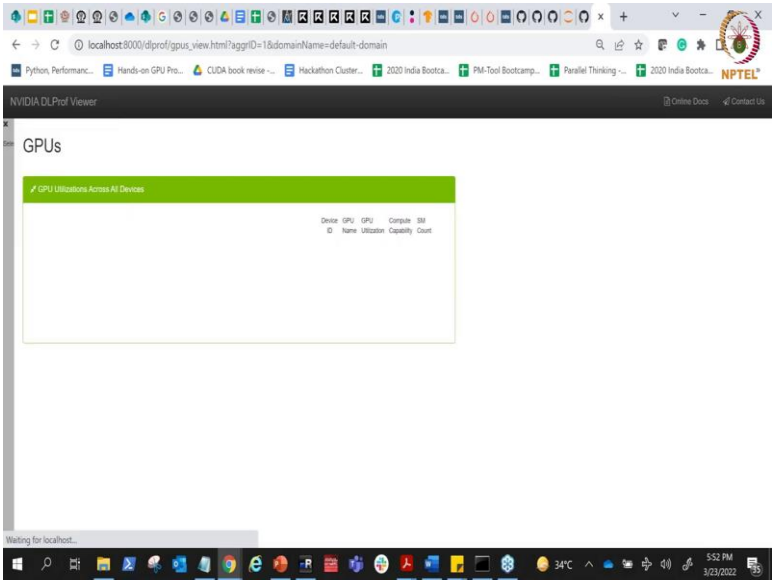
(Refer Slide Time: 21:47)



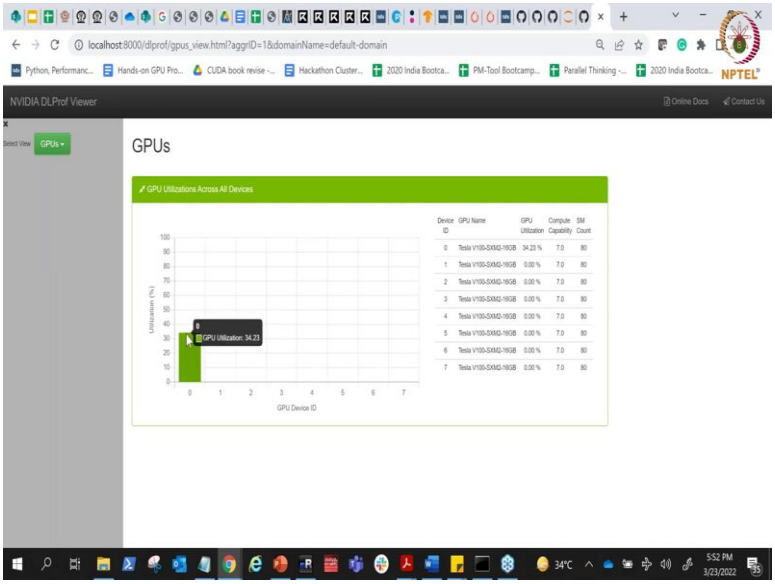
You have other kinds of view as well like operation types. So, I was showing you one of the views of dashboard and now we have another view which is of type summary, this is much more detailed view which gives you different types of operations which are done in your network and how much time we spent where.

So, you can see here it is telling there is an operation which is called as convolution 2D back propagation, convolution 2D back propagation of different type. You are also doing Relu, you are doing matrix multiplication, you are doing max pooling. And it tells you for each and every operation how much time was actually spent in the CPU time and how much time was spent on the GPU. So, you can give that view also, you can go via there are different different things here right.

(Refer Slide Time: 22:32)



(Refer Slide Time: 22:33)

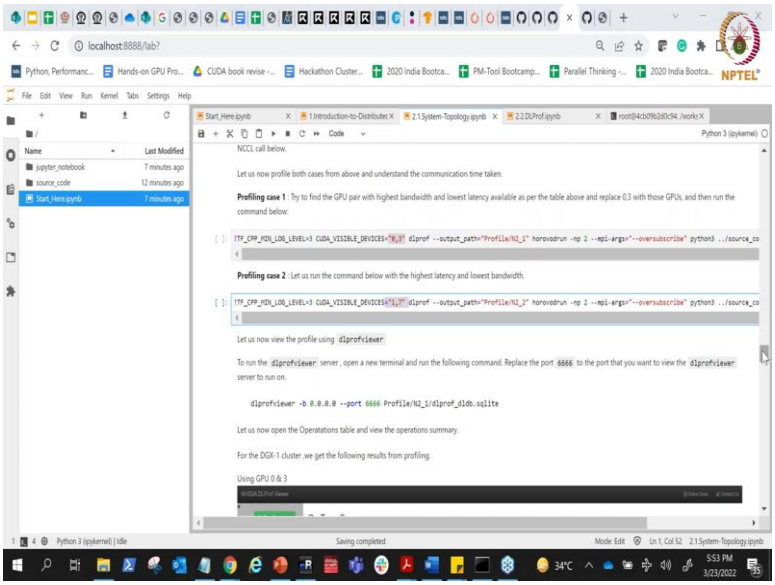


You can go to the GPU view and there you would be able to see, how much utilization is there for every GPU. As I told you that we are utilizing only one GPU, while rest all GPUs are not even utilized. In fact, the GPU 0 itself utilized only for 30 percent right and that is what can be seen via this DLProfiler which can give you very very good insights into your existing utilization of the GPU itself right.

So, we can get more and more details about each and every operation, each and every GPU and it will give you all those statistics. So, that is what is dl prof all about. So, you

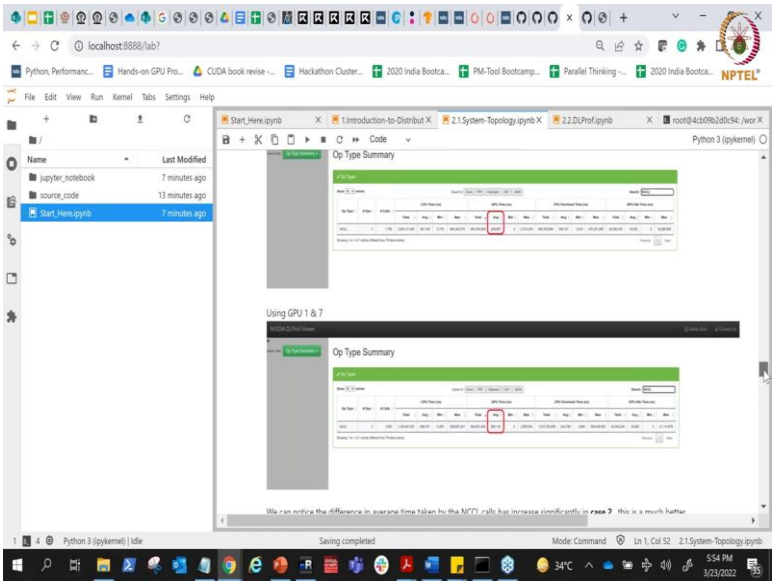
can utilize the dlprofiling tools to make sure that you are getting really good, really good insights into whether you are utilizing your overall system also efficiently or not.

(Refer Slide Time: 23:24)



So, in this particular case, what we were trying to do was to do the same thing that we can run the same using say 0 and 3 GPU using Horovod and 1 and 7 and run the same task which we were running sometime back.

(Refer Slide Time: 23:43)



And you would observe that when you do that and when you run via dlprof you would see that when I use GPU 0 and 3, the average time which is spent on the GPU is 256

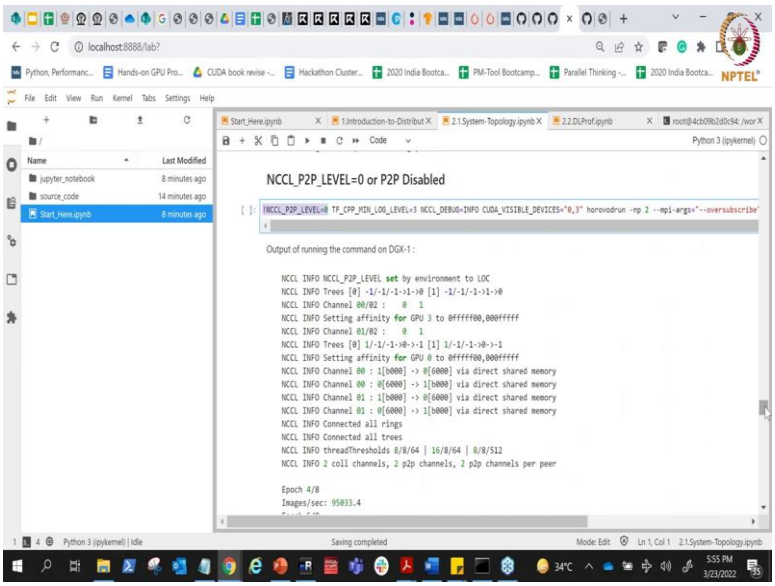
nanoseconds. So, it is showing if it is not visible, I will let me just increase the resolution and you can see here it says 256 nanoseconds.

While if I use 1 and 7, it shows that the time spent in the GPU is actually higher, which is 500 nanoseconds which you might feel I am spending more time in GPU is good actually it is not, because the NCCL communication times are actually increasing and hence you are spending more time.

So, ideally the time should be reducing and not increasing and that is what can be utilized and you can see here I am specifically selecting that I want to see only NCCL time. So, you can see here that the NCCL time which is the communication time on the GPU got doubled when I ran from 0 to 3 to 1 to 7, because 0 to 3 NVLink is there 1 to 7 NVLink is not there.

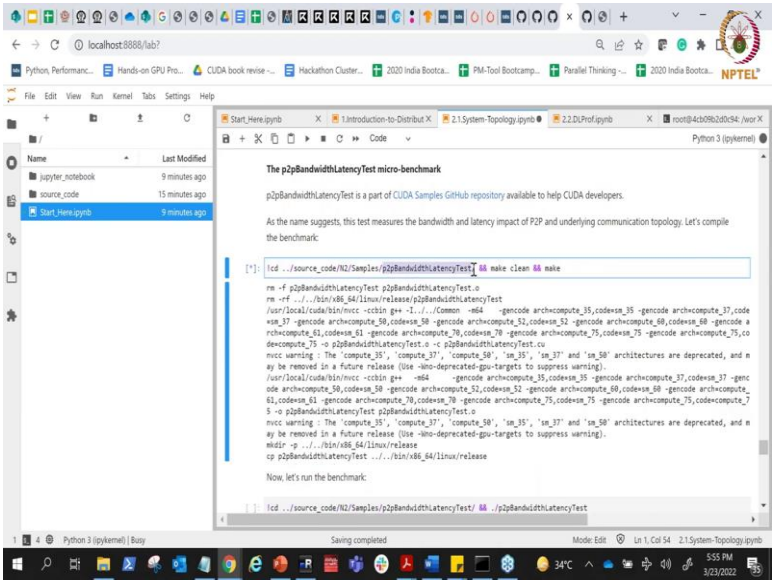
So, NCCL actually spends ends of spending more time on the GPU reducing your overall scaling.

(Refer Slide Time: 25:03)



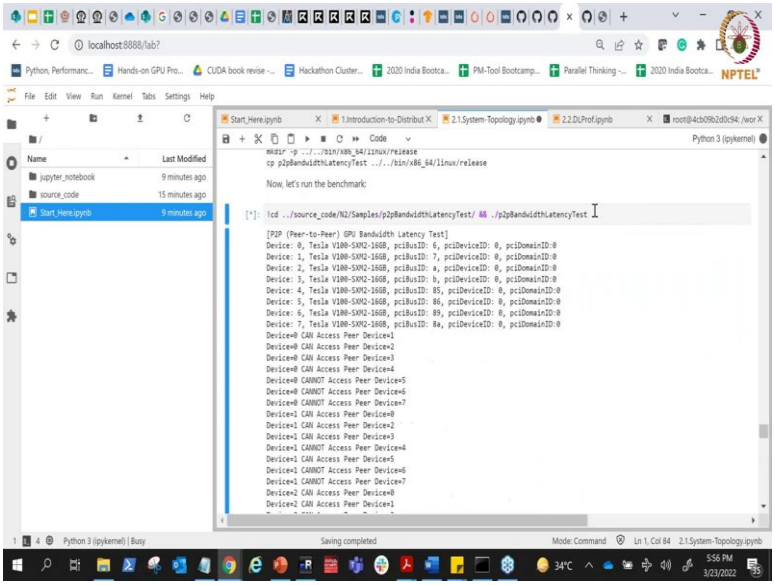
And you can basically do the same experiment. Now, here in this particular case what we are doing is we are intentionally removing peer to peer transfer and doing the same thing and you will end up seeing the same statistic, by you can enable and disable peer to peer on your own right. So, we can see different stats and you will be able to get the same stuff. Another very quick thing I wanted to show you with respect to your system.

(Refer Slide Time: 25:40)



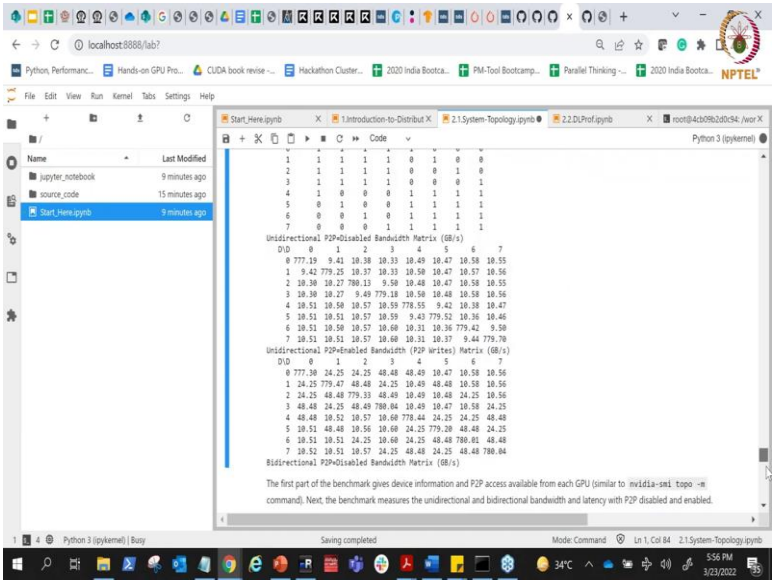
Now, if you have a system and you do not want to get bottleneck by running these kinds of tests, there are open source benchmarks which are available for you to do testings. Like here in this particular case I am using a benchmark which is called as peer to peer bandwidth latency test.

(Refer Slide Time: 25:57)



So, I am combining that benchmark right now and I am going to run this test and I am going to explain it to you what this test can actually do. The test first of all will show you whether peer to peer is can be enabled or not enabled.

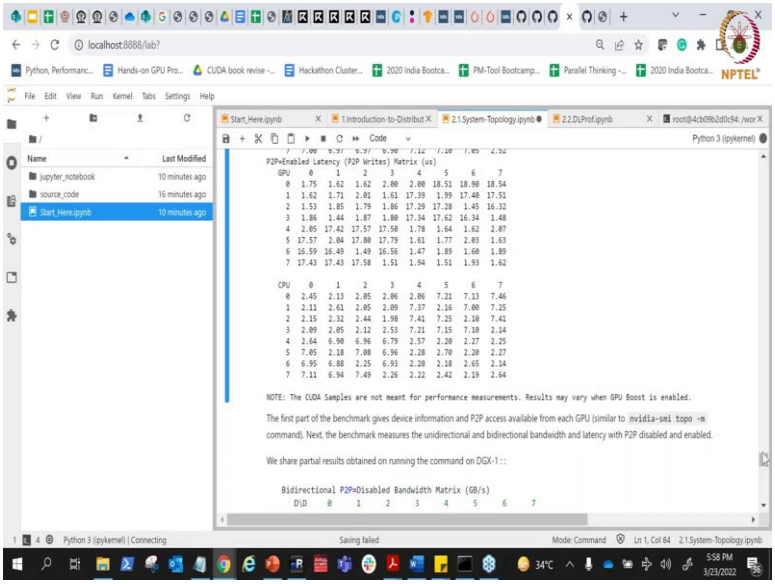
(Refer Slide Time: 26:08)



But more importantly, it is going to show you certain matrix. Here you can see its telling when I disable peer to peer how much bandwidth I can achieve across different GPUs. So, you can see here when I do a bandwidth test between GPU 0 and GPU 1, the bandwidth I am getting is only 9.41 in case I disable peer to peer.

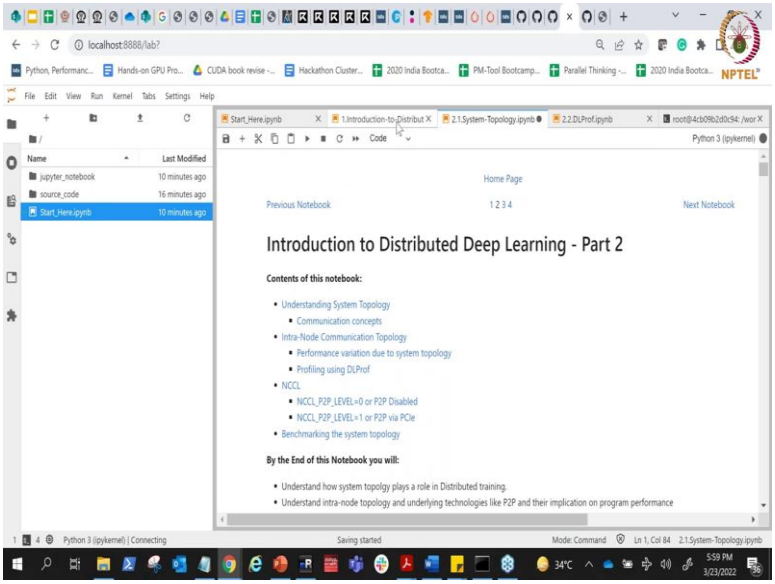
But if I enable peer to peer, in that case my bandwidth increases from 9.41 Gbps to 24 Gbps. Same thing between GPU 0 and GPU 3, if I enable peer to peer it increases because I have double NVLink it increases to 48 Gbps. So, you can run some of this test and you can find out those thing.

(Refer Slide Time: 26:58)



And we are almost at the end where we are talking about why are running peer to peer. You can see here you can also observe that the latency if you disable the peer to peer versus if you enable the peer to peer. If you disable if you do not have peer to peer it is taking much more time like 17 user seconds to reach between GPU 0 to GPU 1. But if the peer to peer is enabled it only takes 1.62 user seconds which is quite good.

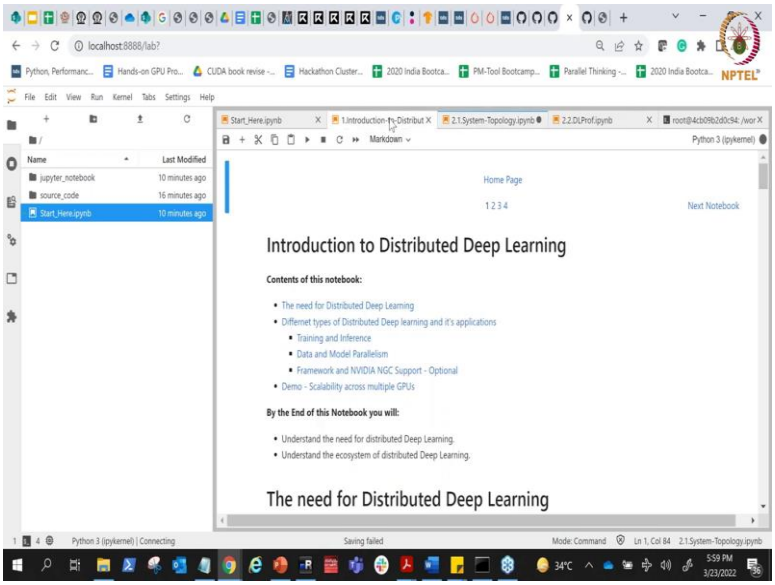
(Refer Slide Time: 27:30)



So, in short what I showed you is couple of things today, what we did was we actually used Horovod and TensorFlow to run it across multiple GPUs. We also showed you the

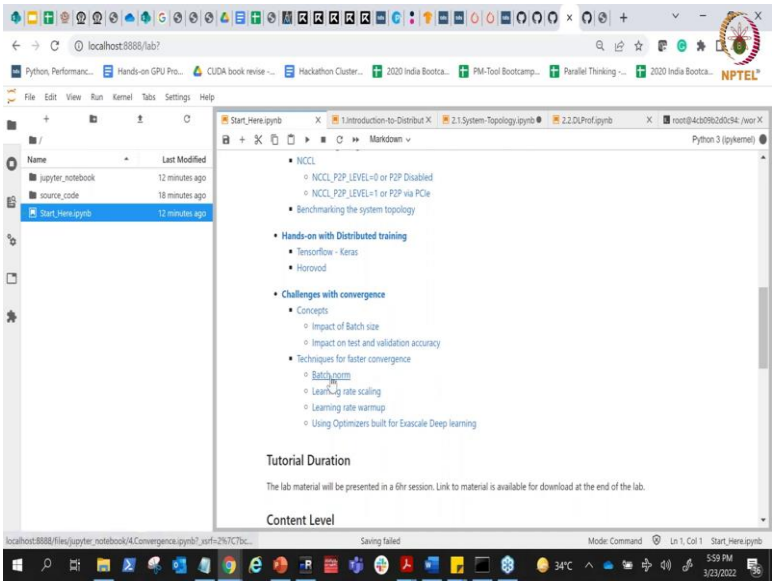
impact of network topology, how you can find those network topology and also you can basically find out in more details by using certain of this profiling tools.

(Refer Slide Time: 27:59)



Tomorrow we are going to go into more details of how to utilize or how to use different frameworks and get into more details of running it on TensorFlow as well as on Horovod.

(Refer Slide Time: 28:05)



And also we are going to cover some of the parts of what kind of a challenges can come our way or with respect to conversions, when you have different size of batch or when

you run it across different say number of GPUs. And you might have to change your convergence strategy or your adaptive strategy in terms of say what kind of a method you use for back propagation. And we are going to cover that into more details in the tomorrow's session.