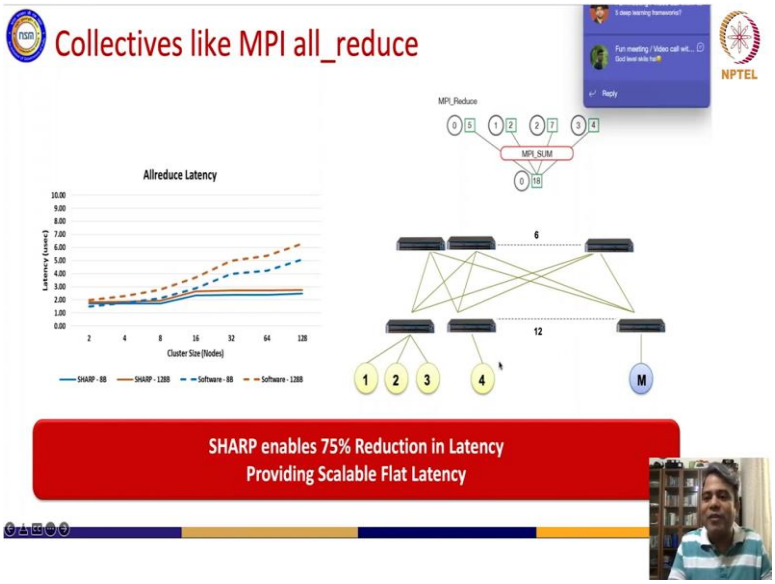
Applied Accelerated Artificial Intelligence
Prof. Ashrut Ambastha
School of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture - 16
Design Principles for Building High Performance Clusters
Networking Fundamentals Part - 3

(Refer Slide Time: 00:15)

So, let us look at some advanced technologies and let us try to put intelligence into the networks. Right now, all the network did was work as a conduit a very high speed and low latency conduit between compute elements. But we want to look at it now from an element which can do some more intelligence ok; so, what do I mean by intelligent networks?

(Refer Slide Time: 00:38)



Let us look at a simple typical high performance computing operation ok or an AI operation. Why are we making this cluster, why are we connecting all these computes together; so, that we can run a very large low large single job by dividing it into multiple compute elements. When you do that let us say you are doing a simple matrix vector multiplication right.

Any application which is trying to solve a partial differential equation will break down into a matrix vector multiplication or a matrix sparse vector multiplication. If it is a very large problem, it will be a very large metric which will not fit into a single system.

And; obviously, you need to break it down and send it across to multiple systems; when you have sent it to multiple systems, they will all compute their portion ok. After they have computed their portion, they will try to exchange the data to each other or they will try to send their results to somewhere right.

So, there are I would say operations in which you would distribute data and then you would collect data, one to many and many to one. These kind of operations in networking domain are called collective operations. And it is required for every I mean even in iterative solvers right, you do one iteration and then you will calculate a new weight and you will do another iteration and then again calculate a new weight and so on.

Let us say everybody is you know one big job was distributed amongst 100 servers and everybody got their job. And now they will do their job and they will send out their results to a master processing which will collate the result, get a new weight or get a new answer and then send across more data to the other processing element.

And this entire cycle will keep on repeating till your iterative process converges into a particular solution or till the residual error is within the acceptable limits. So, every scientific application and every AI and machine learning application also does this kind of communication. Now, you can see ok, one question was what is a factory topology? A factory topology is nothing but; so, as I said if you wanted to connect 100 computers and I had a switch which had only 400 ports.

And let us say I wanted to connect I had a switch of 40 ports and I wanted to connect 400 servers. How would I do that? I will take lots of switches, I will do every switch, I will connect maybe 20 servers and I will uplink the switch to another layer of switches. So, that now I form a two layer topology where I have 20 ports connecting to servers and 20 ports up linking to 20 different switches. Like that I have a second switch as well which has got 20 ports down linking to servers, 20 ports connecting to again different switches.

This whole thing forms a factory network, the way I have connected over here in a two level distributed architecture it is a factory network ok. If I have to cover topologies it is a full you know session on its own; so, let us not go there. But I just showed a simple factory topology to make a 400-node cluster.

Now, the problem is when I am running a job where I have a master process which needs to collect after every maybe 110 microseconds or 100 microseconds it needs to collect data from everybody else. This data collection is the master process is running on one particular server connected to one switch ok.

Because master as the name implies can only be one and every all the safe slave processes needs to send data back to the master. So, master will collect data from slave number 1, it will connect from slave number 2, from 3 and so on and so forth. It can collect only one by one, why? Because master has a single link to the switch. You cannot say that hey why do not we have a master with multiple links to all the switches well we cannot.

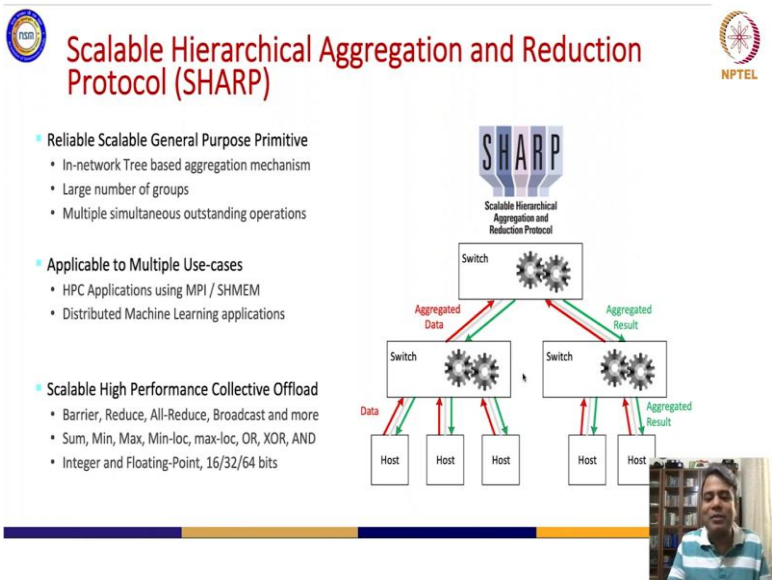
Because, then we will end up taking up the ports of all the switches and you cannot make a big enough network and you cannot always assign only one master, master can be anybody right depending on what is being solved here. So, the topology whatever way it is if there is a master processing it will always result in getting data one by one from everybody else ok.

When it does that it is a in cast scenario, it will take the data it will do some operation like you know MPIs, it will do some average or whatever it will do some mathematical operation and then share its data back. This is what is called a reduction operation MPI all reduce, people familiar with MPI programming would know that ok.

When master processing because it has to do this particular thing, because it has to take data one by one ok it forms kind of a bottleneck. Because, if it is if a cluster now imagine a cluster which is 1000 nodes running a single job. The master needs to take data 999 times like it will take data from 1, it will take data from 2 and so on. So, it will take 999 individual send receive steps ok, before it has collected data from everybody and then it will be able to do the particular operation.

And once the operation is done, it can maybe share the data in a single step, because there is something called multi casting or broadcasting right. The data is a single number the result is a single number which can be sent as a broadcast packet to everyone. But collecting the data is an end step process, if you have a n node cluster, what todays networks start doing is we are so we have made it intelligent.

(Refer Slide Time: 06:55)



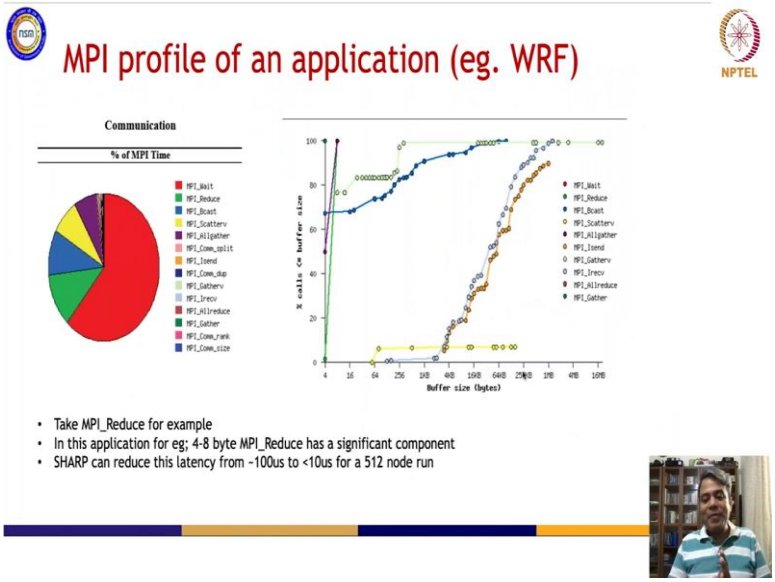
So, that we actually use the networking elements to also take part into computation. Computation which are not standard mathematical computation, but computation which are a network-oriented computation. Like when you are trying to run that reduction operation, the switches or network switches themselves can actually collect data in a single step from all the connected slaves and do a reduction. Reduction can be as simple as doing a sum or a minima or a local minimum maxima or doing an XOR, OR, AND operation depending on whatever application demands ok.

So, now that you look at it, every switch in the network can take data in a single step, because all the servers which are connected to it are connected in a parallel way manner right. So, in a single step it will be able to receive data from any you know 20 servers or n number of servers. Where now, n is equal to the number of servers connected to that switch every switch will do it in a single step, it will do the operation and then send the result to the next level of switch ok

So, now you see we reduced a n step operation to a two-step operation, two step for network size and a topology which is two levels of factory ok. This is one of the advancements in networking where we start integrating compute elements. So, that high performance codes can be accelerated by offloading certain things into the network itself ok. And when because you are able to do it in two steps the top level switch is able to

now share the aggregated result in a single go, but all the lower level switches can collect the data in less number of steps.

(Refer Slide Time: 08:57)



And therefore, and by the way if you look at practical application, I am giving you a simple profile of a weather application called WRF ok. WRF is a weather application which is used for weather prediction and forecasting; it takes huge amounts of input data, it distributes it to multiple servers all the servers, now they need to talk to each other. And these are the various calls that they are doing, they are you know MPI wait refers to the servers waiting to for everybody to finish.

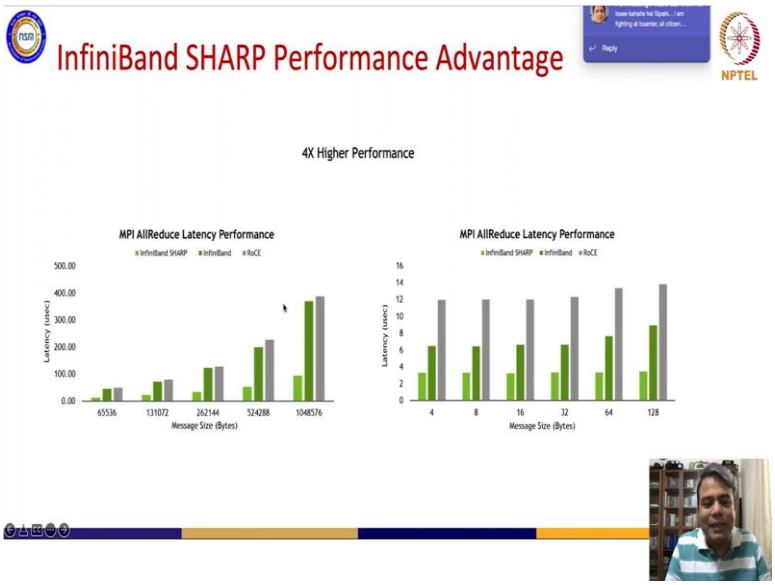
Like, they have to synchronize right, when everybody is doing their job one rogue person cannot keep doing whatever it wants. There has to be some synchronization step where everybody will come to their state take the new data and then start the next iteration. Then when everybody is completed the next iteration, it will again wait and then start the next one.

So, there are you know calls made in the system called MPI wait; you need to do reduction which I talked about, you need to do some broadcast, you need to do some point to point, you need to do all reduce. So, you need to do lots of you know communication operation. I just want you to show want to show you one particular operation which is there in this dark green and which is there in this brown ok, it is all reduce ok.

You see that in a particular runtime almost 20 percent of the time was spent in doing a reduction and also for wait. Wait is one of the main thing you know, because most of the jobs are wanting to be run compute and communication has to synchronize, but you see there are collectives as well. So, there was so much of broadcast the blue part, there was so much of all reduced reduction and the reduction was only 4 byte and 8 byte if you see this green line over here and ok.

So, if we are able to do this particular portion inside the network, we will actually reduce this 20 percentage to less than 2 percent. And therefore, network has actually helped in accelerating the compute. So, it is an intelligent network ok, it is not doing only packet passing this is one of the thing.

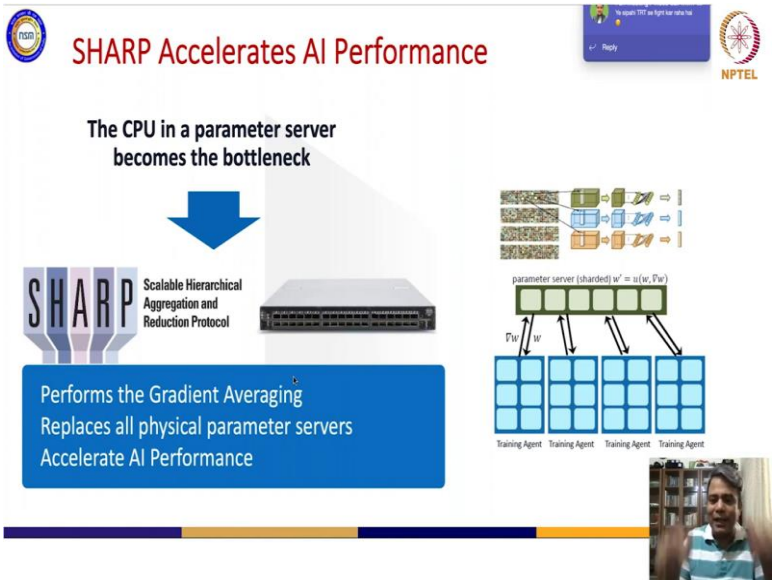
(Refer Slide Time: 11:18)



So, just to give you some examples MPI all reduce performance for a particular message size. If you look at small message size you look at it with look at it as software versus you know hardware when I say the dark green line which is standard InfiniBand versus the light green which is InfiniBand with sharp. If for a certain message size, I took 400 microsecond to complete, a operation with in network computing enabled or with intelligent networks I am able to do it in around 60 microseconds.

Generally, we are able to reduce 10 times the time spent on doing these kind of communication with intelligent networks.

(Refer Slide Time: 12:02)

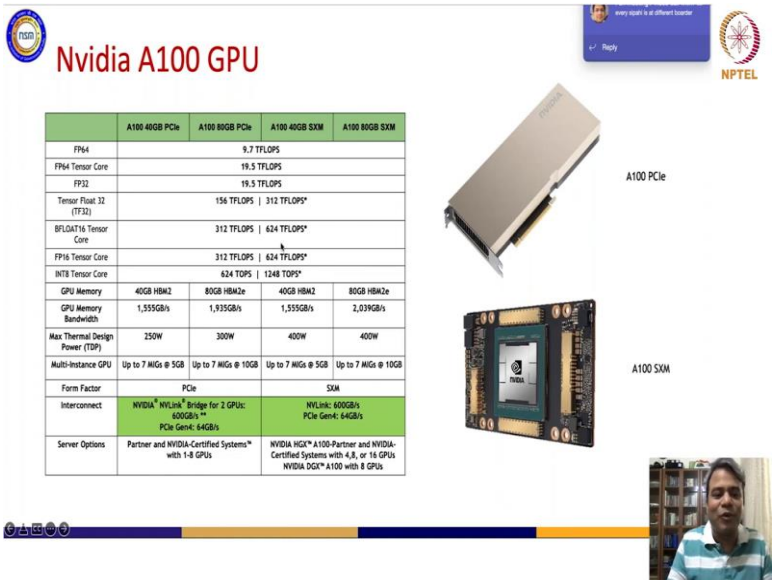


And this whole thing also is required for AI performance, why? Because, think about a simple image recognition right. Whenever you do any training forget image recognition, let us talk about simple training applications right. Whenever you do any training application what does it do I mean, what does the neural network do it.

Actually, takes in data there is a parameter server which actually looks at in every iterative step it looks at all the data input data and tries to train itself by getting a new weight ok tries to assign a weight. And this weight the new weight is a calculation done by collecting data from everybody the delta from everybody and then calculating a new weight factor and then distributing the new weight factor.

Then again, the next training cycle starts where every training agent will again give its own delta weight to a parameter server and then the parameter server will do some operation and again you know try to fine tune. So, this is again the same kind of operation that I talked about earlier right. And therefore, it can also be mapped into network acceleration which is what is done by the current intelligent networks today as well.

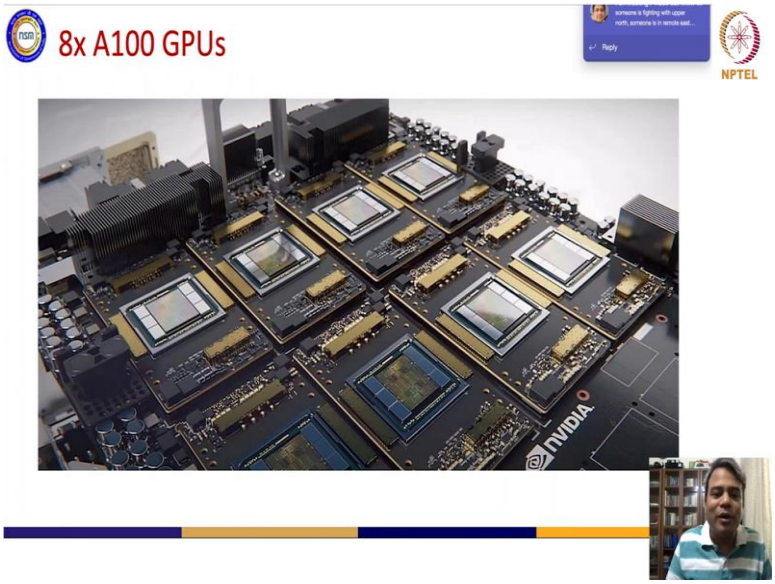
(Refer Slide Time: 13:24)



	A100 40GB PCIe	A100 80GB PCIe	A100 40GB SXM	A100 80GB SXM
FP64	9.7 TFLOPS			
FP64 Tensor Core	19.5 TFLOPS			
FP32	19.5 TFLOPS			
Tensor Float 32 (TF32)	156 TFLOPS | 312 TFLOPS*			
BFLOAT16 Tensor Core	312 TFLOPS | 624 TFLOPS*			
FP16 Tensor Core	312 TFLOPS | 624 TFLOPS*			
INT8 Tensor Core	624 TOPS | 1248 TOPS*			
GPU Memory	40GB HBM2	80GB HBM2e	40GB HBM2	80GB HBM2e
GPU Memory Bandwidth	1,555GB/s	1,935GB/s	1,555GB/s	2,039GB/s
Max Thermal Design Power (TDP)	250W	300W	400W	400W
Multi-Instance GPU	Up to 7 MIGs @ 5GB	Up to 7 MIGs @ 10GB	Up to 7 MIGs @ 5GB	Up to 7 MIGs @ 10GB
Form Factor	PCIe		SXM	
Interconnect	NVIDIA® NVLink® Bridge for 2 GPUs: 600GB/s ** PCIe Gen4: 64GB/s		NVLink: 600GB/s PCIe Gen4: 64GB/s	
Server Options	Partner and NVIDIA-Certified Systems™ with 1-8 GPUs		NVIDIA HGX™ A100-Partner and NVIDIA-Certified Systems with 4,8, or 16 GPUs NVIDIA DGX™ A100 with 8 GPUs	

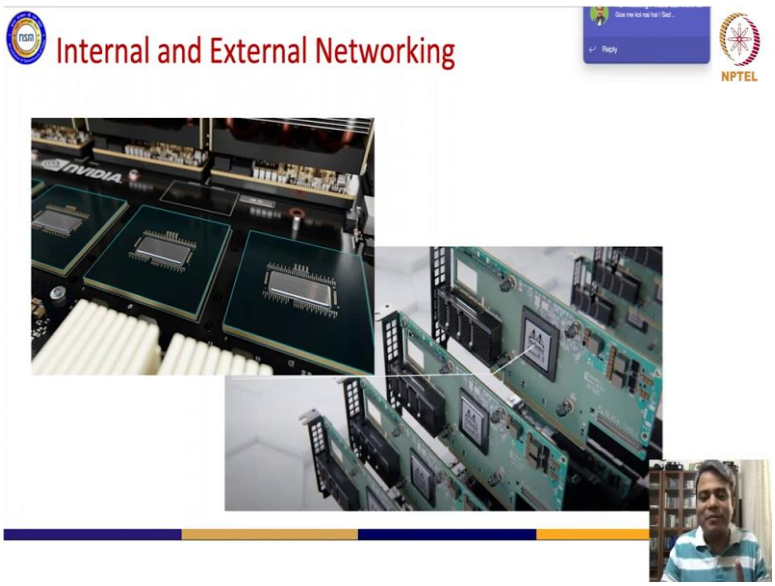
So, one thing is you know. So, that is one technology let us say ok, how do we apply it. So, to apply it let me go into a little bit of details of a computer architecture, a current GPU accelerated computer architecture. You all are aware of standard GPUs, you must have seen you know I am showing a current generation todays Nvidia A100 GPU. It has certain amount of teraflops or floating point 64 performance, it has got certain amount of connectivity you know you can see here different interconnects. So, that multi GPU can communicate to each other and solve a job.

(Refer Slide Time: 14:07)



And you and therefore, and this GPU is then put into a motherboard multiple of them are put into a motherboard to form a GPU accelerated system. All these GPUs are connected to each other using very high speed interconnect and they exchange data between each other ok.

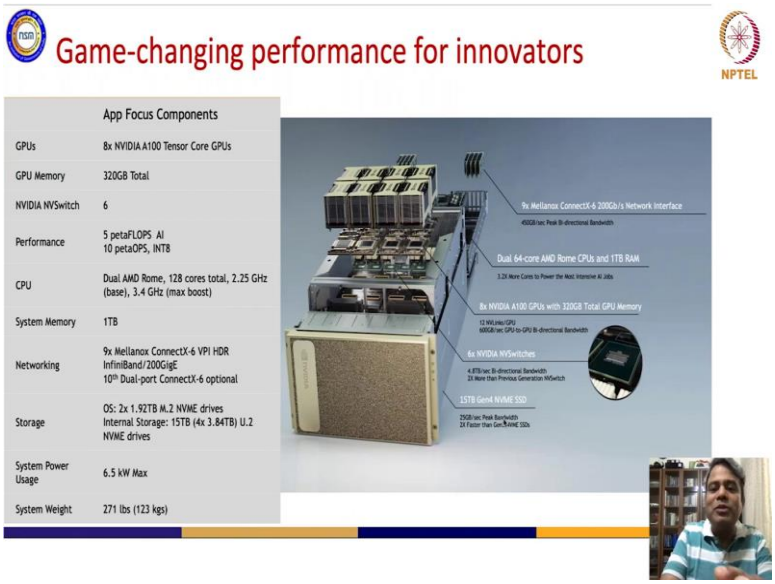
(Refer Slide Time: 14:24)



When you want to go out of a server what if you need more than 8 GPUs for your training workload. There are clusters right now if you look at top 500 which have got 1000 of GPUs doing a single job ok; so, you need to go out of that 8 GPU. So, here we have connected those 8 GPU in a single motherboard, but if I want to go out then I need to have those network cards this is what I show in this picture.

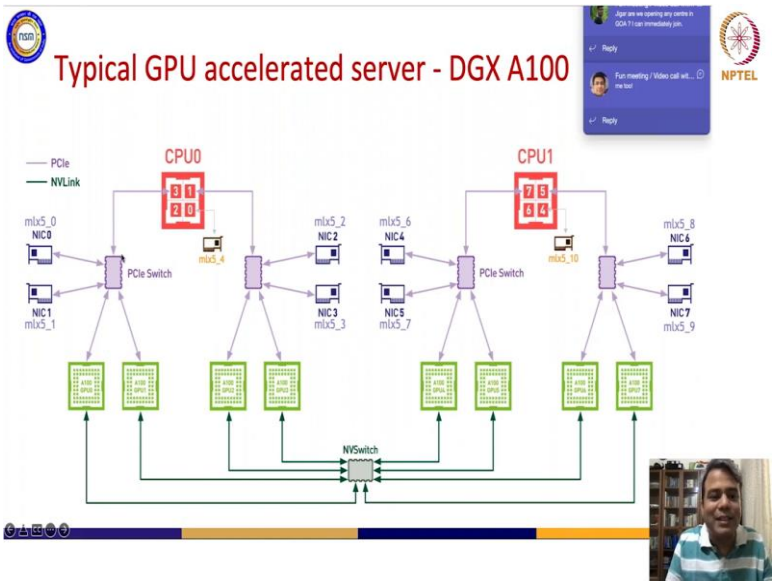
There are multiple network cards over here right, they also connect on to the motherboard and it forms a full GPU accelerated system.

(Refer Slide Time: 14:51)



This is one particular building block of a particular supercomputer right, so many GPUs on it, so many network cards on it, but and this is how it looks like.

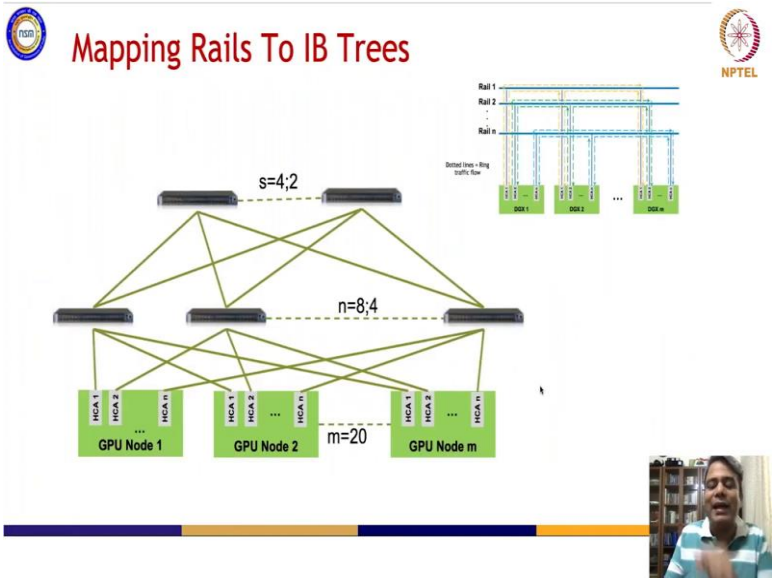
(Refer Slide Time: 15:02)



If you look at the logical connection of various elements, it contains the processor, it contains some PCI express switches. You would have seen any standard PC there is a PCI card right, how do peripherals connect into a server there are PCI cards. PCI cards are put into slots which have their electrical signals coming from a PCI switch ok. But,

what I want to create a cluster which is bigger than this ok, then I will connect multiple of these systems onto a single cluster.

(Refer Slide Time: 15:37)



And again, I give you an example of that standard factory topology ok, what I do here is as you can see; then I should be able to show my pointer as well yes. So, what I do here is, I take a GPU node with multiple GPUs inside which has got multiple network cards to exchange data outside and I connect it to multiple switches ok. And all these switches are in turn connected to another layer of switches; so, that every node can talk to every other node without any blocking ok.

When I have connected all these nodes in this particular manner, I form communication rings between the various GPUs inside the servers. When I form these communication rings, I am able to do those reduction operations that I talked about within the domain of these rings. So, my switches can actually contribute to doing calculations of an AI workload as well ok and that is the reason. And the reason, I have connected my servers in this particular manner is to make use of the technology that I talked about earlier.

So, when you look at; so, the whole point of this talk was design principles for high performance compute right high performance clusters. This is one of the design principles, you do not when you create a large supercomputing cluster or a large AI or a machine learning cluster. You do not randomly go and connect anything anywhere, why?

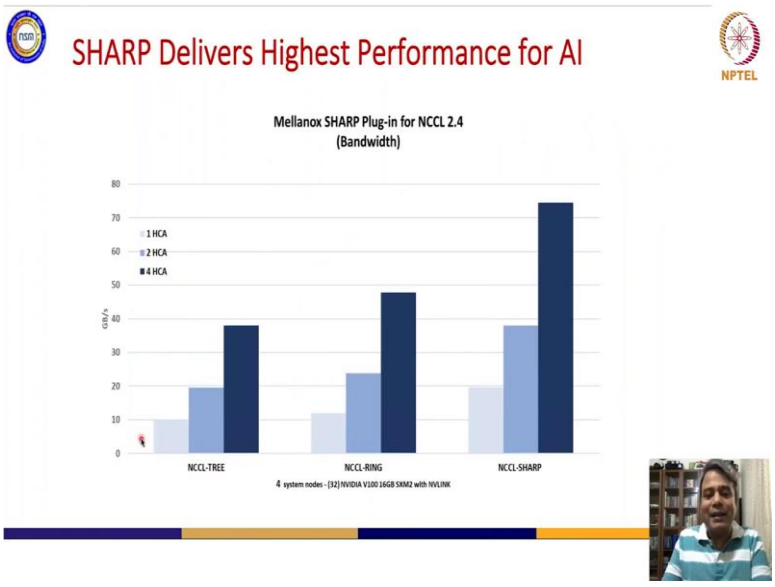
Because you need to exploit all the benefits of the technologies that are present in high performance networks.

And for doing that you need to look at the internal architecture and connect it in a certain way and when I say you need to do it I am not saying that every user needs to look into it, but every user should be aware of it. Finally, when it comes and gets deployed from a particular company or a particular institute, a most of these things are taken care off.

But when the user understands in depth of why it is connected like this, what maps to what then, when you do your parallel programming, you will try to pin things properly. You will try to pin the buffer of GPU 1 to the adapter 1 or the data transfer that has to happen from the buffer of GPU 1 should go from the adapter number 1 ok.

Again, it is all taken care of by the internal libraries that are built in, but those libraries are configurable right there can be systems which is like what I showed you right now, but there can be systems which might have a different kind of architecture. So, to be able to tune the libraries ok; so, that the distribution of data and so that the distribution of buffers is done in such a manner that it can actually make use of the entire system with all the advanced technologies is the holy grail over here right.

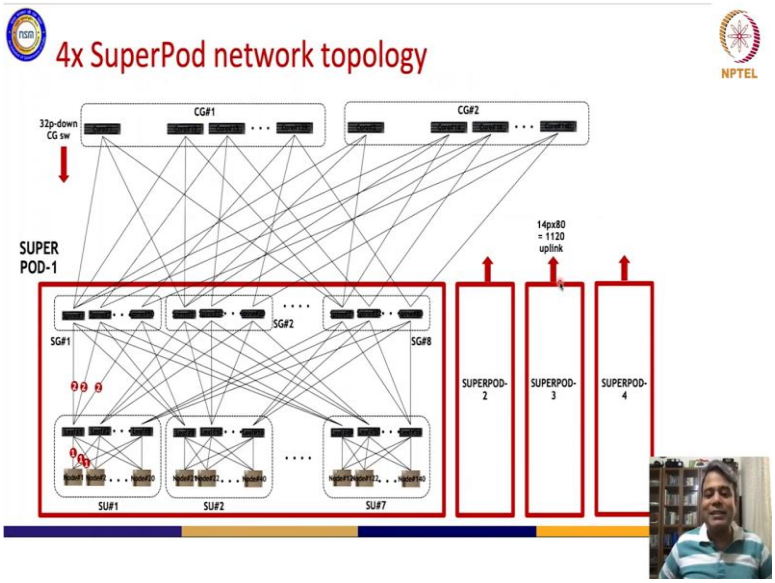
(Refer Slide Time: 18:56)



So, this is one thing, again what all does it give you. So, I am just showing you again the reduction bandwidth if you were using nickel library. Let us say you are doing CUDA

you have collective communication library from Nvidia, it is nickel. If nickel was running as software, you would get a certain amount of bandwidth. As soon as you were able to do offload of many of the processes onto the switching element you would almost double the bandwidth ok.

(Refer Slide Time: 19:34)



So, and that is the reason ok this what I am showing is this is actually the topology of one of the largest clusters that Nvidia has or one of the top. And it is also the topology of many of the largest supercomputing clusters in the top 500 list. It is connected in this manner what we talked about; you know you have got these compute nodes at the bottom which do a crisscross connection into a group of leaf switches which do a crisscross connection into a group of spine.

The number of crisscross connections versus the group number of switches in a group is all defined by what would be the most optimal way of offloading the collective operation. And then because it is such a large cluster you need a third level of switching as well. Again, this is an example of a factory topology, a three tier or a three-level factory topology. So, the reason these clusters are made like this is because of all the things that we talked about till now.