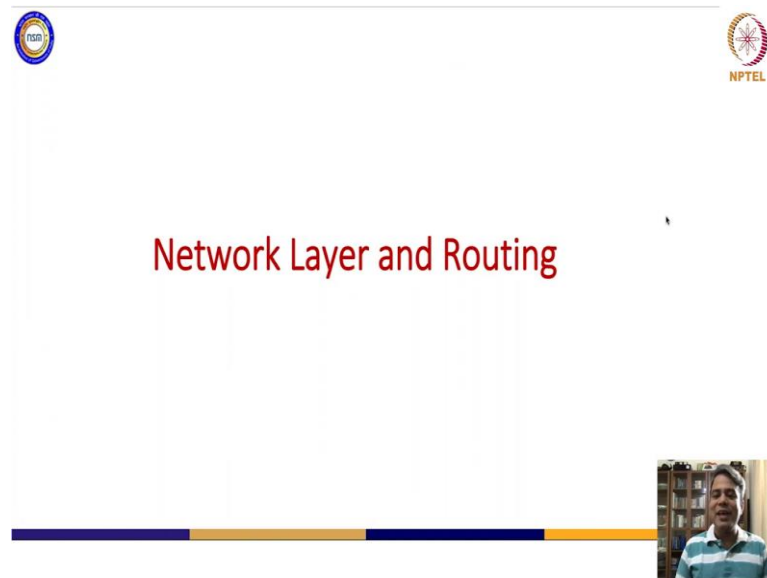


Applied Accelerated Artificial Intelligence
Prof. Ashrut Ambastha
School of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture - 15
Design Principles for Building High Performance Clusters
Networking Fundamentals Part - 2

(Refer Slide Time: 00:15)



Let us say we have created good enough physical media and we are able to send data from one point to the other, but when you are trying to create a cluster it is not point to point as we as I mentioned previously right. So, you need to do you need to connect it to some element. So, on top of the physical layer now comes the network layer and routing ok.

(Refer Slide Time: 00:41)

HDR InfiniBand Switch

QM8700, 1U Series

40 QSFP ports (PAM4, 50Gb/s per lane)
40 HDR (200Gb/s)
80 HDR100 (100Gb/s)

130nsec latency
390M messages per second (64Byte)
16Tb/s aggregated bandwidth
SHARPV2 low latency data reduction and streaming aggregation
22" depth
6 fans (5+1), hot swappable
2 power supplies (1+1), hot swappable

<https://www.nvidia.com/en-in/networking/infiniband/qm8700/>

So, before we get into the network layer we need to look at a switching element I am giving an example of an InfiniBand switch because I come from Nvidia we use InfiniBand we also use many other kind of interconnects like NVLink you know industry uses many different kind of interconnects as well, but this is one of the most popular interconnect for HPC systems.

InfiniBand constitutes around 99 percentage of top ranking systems right. What is this switch? Well it is actually a silicon device which has multiple ports as you can see on the front panel, on these ports you connect the physical media that I talked about you connect a copper cable or an optical transceiver ok.

The transceiver or the copper cable is receiving data from one server, the data comes in at a very high rate and we are looking at this one is let us say 200 gigabits per second, the data comes in and now the silicon needs to decide what to do with the data. So; obviously, it contains some sort of header; you know header contains the source destination the source address and the destination address.

So, the silicon inside this switch is required to look the destination and then it is required to check from which port it should send that data out, so that it will reach it its destination for doing all these checks it needs to have some sort of a table inside ok. So, that and the table needs to know to reach a certain endpoint what port I should use, this is where the concept of switching and routing comes into picture.

And the time taken for this silicon or the time taken for this ASIC inside the switch to take a packet in decide, where to send it out and to send it out is the latency between its two ports. And this is again one of the most important parameters for high performance network, when we talk about latencies in today's terms it is of the order of 130 nanoseconds for this particular switch.

That means 1 data byte coming in or a packet coming in from port number 1 after 130 nanoseconds will appear outside a destination port ok so, but to do that first we need to create a network, right. So, how do we what do we do?

(Refer Slide Time: 03:16)

The slide is titled "Designing Small HPC/AI Clusters" and features the NPTEL logo in the top right corner. On the left, there is a screenshot of a Mellanox website article titled "Designing an HPC Cluster with Mellanox InfiniBand Solutions". The main part of the slide contains three network diagrams:

- Diagram 1 (Left):** Shows two server racks, each labeled "24-ports", connected to each other by a horizontal line labeled "12x links".
- Diagram 2 (Top Right):** Shows a central switch labeled "18-ports" at the top. It is connected to two server racks, each labeled "18-ports", below it. Each connection is labeled "9x links".
- Diagram 3 (Bottom Right):** Shows a 2x2 grid of four server racks, each labeled "18-ports". The racks are interconnected horizontally and vertically, with each link labeled "9x links". A red prohibition sign (a circle with a diagonal slash) is overlaid on the central intersection of the grid, indicating that this configuration is not recommended or is problematic.

In the bottom right corner of the slide, there is a small video inset showing a man speaking.

So, if you need to create a cluster ok, small cluster you will take multiple servers and you will let us say connect it all connect it all to that particular switch, the switch as you saw on the previous diagram contained around 40 ports.

So, data it can actually connect 40 different compute elements together and it can form a single cluster, when you want to make larger cluster you need to how do you do it? You know either you can increase the size of the switch or you can try to connect multiple switches together to create a particular topology.

Now; obviously, the limitation of what size a switch you can make you know you would see there are large switches as well. If people are from the Ethernet world you will see

there are switches which contain 1000 ports or 500 ports or 40 ports or 100 ports even in InfiniBand there are switches which contain 1600 ports.

But this is not a single switch it is actually a big device which contains multiple ASICs inside and they are connected to each other in a certain way. A single ASIC there is limit to how big an ASIC you can make one of the simplest limit is the die size you know you take a large wafer and to get good amount of yield you will have certain size of die on it, a certain size of die will be able to service only certain amount of ports.

Because finally, you need to take electrical signals out of that die how many ports you can take out of that die would depend on how many pins or ball grid arrays you will have from that silicon. It is a very simple way of saying that ok I cannot take a 2 inch by 2 inch die because then you know my yield is not very high.

So, we always you know industry always tries to do the best price performance. And based on that the largest radix or I would say the largest port count processor network processor that you see today is of the order of maybe 40 ports or 128 ports ranging between that, right.

And obviously, you know that there are clusters which contain thousands of compute elements. So, if a single chip has got only 40 ports how will you make a thousand of compute elements. So, how will you connect thousand of thousands of compute elements? You start making these topologies ok. Now, topologies in a high performance computing cluster cannot be random, why?

So, the whole World Wide Web the whole internet is a particular topology right. Every endpoint is connected to every endpoint through some link you know the we have the way I am presenting over here right now my computer is connected to my service provider, to my society is you know fiber box from there to my service providers distributed unit from there to a core switch from there to one of the major routers or the gateways in India in Mumbai maybe from where you know there are some intrinsic cables it goes somewhere.

So, you know Ethernet a World Wide Web it is a connection of all these elements and you have a standard protocol running through all this. In an HPC cluster you cannot do just random connections why? Because the aim is different the aim is that all compute

elements it is a democracy all compute elements have same kind of capability they have therefore, they should have same kind of network bandwidth and latencies and delays.

Therefore you need to connect them in a well defined manner and because you and one of the another reason that you need to connect them in a well defined manner is because and will come to that is because you need to get them to talk to each other in a symmetric way right.



(Refer Slide Time: 07:25)

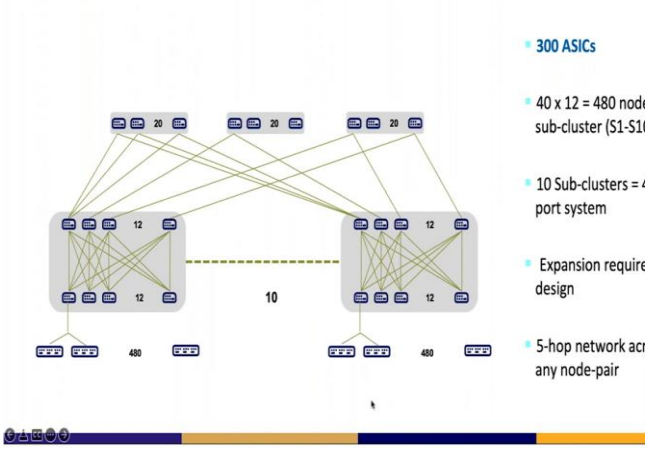
The slide is titled "Network Topologies" and features four diagrams illustrating different network topologies. From left to right, they are: "Fat Tree" (a hierarchical tree structure with a wider base), "Hypercube" (a 3D cube with internal connections), "Torus" (a grid of nodes with connections forming a torus shape), and "Dragonfly" (a circular arrangement of nodes with multiple interconnections). The slide also includes the IITM logo on the top left and the NPTEL logo on the top right. A small video inset of a speaker is visible in the bottom right corner.

And therefore, we create I will not go into the loops let me go forward. So, you create a certain a network topology which is defined by equations therefore, you can have equation defined routing to be done between these elements.

There are various kind of network topologies we look at standard tree topologies, fat trees, hypercubes, toruses, dragonfly what is common between all these and what is the difference between this and a World Wide Web mesh random mesh? The difference is that all of these topologies are very nicely mathematically defined therefore; you can define routing between all the elements which constitute this particular network.

(Refer Slide Time: 08:17)

 **3-level Clos with 1U HDR switches** 



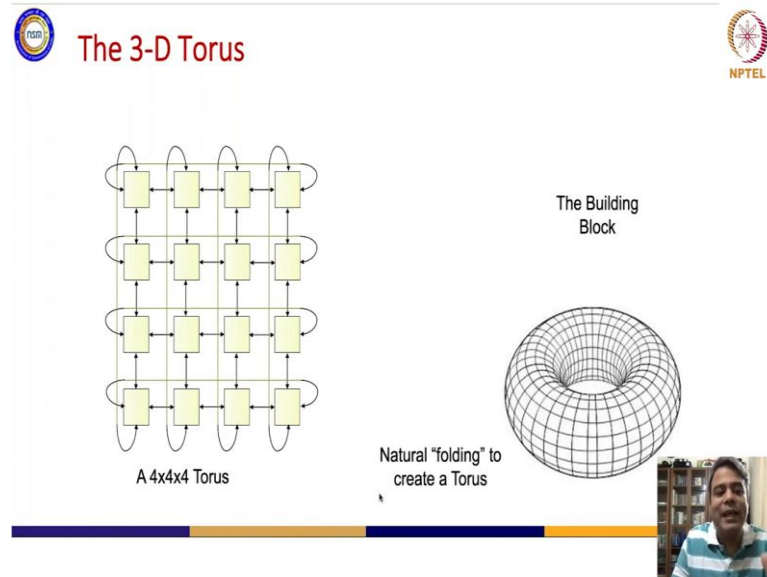
- 300 ASICs
- 40 x 12 = 480 node sub-cluster (S1-S10)
- 10 Sub-clusters = 4800 port system
- Expansion requires re-design
- 5-hop network across any node-pair

And this is where the network layer the link layer and the network layer comes into picture, yeah. These are some examples of topologies you know if you want to make large clusters you will end up putting multiple of these switches, you will connect them to each other in a particular fashion, you will create multiple layers and therefore, you can create very large clusters and talking about a normal 3 level Clos topology.

Because let us say if you connected only 2 you took only 2 switch the maximum compute that you could connect is equal to the number of ports of these two switches minus the number of ports which are required to connect the 2 switches together. So, if it was a 44 port switch maybe you could connect 40 you know 30 servers to switch 1, 30 servers to switch 2 and there were 10 links between the two switches.

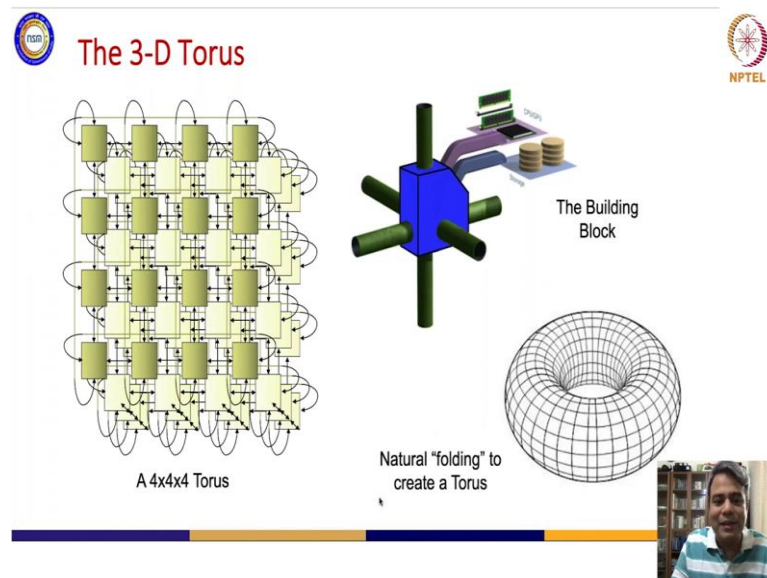
If you want to grow more you could create a triangular or ring network, if you want to grow more you start making these kind of trees, ok.

(Refer Slide Time: 09:10)



Or you start making these kind of toruses, here every box represents a switching element, every arrow represents the link between the switching element ok.

(Refer Slide Time: 09:20)



And then you keep on folding these connections and you create a very nice well defined mathematically explained topology. Therefore, you can actually put this equation into the routing engine of the switching element there which will now know where to send the incoming packet if it has a certain destination address.

High performance network need this deterministic routing and there are now adaptive routing as well which we will cover in the next session next part of this presentation.

(Refer Slide Time: 09:59)

Dragonfy+ {240 ASICs}


- All-connect "Mesh" of "Groups"
- Each Group is full bi-partite
 - 20x 40p 200G switch per group
 - 200x 200G intra-group links per switch
 - 200x 200G inter-group links per switch
- 400 servers per group
- System can accommodate up-to 20 groups (8000 servers)
- Min-hop routing will have 4-hops across any node pair

But you know the bottom line and the takeaways to create a high performance cluster you need the network to be well defined, so that the routing elements are they contain routing tables which are given by an equation, so that they can send out packets without any issues or without any delays ok.


More than that you always want whenever we are creating clusters we always want there should not be any packet drop in Ethernet we are allowed packet drops right. Why? Because well it connects to the entire World Wide Web and you know there can be congestion if you know it is not a 100 percent reliable thing, if there are problems in the middle you can drop packets and then the software can request for retransmission.

In HPC if you have the software trying to request for retransmission the whole process will become so slow that you cannot actually scale your application. So, you need to have a routing which is credits based as in every element knows that it has got a path available to its end point. And therefore, every source can send data to the destination without dropping packets.

(Refer Slide Time: 11:25)



Infiniband Packet Format



- Packets are routable end-to-end fabric unit of transfer
- Link management packets: train and maintain link operation
- Data packets
 - Send
 - Read
 - Write
 - Acks

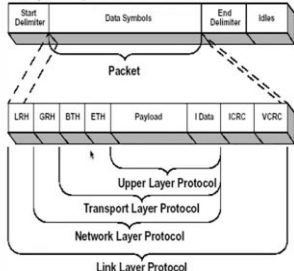



Figure 27 IBA Data Packet Format



If you just now look at now let us go one step back you know. So, let us complete the link layer and the network layer portion. So, as I said we are sending zeros and ones, zeros and ones are bytes which are put into packetized formats from the server.

The packet contains maybe some you know delimiter to basically tell the server that ok. One packet is coming send it out, it will contain a local routing header or global routing header some transport header you know it is a standard OSI 7 layer model, which is actually the basis for most of the network actually I would say all of the network ok.

So, there will be some identifier which says where is my from where I am coming the main payload is this in the center, but there has there are bytes which define hey from where this payload is coming for where this payload is intended to go at least this much is needed right. And then what kind of operation is it. So, all this comes inside the packet format comes into the network layer and the transport layer ok.

(Refer Slide Time: 12:46)

Transport Layer and RDMA

So, we covered till physical layer link layer and networking layer, let us go into the transport side ok. Transport layer is the entity which defines for high performance computing cluster, it defines a so for normal you know normal Ethernet you know there is sockets like there is a sockets interface if you are trying to send data between two servers you send it through sockets ok.

In high performance clusters you do not open sockets, if you open sockets and you put data into that it needs to be processed by the kernel and the driver ok.

(Refer Slide Time: 13:21)

RDMA in SuperComputing Clusters

- Remote Direct Memory Access (RDMA)
- Advance transport protocol (same layer as TCP and UDP)
- Main features
 - Remote memory read/write semantics in addition to send/receive
 - Kernel bypass / direct user space access
 - Full hardware offload for network stack
 - Secure, channel based IO
- Application Advantage
 - Lowest latency
 - Highest bandwidth
 - Lowest CPU consumption
- Verbs: RDMA SW Interface (Equivalent to Sockets)

Server - Initiator

Server - Target

TCP/IP

Server - Initiator

Server - Target

RDMA

So, actually if you look at it on the bottom side on the top side is what happens in a typical TCP/IP protocol. The application which is a routine let us say it is a routine which contains data in a particular buffer and that buffer is part of the; obviously, the RAM ok, the application copies that buffer to the sockets layer that copies the data from its own buffer to the transport and protocol driver inside the adapter in a server, the adapter breaks it down into an Ethernet we call it MTU right.

So, it has to break it down into a particular transfer unit and then it will send it out through the physical adapter through the cable that we talked about or through the interconnect media there can be switches in between and connected to various topologies.

The job of the switch is to look at the source and destination and route it accordingly we are assuming that it has routed it correctly and it has reached the destination that in the destination again there will be multiple copies of that data between various portions in the buffer and the data will reach the target.


This whole cycle of copying data from server to a client or server number 1 to server number 2 is the latency. Let us say I wanted to send only one floating point number a double precision floating point number from server number 1 to server number 2 in today's world we take around a microsecond ok. In TCP/IP based connection we take around 10 microsecond. Why do we take 10 microsecond? Because of these so many buffer copies are involved.

Therefore for high performance network you need to have the transport layer which kind of bypasses all these intermediaries, it bypasses the entire kernel driver and protocol driver and buffer copies and so on and so forth. And which is where an example like InfiniBand which does something called Remote Direct Memory Access RDMA operation where it can actually read data from buffer of one application, which is in the RAM of one application and put it across to the memory or in the RAM of the second application ok


Because it is able to do that the latencies are of the order of some microseconds or actually some 100s of nanoseconds in today's world. If you take 2 servers ok let me come into a real world today's world right now if you take 2 servers GPU enabled or not connected with a wire through an InfiniBand adapter or any you know high performance

network adapter and you try to do a ping pong between the two server by exchanging maybe few bytes ok. You will be able to do it in around 600 to 700 nanoseconds, because we bypass all these points ok.

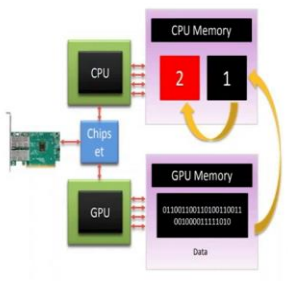
(Refer Slide Time: 16:29)




GPUDirect RDMA



- **Prior to GPUDirect**
 - GPU use driver-allocated pinned memory buffer
 - RDMA use pinned buffers for zero-copy kernel-bypass communication
- Impossible for RDMA drivers to pin memory allocated by GPU
- Two copies
 - GPU copies data from GPU internal memory to GPU driver system pinned memory (1)
 - User space needs to copy data between the GPU driver system pinned memory (1) and RDMA system pinned memory (2)
 - RDMA device sends data to network





Now, comes another important concept of having accelerated computing. We all have been doing these sessions for GPU accelerated computing, I am sure you have gone through sessions where people talked about GPU acceleration, CUDA programming and you know kind of offloading parallel workload onto the GPUs and so on.

When you do that and you know as long as things are confined within a single GPU its fine, but now when you want to create a cluster of GPUs again you need to transfer data from one GPU to the other. So, currently if you look at how systems are made you would have a CPU you just open a you know if you look at a motherboard of a standard server you will see a CPU, you will see a chipset you know some PCI express switch on it on the motherboard, the CPU will have the drams which are the memory it will also contain a GPU ok.

The GPU will have its own GPU memory its own RAM, when you are trying to accelerate a workload onto a GPU you will actually copy the operands from the CPU memory onto the GPU memory. And the GPU will then do some matrix-vector multiplication matrix-matrix multiplication and then after certain iterations if it needs

some newer operands it will write down the result onto the CPU memory get some new operands ok.

All this communication is done from the CPU and the RAM through the chipset to the GPU memory. What if we wanted to send data across two servers containing GPUs? Let us say my matrix was quite big and I want to send from memory of GPU number 1 to memory of GPU number 2 which is a different server. So, now, I need to copy my data because I am able to do DMA operation from only the system RAM, we were in sending data from GPU or memory from system number 1 to memory of system number 2 ok.

When you want to do that you can only address the memory which is in the system which is the RAM, when you want to send data from GPU memory you will need to copy the contents of the GPU memory into the system RAM into the area which is addressable by the GPU.

And then there needs to be a memory copy done between the area which is addressable by the GPU and the RAM area which is addressable by the network card you need to do that copy and then the network card can do a DMA operation which I explained in the previous slide.

(Refer Slide Time: 19:10)

The slide is titled "GPUDirect RDMA Evolution (cont.)" and features the NPTEL logo in the top right corner. It contains a bulleted list on the left and a diagram on the right. The diagram illustrates the flow of data from GPU memory to CPU memory. A CPU is connected to a GPU via a "Chipset". The CPU has "CPU Memory" containing a red box with the number "1". The GPU has "GPU Memory" containing binary data. An orange arrow indicates data being copied from the GPU memory to the CPU memory. Below the diagram is a video player interface with a small video thumbnail of a man speaking.


- **GPUDirect / GPUDirect P2P (Peer-to-Peer)**
 - GPU and RDMA devices share the same pinned memory buffer
- **One copy**
 - GPU copies data from GPU internal memory to system pinned memory (1)
 - RDMA device sends data to network

What we did was rather than doing this memory copy ok we worked with systems where we worked on chipsets and technologies where we said ok let us have a unified memory


in the system RAM which is addressable simultaneously by the GPU as well as the adapter. Obviously, there are lock mechanisms implemented in the control plane. So, that you do not end up corrupting the memory, but once GPU has written its data onto the system RAM the adapter can access the same area in the system RAM and do a DMA operation to the other side.

And therefore, it gives a good amount of acceleration because now you have completely avoided the memory copy within the system RAM; he went one step ahead and said hey if I have to exchange data from one GPU's memory to the other GPU memory.

(Refer Slide Time: 20:00)



GPUDirect RDMA Evolution (cont.)



GPUDirect RDMA

- GPU memory is exposed to RDMA NIC
- Direct data path from GPU to network - Data path is zero copy
- CPU is involved in the control path - WQE preparation, ring doorbell, handles completions for incoming packets to GPU

Zero copy

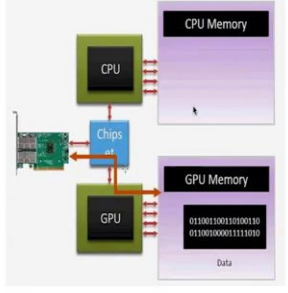
- RDMA device sends data to network from GPU memory
- RDMA device receive data from network to GPU memory


The CPU still synchronizes between GPU tasks and data transfers

```

while(true) {
    gpu_kernel <<<... stream>>>buf;
    cudaStreamSynchronize(stream);
    lib_post_send(buf);
    lib_poll_completion;
}
                    
```

100% CPU Utilization






Why should I even copy it into the CPU memory, why cannot I have my network card directly access to GPU memory and therefore, save up all the cycles of sending data from GPU memory to CPU memory and this was also done.


So, these technologies in the transport layer these are all part of the transport layer ok, these technologies in the transport layer of network are called RDMA, GPUDirect RDMA you know GPU RDMA sync and so on and so forth fancy words. But all they do is in that packet that I showed in the previous slide apart from the source and destination identifier address it contains a transport header, which is nothing but a DMA pointer to the memory location of the destination where the payload needs to be put.

On reception of that packet an intelligent adapter makes sense out of the transport header and puts the data directly into the GPU memory. And therefore, now you can get huge amount of bandwidth gains without using any of the CPU cycles and huge amount of latency gains as well. So, whenever you hear the word GPU RDMA or RDMA this is the technology that people are referring to.

(Refer Slide Time: 21:35)



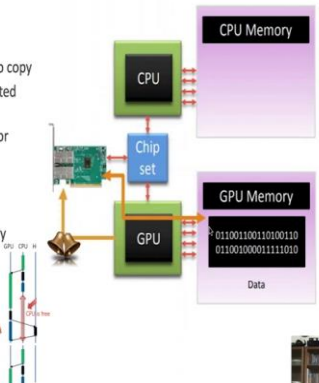
GPUDirect RDMA Evolution (cont.)




- **GPUDirect RDMA Async**
 - GPU memory is exposed to RDMA NIC
 - Direct data path from GPU to network - Data path is zero copy
 - CPU is involved in WQE preparation and release completed WQEs
 - GPU is involved in Ring Doorbell, Handles completions for incoming packets to GPU
- **Zero copy**
 - RDMA device sends data to network from GPU memory
 - RDMA device receive data from network to GPU memory
- **Reduce CPU utilization**

```

while(1) {
    gpu_kernel <<<... stream>>>(buf);
    gpu_stream_queue_send(stream, qp, buf);
    gpu_stream_wait_completion(qp);
}
                    
```

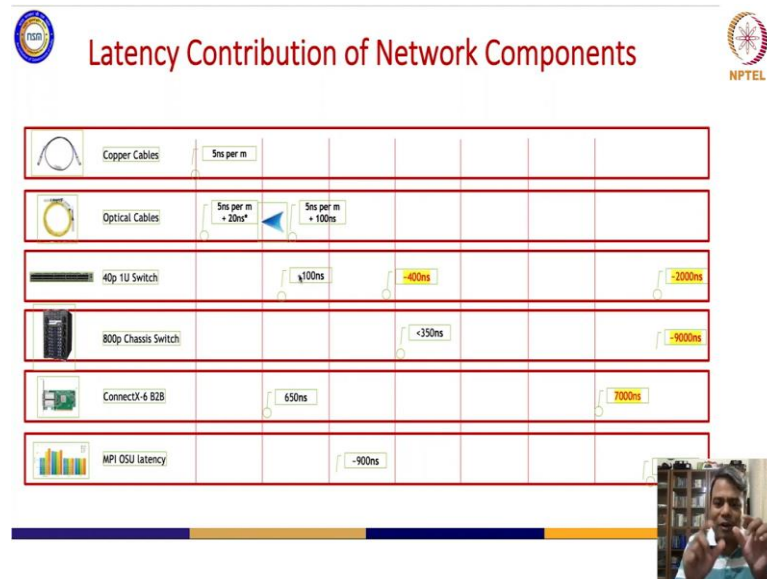




There are you know I talked about like there needs to be a control plane, which ensures that data corruption does not happen when two, three devices are simultaneously accessing or you know when to packetize and what to send, so there is a control plane. Earlier the control plane was actually controlled by the CPU, so and only the data plane was accelerated.

Now, with the now the latest technology that you will see in the GPU clusters that you are actually being exposed to even the control plane is running inside the GPU in sync with the adapter. So, there is no involvement at all of the CPU for GPU to GPU communication. And therefore, we end up making a very very efficient GPU accelerated cluster of multiple hundreds and thousands of nodes, which we will talk about in the coming slides ok.

(Refer Slide Time: 22:35)



So, just to conclude this part of the presentation I think we are 5 minutes over a just to conclude this part of the presentation we talked about the fundamentals, we talked about latencies, we talked about bandwidth, we talked about what are the limits to this bandwidth we. So, let me just give a quick picture of latency disk you know why are we doing this RDMA, GPU RDMA?

Because we want to have as low latency as possible between all communicating compute elements whether be it accelerated it computes with GPU or standard x86 CPU or ARM CPUs right, we want to reduce the latency. Latency is a combination of all these factors, it is a combination of the cables which are there in the network, it is a combination of the switch and the topology I am showing here a big switch which contains multiple individual switches in a particular topology. It is a combination of how fast the network adapter on the endpoint is able to look at the packet and do a DMA operation.

And eventually what a programmer sees is the MPI latency, the message passing latency or the communication framework latency in when you take the example of you know the communication libraries for maybe a you know Tensorflow or a PyTorch right. So, just to summarize the latency of the media is actually 4 to 5 nanoseconds per meter and it is a it is a hard limit why speed of light ok.

Light travels at a certain rate your 3 lakh kilometers a second which corresponds to certain nanoseconds per meter in a media the speed of light or the speed of the

electromagnetic wave is defined by the speed of light divided by the refractive index, if it is an optical media or divide by the permittivity if it is a standard dielectric media which corresponds to usually 5 nanoseconds per meter its physics you cannot change it.

So, if you have a very large cluster ok, people sometimes ask me very naively that ok why cannot I have a very large supercomputing cluster by combining all the computers in a city you can do that. You can run problems which are embarrassingly parallel which you can divide amongst all the compute elements and they do not need to talk to each other so frequently. You can do that and you say ok I have got a big system, but when you are looking at scientific applications and we are looking at you know coupled applications you need to talk very fast ok.

And that is where if you make clusters which are very large you actually keep on adding latencies when you look at optical cables you add some more latency for the electro optical and opto-electrical conversion back, when you make a large cluster you will have to have multiple of these switching elements as we talked about. The switching elements have certain amount of latencies between them I said 100 nanoseconds 130 nanoseconds right.

Then there is this transport layer which presents the latency of around 600 odd nanoseconds. Again I am showing the numbers just to give you a practical web I am showing you the numbers which are there for a high performance interconnect like InfiniBand the highlighted numbers are what you see in a standard network like Ethernet ok. Again physics does not differentiate between high performance network or Ethernet or whatever.

So, the latency of the media is the same for both the protocols, but the other latencies starts differing, why? Because if you take a switch Ethernet has a protocol which is designed for the World Wide Web so, it contains a lot of complexities in its headers, it contains lot of things like you know you have IP addresses, you have you know protocol, port numbers and you have so many quality of service things v laning whatever right because that is how Ethernet was designed.

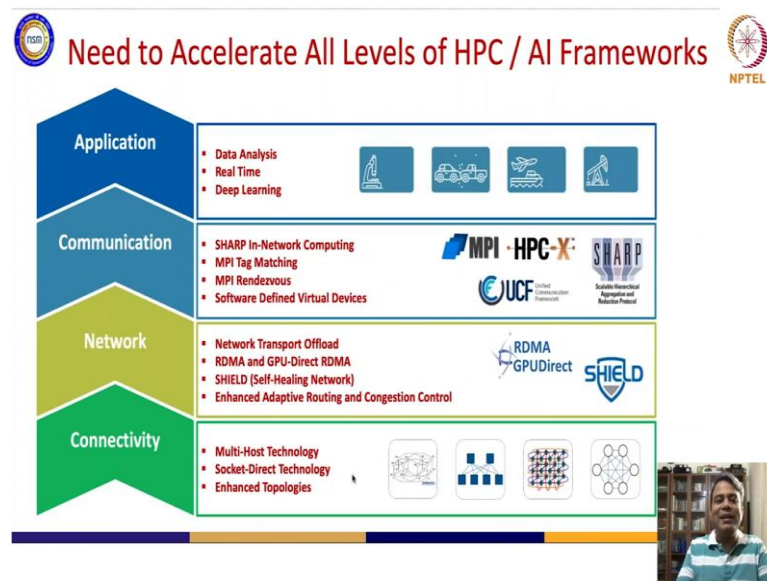
Therefore when an Ethernet switch receives a packet to take a decision the time it takes to make a decision of where to send depends on the complexity of all the headers. In a high performance network you are not connecting it to World Wide Web it is for a very vast

set of computers within a data center and therefore, you can do away with the complexities of all the other headers. And therefore, the switch can take a decision very fast and therefore; the switch has very low latency ok

TCP since you are allowed to drop packets you do not do a DMA operation you have 7000 nanoseconds of latency when you want to do a single byte transfer from one computer to the other whereas, high performance networks RDMA enabled networks are one tenth of that.

So, with this slide I am just showing the differences between you know the if somebody says ok what is the difference between a standard Ethernet with that RJ 45 versus a supercomputer that is in CDAC let us say you know some PARAM Siddhi or PARAM Yuva that is in CDAC. Well it contains right up from the hardware to the networking equipment to the software.

(Refer Slide Time: 28:00)



It contains elements which are highly tuned and designed for high performance clustering and communication ok. So, with that I will stop here and then.