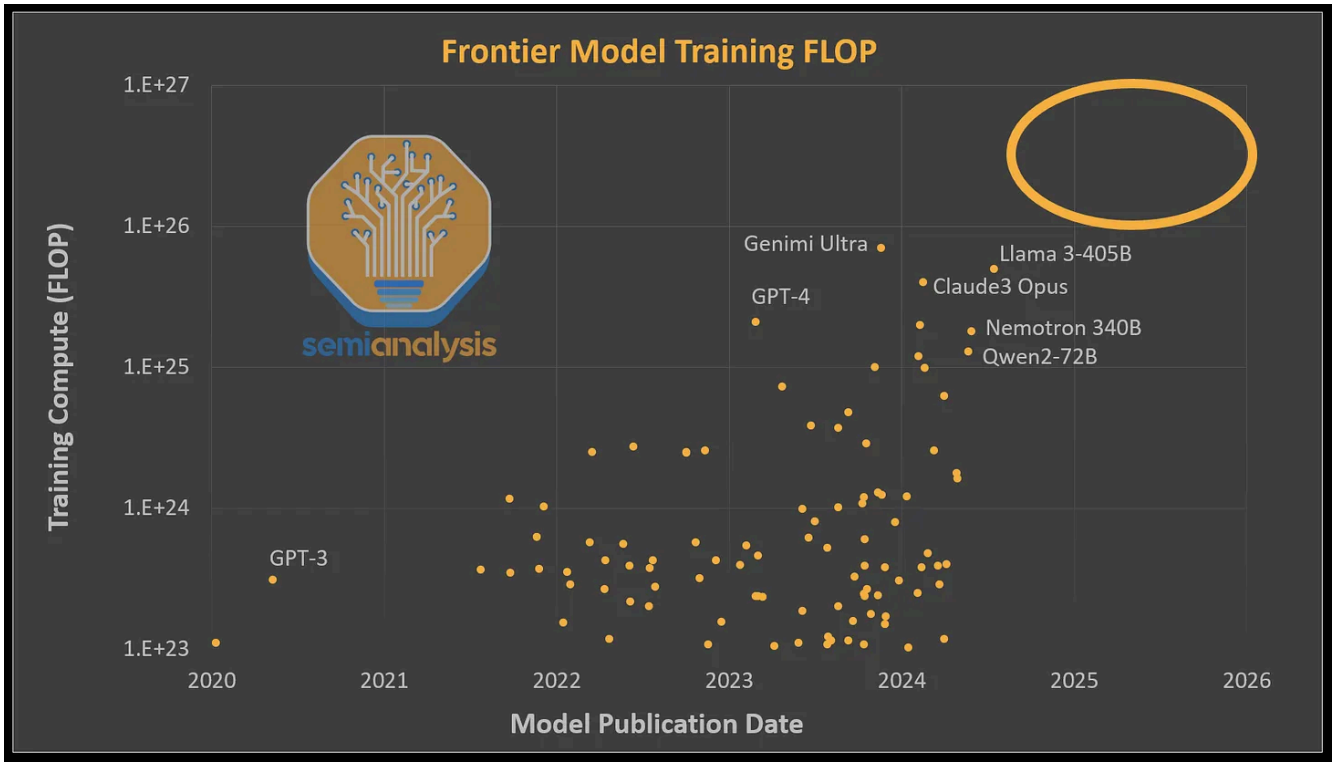
100,000 H100 Clusters: Power, Network Topology, Ethernet vs InfiniBand, Reliability, Failures, Checkpointing

Frontier Model Scaling Challenges and Requirements, Fault Recovery through Memory Reconstruction, Rack Layouts

DYLAN PATEL AND DANIEL NISHBALL

JUN 17, 2024 • PAID

There is a camp that feels AI capabilities have stagnated ever since GPT-4's release. This is generally true, but only because no one has been able to massively increase the amount of compute dedicated to a single model. Every model that has been released is roughly GPT-4 level (~2e25 FLOP of training compute). This is because the training compute dedicated to these models have also been roughly the same level. In the case of Google's Gemini Ultra, Nvidia Nemotron 340B, and Meta LLAMA 3 405B, the FLOPS dedicated were of similar magnitude or even higher when compared to GPT-4, but an inferior architecture was utilized, resulting in these models falling short of unlocking new capabilities.



Source: SemiAnalysis Estimates

While OpenAI has gotten more compute, they have mostly directed it at bringing smaller, overtrained, cheaper to inference models such as GPT-4 Turbo and GPT-4o. OpenAI admits they only just started training the next tier of model recently.

The obvious next step for AI is to train a multi-trillion parameter multimodal transformer with massive amounts of video, image, audio, and text. No one has completed this task yet, but there is a flurry of activity in the race to be first.

Multiple large AI labs including but not limited to OpenAI/Microsoft, xAI, and Meta are in a race to build GPU clusters with over 100,000 GPUs. These individual training clusters cost in excess of $4 billion of server capital expenditures alone, but they are also heavily limited by the lack of datacenter capacity and power as GPUs generally need to be co-located for high-speed chip to chip networking. A 100,000 GPU cluster will require >150MW in datacenter capacity and guzzle down 1.59 terawatt hours in a single year, costing $123.9 million at a standard rate of $0.078/kWh.

100k H100 GPU Cluster - Annual Electricity Costs		
Region	Tariff (USD/kWh)	Annual Cost $M
USA - Average	$0.083	$131.9
USA - North Dakota	$0.074	$117.6
USA - Utah	$0.068	$108.0
USA - Arizona	$0.078	$123.9
USA - ND Wind PPA	$0.033	$52.4
USA - Solar PPA (CAISO)	$0.033	$52.4

Source: SemiAnalysis, US EIA

Today we will dive into large training AI clusters and the infrastructure around them. Building these clusters is a lot more complicated than just throwing money at the problem. Achieving high utilization with them is even more difficult due to the high failure rates of various components, especially networking. We will also walk through power challenges, reliability, checkpointing, network topology options, parallelism schemes, rack layouts, and total bill of materials for these systems. Over a year ago, we covered Nvidia's InfiniBand problem which resulted in some companies choosing Spectrum-X Ethernet over InfiniBand. We will also cover the major flaw with Spectrum-X which has hyperscalers going with Broadcom's Tomahawk 5.

To put in perspective how much compute a 100,000 GPU cluster can provide, OpenAI's training BF16 FLOPS for GPT-4 was ~2.15e25 FLOP (21.5 million ExaFLOP), on ~20,000 A100s for 90 to 100 days. That cluster only had 6.28 BF16 ExaFLOP/second peak throughput. On a 100k H100 cluster, this number would soar to 198/99 FP8/FP16 ExaFLOP/second. This is a 31.5x increase in peak theoretical AI training FLOPs compared to the 20k A100 cluster.

Total Cluster ExaFLOPs			
AI Data Type	100k H100	20k A100	Improvement
INT8	197.9	12.6	15.8x
FP8	197.9	6.3	31.5x
BF16/FP16	98.9	6.3	15.7x
TF32	49.5	3.1	15.8x

Source: Nvidia, SemiAnalysis

On H100s, AI labs are achieving FP8 Model FLOPs Utilization (MFU) as high as 35% and FP16 MFU of 40% on trillion parameter training runs. As a recap, MFU is a measure of the effective throughput and utilization of the peak potential FLOPS after taking into account overhead and various bottlenecks such as (power limits, communication flakiness, recomputation, stragglers, and inefficient kernels). A 100,000 H100 cluster would only take four days using FP8 to train GPT-4. On a 100k H100 cluster training run for 100 days, you can achieve an effective FP8 Model FLOP of ~6e26 (600 million ExaFLOP). Note that the poor reliability of hardware reduces MFU significantly.

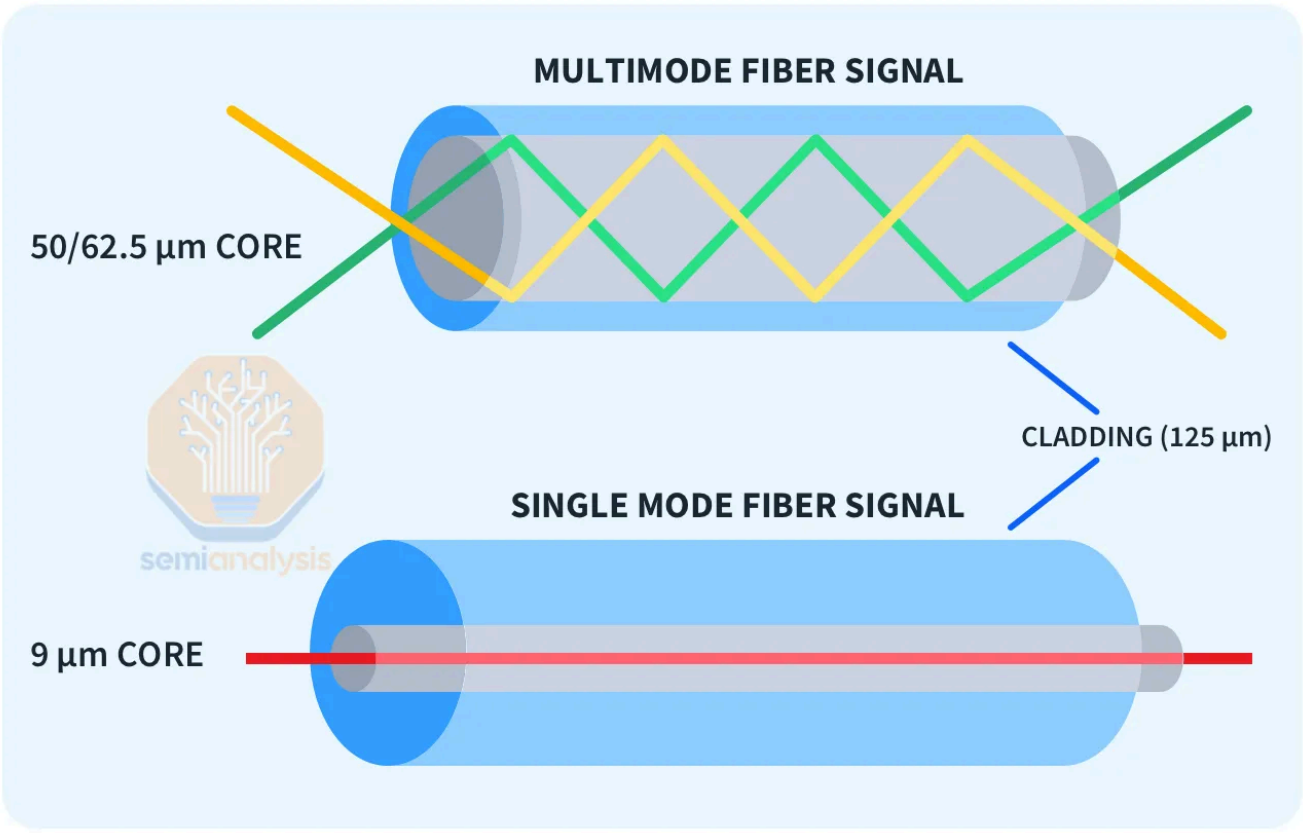
Power Challenges

The critical IT power required for a 100k H100 cluster is ~150MW. While the GPU itself is only 700W, within each H100 server, CPUs, Network Interface Cards (NICs), Power Supply Units (PSUs), account for a further ~575W per GPU. Other than the H100 servers, an AI cluster requires a collection of storage servers, networking switches, CPU nodes, optical transceivers, and many other items that together account for another ~10% in IT power. Putting into perspective how much power ~150MW is, the largest national lab supercomputing, El Capitan only requires 30MW of critical IT power. Government supercomputers pale in comparison to industry.

One major power challenge is that currently no single datacenter building has the capacity for a new ~150MW deployment. When people refer to 100k GPU clusters, generally they mean on a single campus, not building. The search for power is so dire,

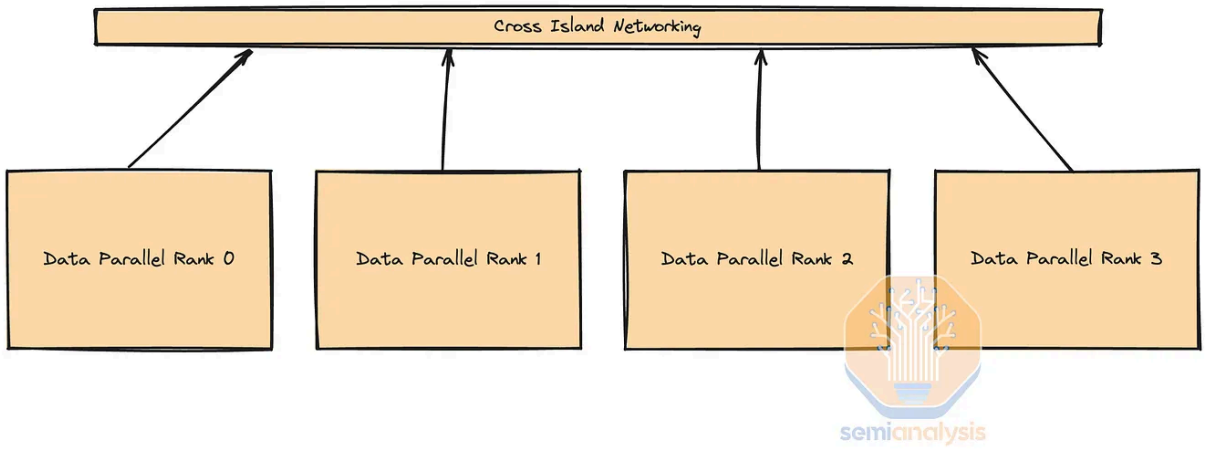
X.AI is even converting an old factory in Memphis Tennessee into a datacenter due to the lack of other options.

These clusters are networked with optical transceivers, which come on a sliding scale of cost vs reach. Longer range “single mode” DR and FR transceivers which can reliably transmit signals ~500 meters to ~2km but can cost 2.5x as much as “multi-mode” SR and AOC transceivers which only support only up to ~50 meters reach. Furthermore, campus level “coherent” 800G transceivers with over 2km of range also exist albeit at 10x+ higher pricing.



Small clusters of H100’s generally connect every GPU at 400G to every other GPU with only multi-mode transceivers through just a layer or two of switches. With large clusters of GPUs, more layers of switching must be added, and the optics becomes exorbitantly expensive. The network topology of such a cluster will differ broadly based on preferred vendor, current and future workloads, and capex.

Each building will generally contain a pod or multiple pods of compute connected with cheaper copper cables or multi-mode transceivers. They will then use longer range transceivers in order to interconnect between “islands” of compute. The image below shows 4 islands of compute that have high bandwidth within the island but lower bandwidth outside the island. 155MW is challenging to deliver in a single location, but we are tracking over 15 Microsoft, Meta, Google, Amazon, Bytedance, X.AI, Oracle, etc data center build outs that will have that much space for AI servers and networking.



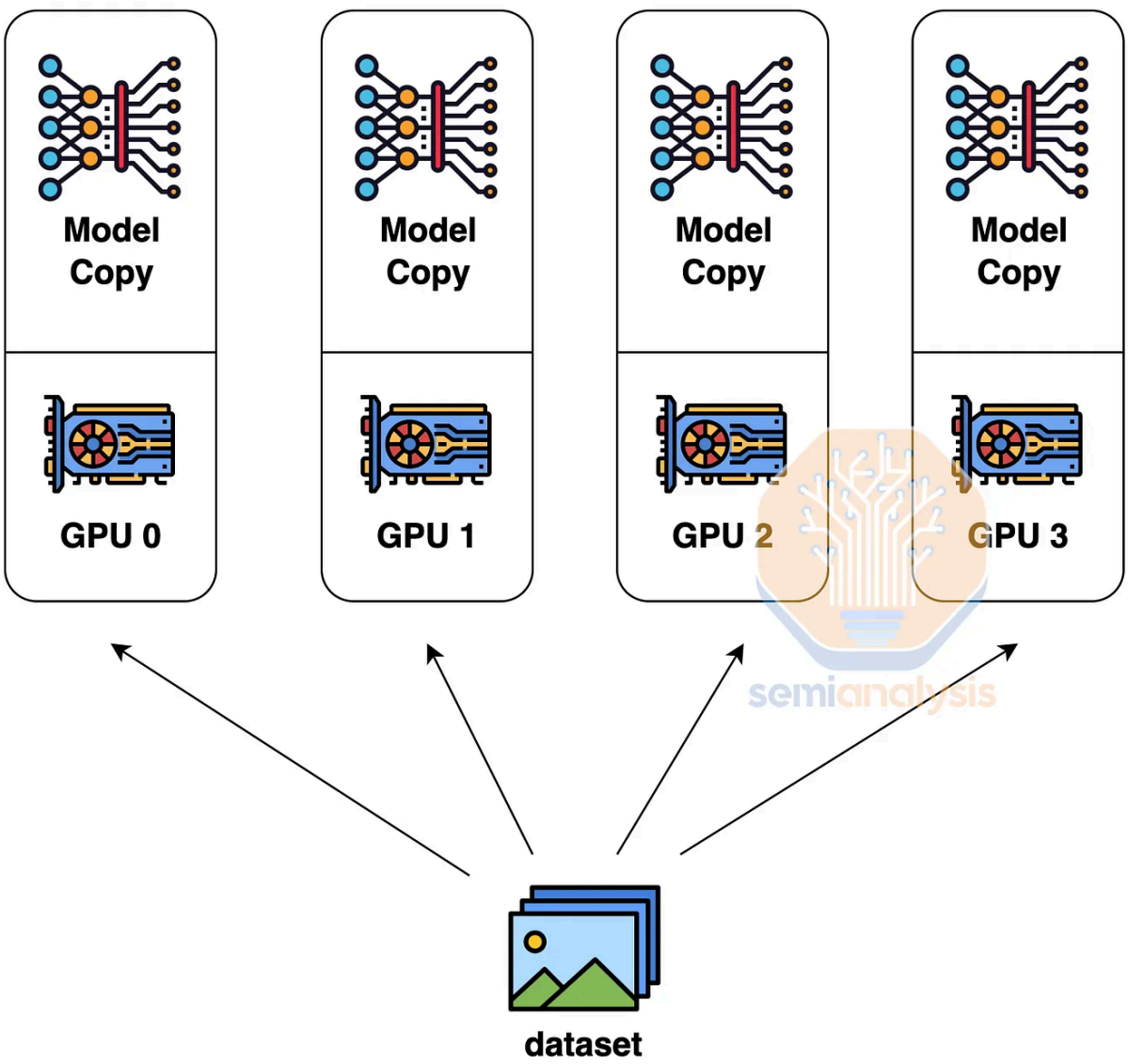
Source: SemiAnalysis

Different customers choose different network topologies based on several different factors such as data haul infrastructure, cost, maintainability, power, current, future workloads etc. As such, some customers are choosing Broadcom Tomahawk 5 based switches, others are sticking to Infiniband, while others are choosing NVIDIA Spectrum-X. We will dive into why below.

Parallelism Refresher

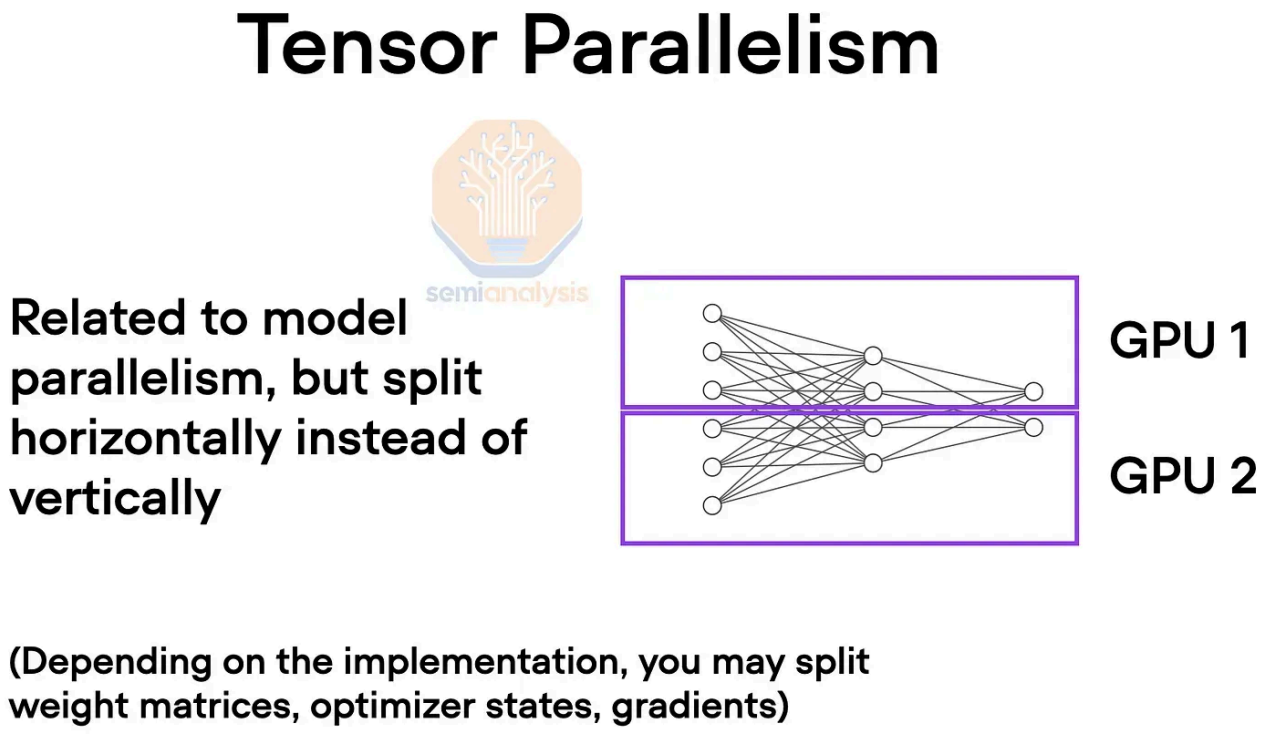
To understand network design, topology, reliability concerns, and checkpointing strategies, we will first do a quick refresher on the 3 different types of parallelism used in trillion parameter training - Data Parallelism, Tensor Parallelism, and Pipeline Parallelism. We have a comprehensive explanation of parallelism here.

Data Parallelism is the simplest form of parallelism in which each GPU holds the entire copy of the model weights and each GPU (rank) receives a different subset of the data. This type of parallelism has the lowest level of communication since just the gradients needs to be summed up (all reduce) between each GPU. Unfortunately, data parallelism only works if each GPU has enough memory to store the entire model weights, activations, optimizer state. For a 1.8 trillion parameter model like GPT-4, just the model weights and optimizer state can take as much as 10.8 Terabytes of memory for training.



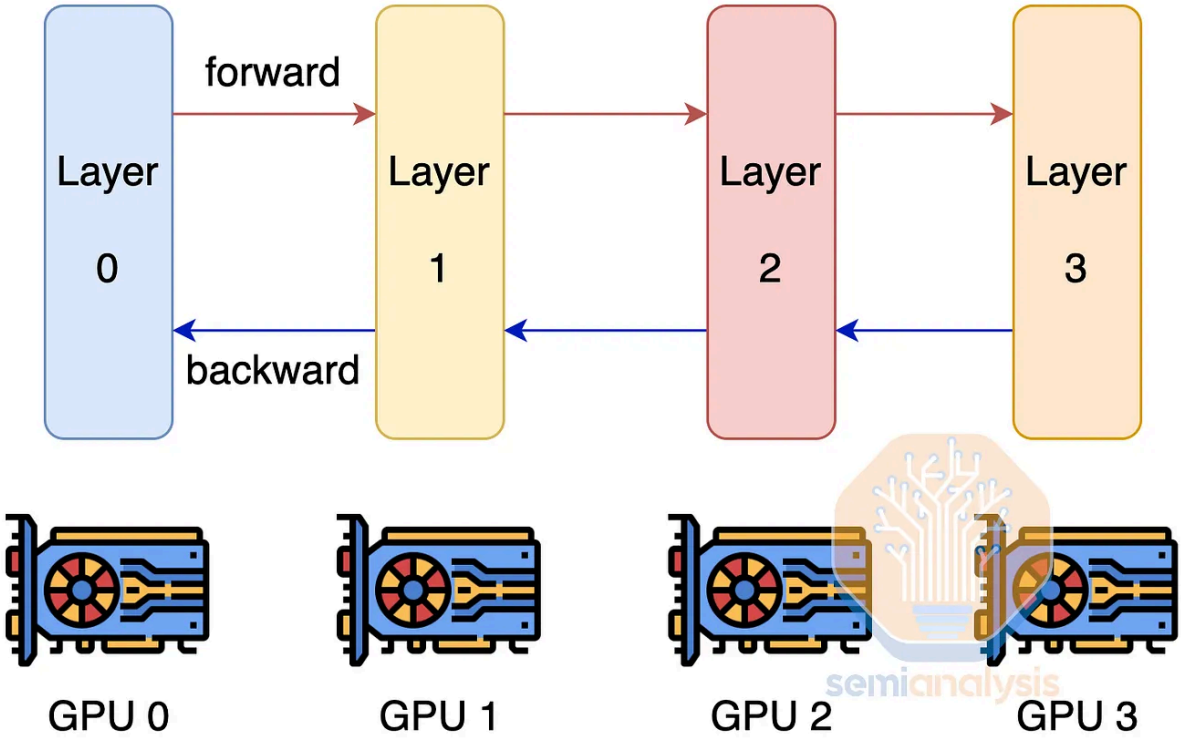
Source: ColossalAI

In order to overcome these memory constraints, tensor parallelism is used. In tensor parallelism, every layer has its work and model weights distributed across multiple GPUs generally across the hidden dimension. Intermediate work is exchanged via all-reductions across devices multiple times across self-attention, feed forward network, and layer normalizations for each layer. This requires high bandwidth and especially needs very low latency. In effect, every GPU in the domain works together on every layer with every other GPU as if there were all one giant GPU. Tensor parallelism reduces the total memory used per GPU by the number of tensor parallelism ranks. For example, it is common to use 8 tensor parallelism ranks today across NVLink so this will reduce the used memory per GPU by 8.



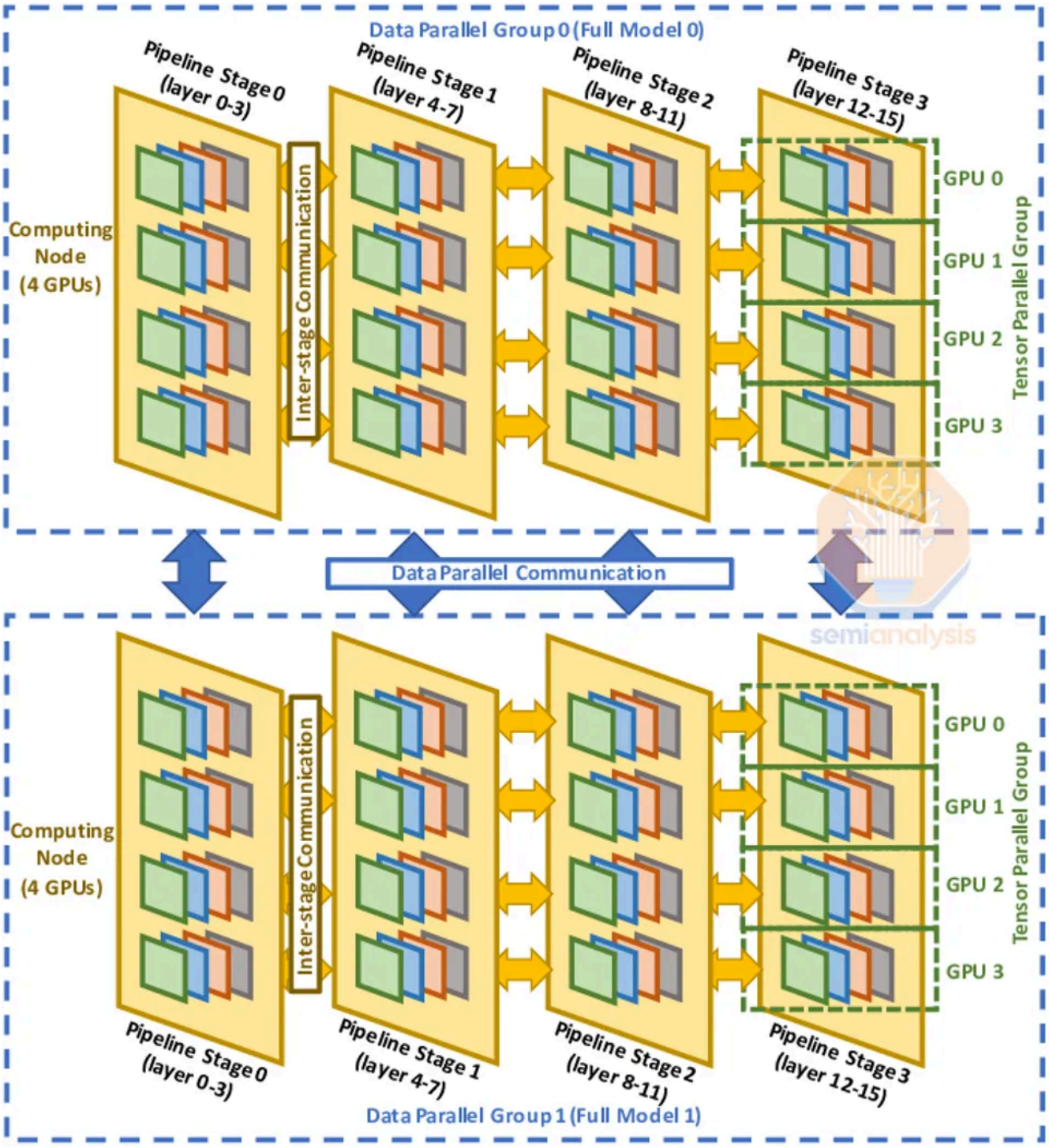
Source: Accelerating Pytorch Training

Another technique to overcome the challenges of each GPU not having enough memory to fit the model weights and optimizer state is by using pipeline parallelism. With Pipeline Parallelism, each GPU only has a subset of the layers and only does the computation for that layer and passes the output other the next GPU. This technique reduces the amount of memory needed by the number of pipeline parallelism ranks. Pipeline parallelism has heavy communication volume requirements, but not as heavy as tensor parallelism.



Source: ColossalAI

In order to maximize Model FLOP Utilization (MFU), companies generally combine all three forms of parallelism to form 3D Parallelism. Then they apply Tensor Parallelism to GPUs within the H100 server, then use Pipeline Parallelism between nodes within the same Island. Since Data Parallelism has the lowest communication volume and the networking between islands is slower, data parallelism is used between islands.



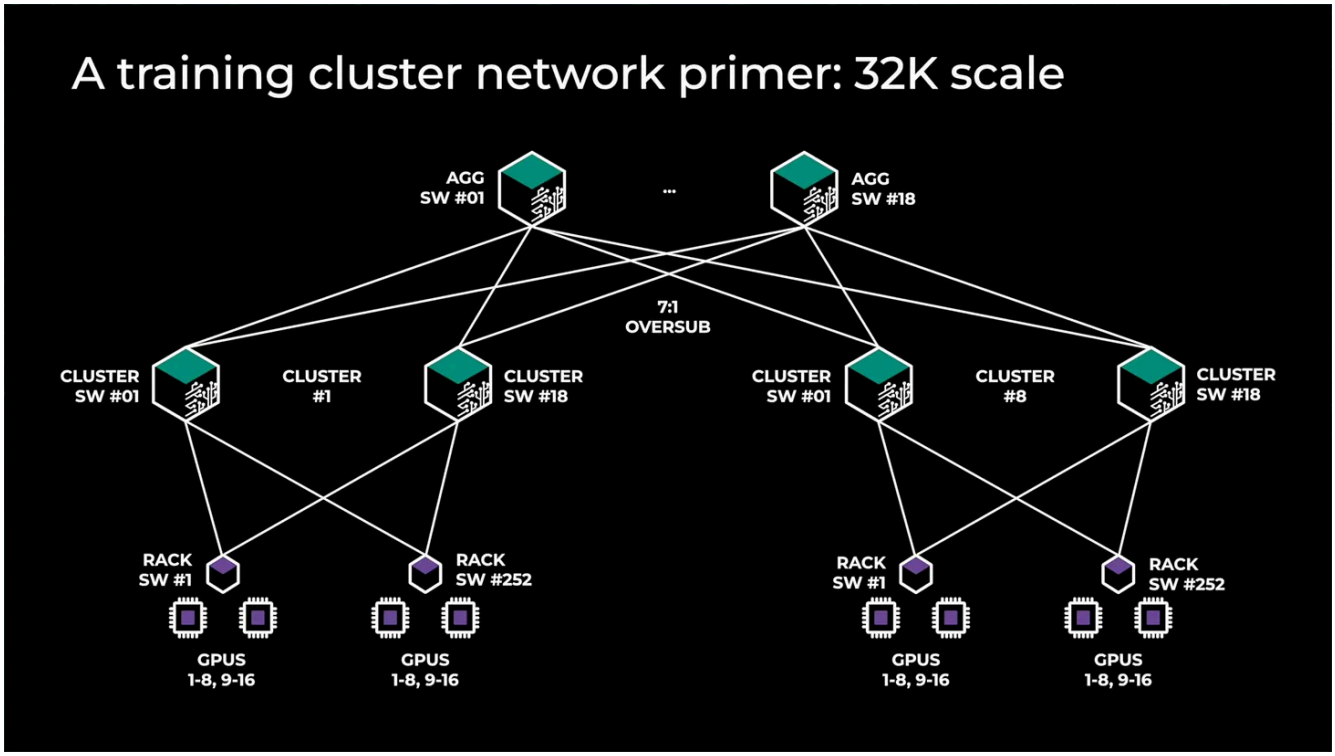
Source: Optimus-CC

Whole techniques like FSDP are common at small GPU world sizes for very large models, it doesn't work. It's effectively incompatible with FSDP too.

Network Design Considerations

Networks are designed with parallelism schemes in mind. If every GPU connected to every other GPU at max bandwidth in a fat tree topology, then costs would be exorbitant as 4 layers of switching would be required. The cost of optics would soar as every additional layer of networking requires optics in between.

As such, noone deploys full fat tree architectures for large GPU clusters. Instead, they rely on making islands of compute that have full fat tree architectures alongside lesser bandwidth between these islands. There are a variety of ways to do this, but most firms are choosing to "oversubscribe" the top layer of networking. For example see Meta's last generation architecture for GPU clusters up to 32,000. There are 8 total islands with full fat bandwidth between them, then another layer of switching on top that has 7:1 over subscription. The networking between islands is 7x slower compared to the networking within the island.



Source: Meta

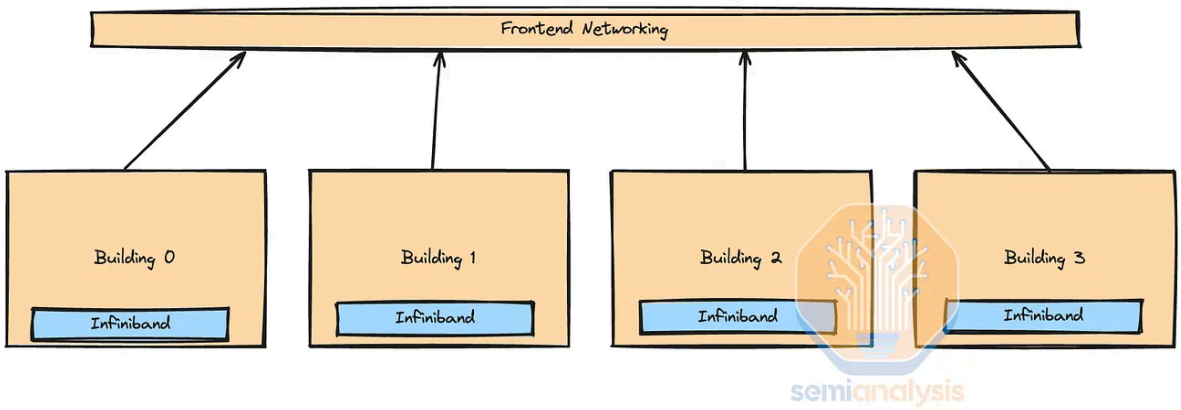
GPU deployments have multiple networks, frontend, backend, and scale up (NVLink). In some cases, you will run different parallelism schemes across each. The NVLink network may be the only one fast enough for the bandwidth requirements of tensor

parallelism. Your backend can generally handle most other types of parallelism easily, but if there is oversubscription, one often is relegated to data parallelism only.

Furthermore, some folks do not even have islands with bandwidth oversubscribed at the top layer. Instead, they use the move out of the backend network into frontend networking.

Hybrid InfiniBand and Frontend Ethernet Fabric

One major firm trains across multiple InfiniBand islands with frontend Ethernet. This is because the cost for frontend networking is much cheaper and can utilize existing datacenter campus networking between buildings and region routing.

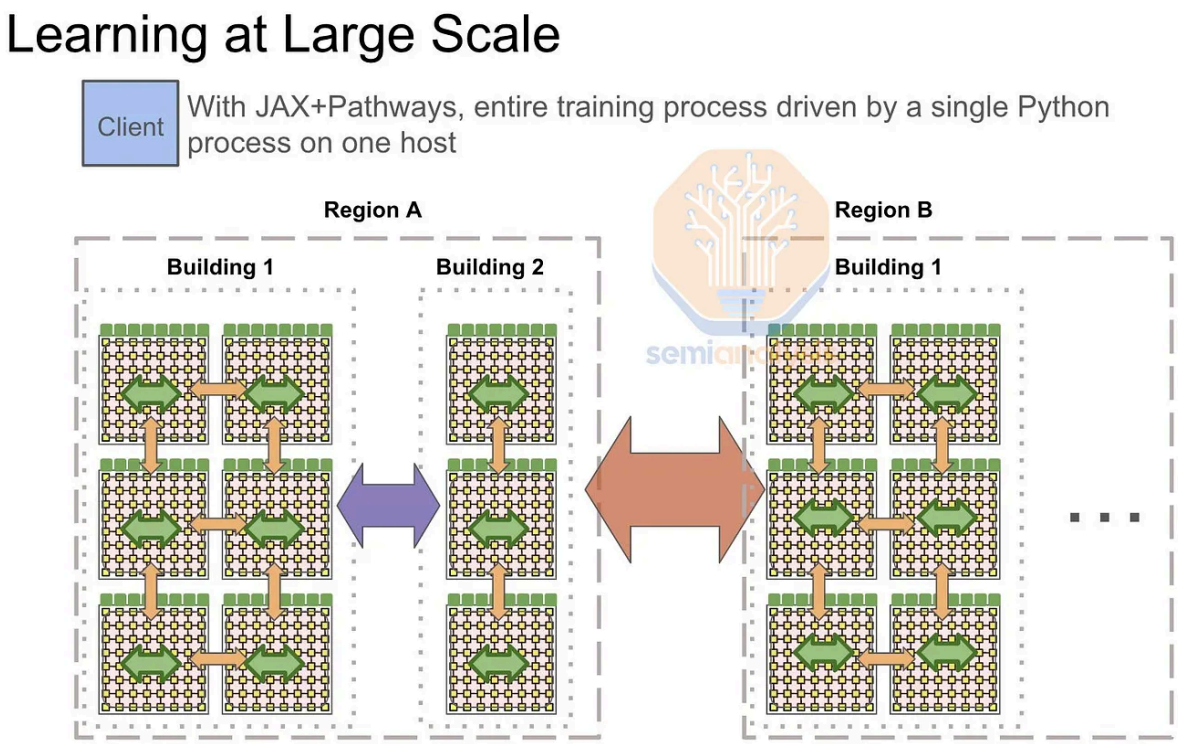


Source: SemiAnalysis

Unfortunately, as model size grow faster due to sparse techniques like MoE, the amount of communication volume that the frontend network needs to handle grows too. This tradeoff must be carefully optimized or else you will end up having two networks that cost the same, as the frontend networking bandwidth will ending up growing so large that it may match the backend networking bandwidth.

It should be noted that Google exclusively uses front end networking for their multi-TPU pod training runs. Their “compute fabric” known as ICI only scales up to a maximum of 8960 chips with costly 800G optics and optical circuit switches connecting

each 64 TPU watercooled rack. As such Google must compensate by making the TPU front end network beefier than most GPU front end networks.

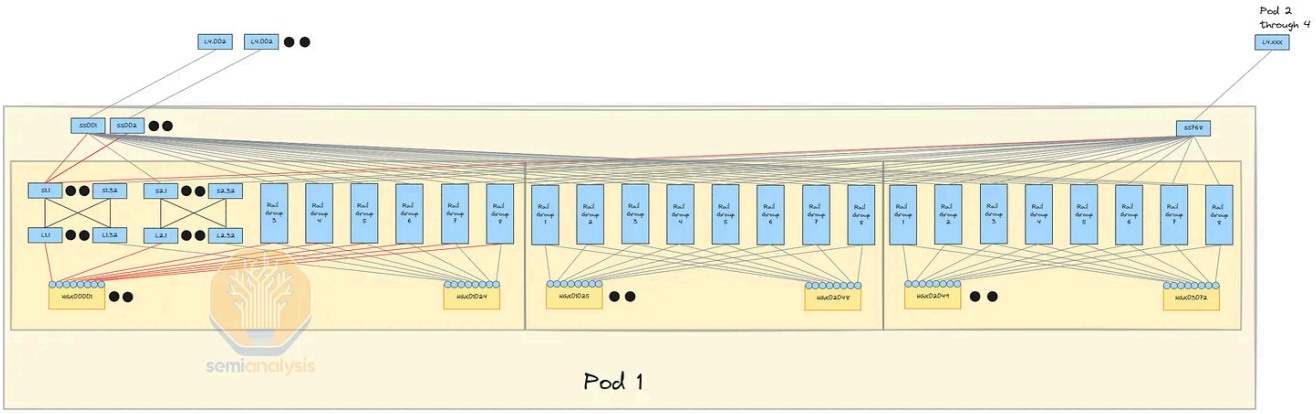


Source: Google at MLSys24

When frontend networks are utilized during training, network topology aware global all-reduce between islands must be done. First each pod or island will perform a local reduce-scatter within the pod InfiniBand or ICI networking, this will leave each GPU/TPU having the sum for a subsection of the gradient. Next, a cross pod all-reduce between each host rank will be performed using a frontend ethernet networking, then finally each pod would perform a pod-level all-gather.

The frontend networking is also responsible for loading in data. As we move towards multimodal image and video training data, front-end networking requirements will increase exponentially. In such a situation, the front-end networking bandwidth will be fighting between loading large video files and doing all reductions. Furthermore, your straggler problem increases as if there is irregular storage network traffic, it would cause your entire all-reduces to slow down and not be predictively modeled.

The alternative would be a 4 tier InfiniBand network with a 7:1 oversubscription with 4 pods with each pod having 24,576 H100s with a non-blocking 3 tier system. This allows for much greater flexibility for future bandwidth increases compared to using your frontend networking as it is way easier to add more fiber optics transceivers from a switch in building A to another switch in building B compared to doing a full frontend network NIC upgrade in every single chassis of the cluster to upgrade it from 100G to 200G, etc.



Source: SemiAnalysis

This creates a more stable network pattern as your frontend network can solely focus on loading data and checkpointing, and your backend network can solely focus on GPU to GPU communication. This also helps with the stragglers problem. But unfortunately, a 4 tier Infiniband network is extremely expensive due to all the additional switches and transceivers needed.

Rail Optimized vs Middle of Rack

To improve maintainability and increase the use of copper networking (< 3 meters) and multimode networking (< 50 meters), some customers are opting to ditch the NVIDIA recommended rail optimized design and instead of opting to do a Middle of Rack designs.

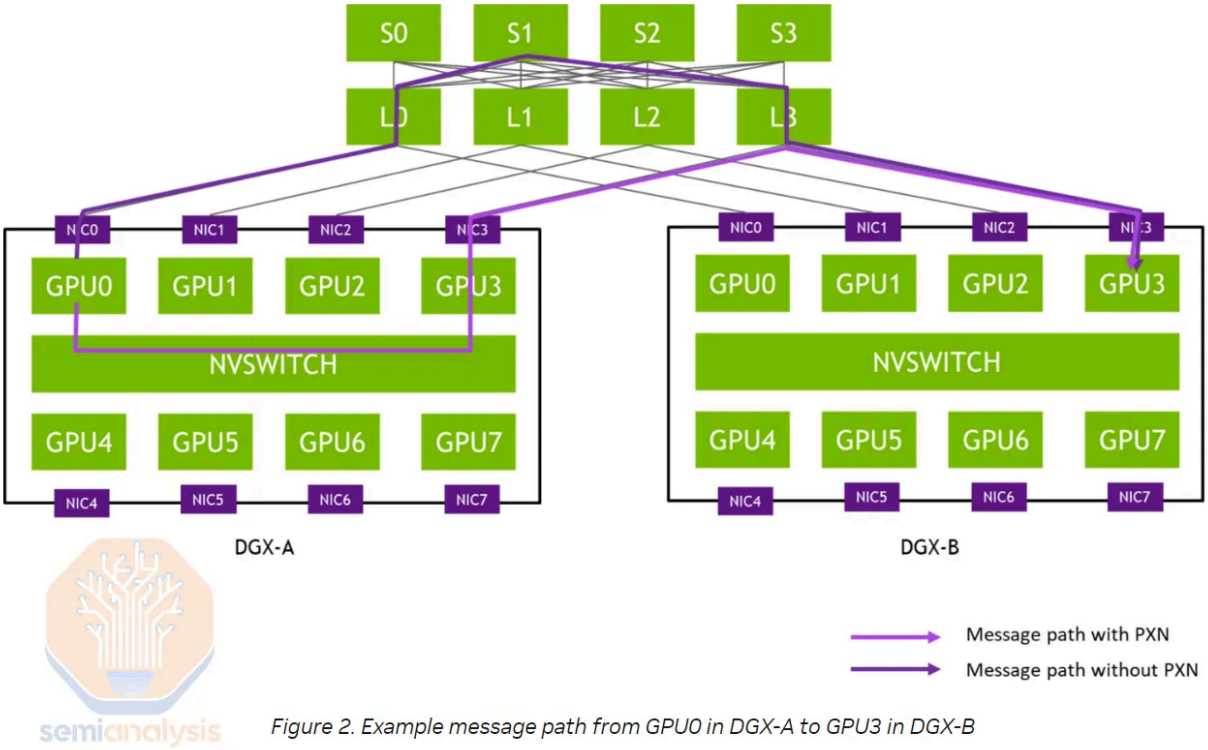
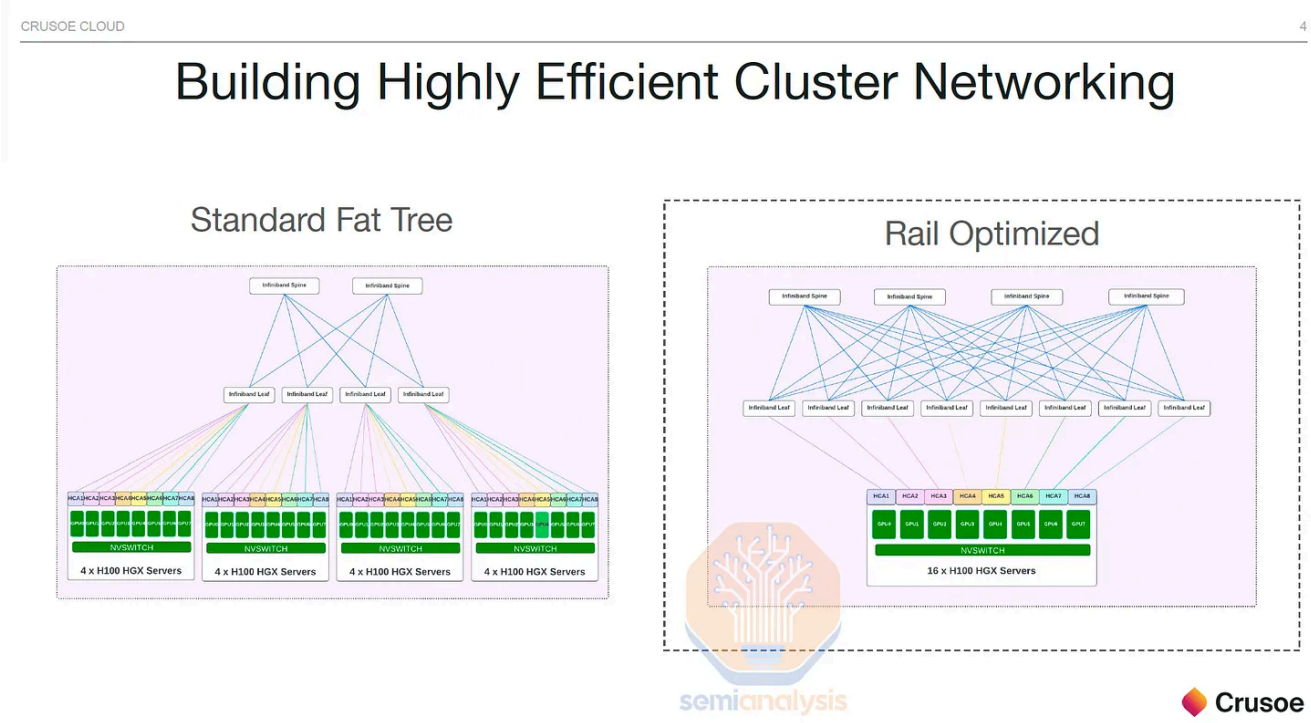


Figure 2. Example message path from GPU0 in DGX-A to GPU3 in DGX-B

Source: Nvidia

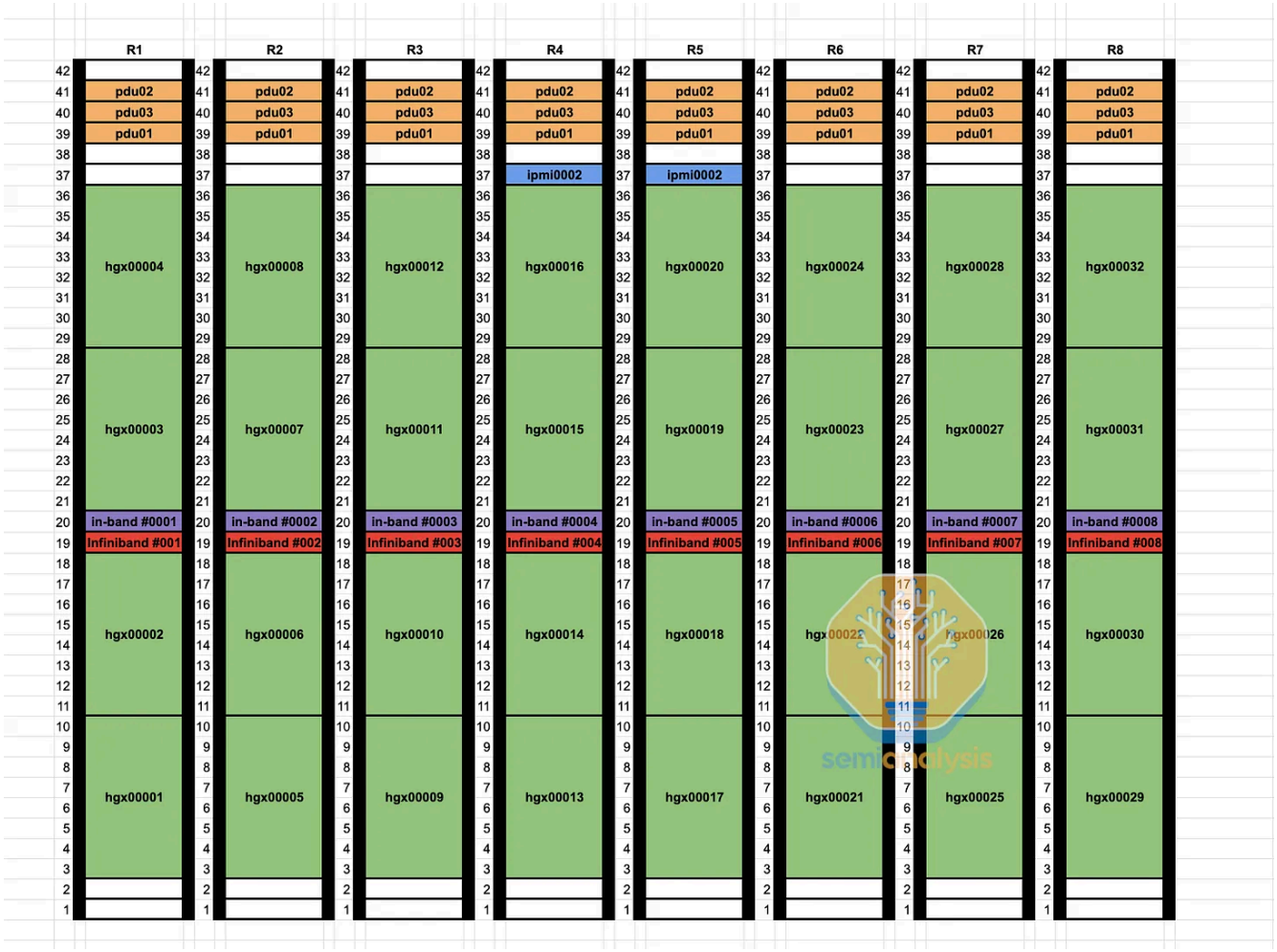
Rail optimized is a technique to have each H100 server connect to 8 different leaf switches (instead of all connecting to the same middle of rack switch) such that each GPU can talk to more distant GPUs with only 1 switch hop. This allows an increase in real world all to all collective performance increase. All-to-All collective communication is used heavily in mixture of experts (MoE) expert parallelism.



Source: Crusoe

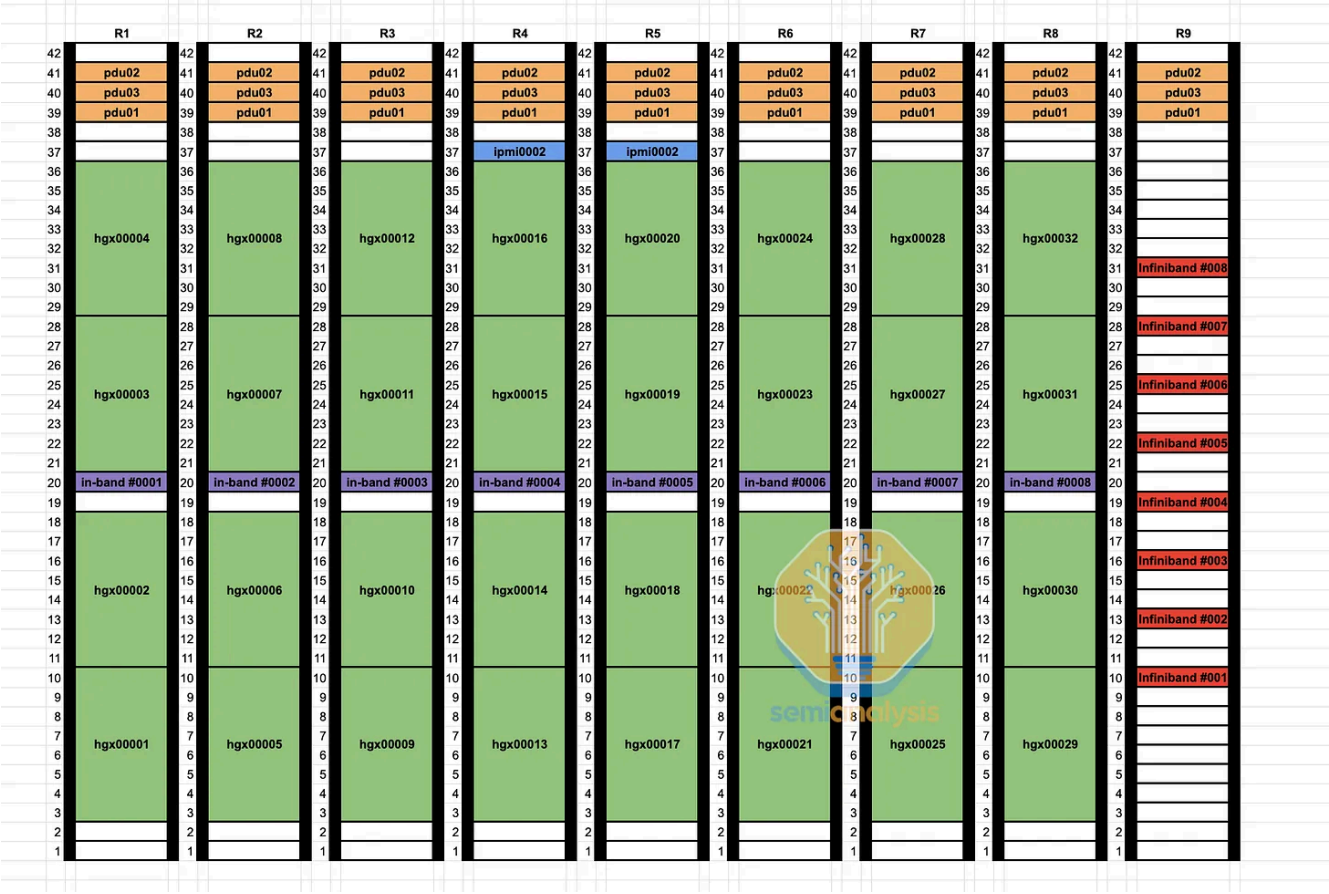
The downside of rail optimized designs is that you must connect to different leaf switches of varying distance as opposed to one middle of rack switch in close proximity to all 8 GPUs in the server. When the switch can be placed in the same rack, Passive Direct Attached Cable (DAC) and Active Electrical Cables (AEC) can be used, but in rail optimized designs where switches are not necessarily in the same rack, optics must be used. Moreover, the leaf to spine distance may be greater than the 50-meter distance, forcing the use of single-mode optical transceivers.

By using a non rail-optimized design, you can replace 98,304 optical transceivers connecting GPUs to leaf switches with cheap direct attached copper, leading to 25-33% of your GPU fabric being copper. As you can see from the below rack diagram, instead of each GPU to leaf switch connection going up to the cable tray then sideways 9 racks to a dedicated rail optimized leaf switch rack, the leaf switches are now in the middle of the rack allowing for each GPU to use DAC copper cables.



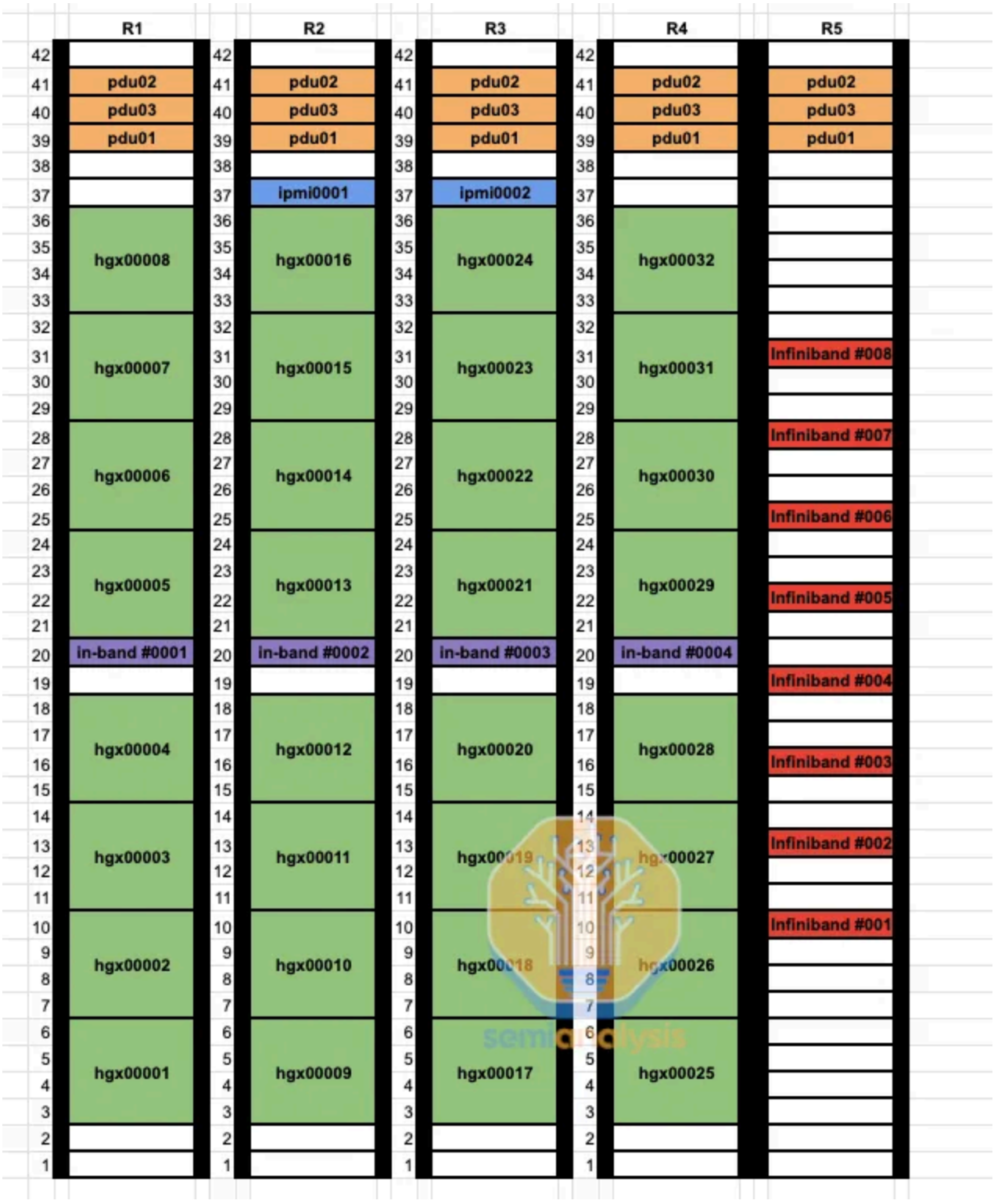
Non-rail optimized middle of rack, Source: SemiAnalysis

DAC copper cables run cooler, use less power, and are much cheaper compared to optics. Since DAC cables run coolers, use less power, and are more reliable, this leads to less flapping (a network link going down intermittently) and failures, which is a major problem for all high-speed interconnects using optics. A Quantum-2 IB spine switch uses 747 Watts when using DAC copper cables. When using multimode optical transceivers, power consumption increases to up to 1,500 Watts.



Rail optimized end of row, Source: SemiAnalysis

Furthermore, initial cabling for a rail optimized design is extremely time consuming for datacenter technicians since the ends of each link are up to 50 meters away and not on the same rack. Compared to a middle of rack design where you have your leaf switch in the same rack as all of your GPUs that connect to the leaf switch. In a middle of rack design, you can even test the compute node to leaf switch links at the integration factory since it is all within the same rack.



Rail optimized end of row water cooled, Source: SemiAnalysis

Reliability and Recovery

Reliability is one of the most important operational concerns with these giant clusters due to the synchronous nature of current frontier training techniques. The most common reliability problems are GPU HBM ECC error, GPUs Drivers being stuck, optical transceivers failing, NICs overheating, etc. Nodes are constantly going down or spitting out errors.

To keep the mean time for fault recovery low and training continuing, datacenters must keep hot spare nodes and cold spare components on site. When a failure happens, instead of stopping the whole training run, it is best to swap in a working spare node that is already on, and continuing training. A lot of the downtime for these servers is simply power cycling/restarting the node and that fixing whatever issue arose.

A simple restart doesn't fix every problem though, and in many cases it requires a datacenter technician to physically diagnosis and replace equipment. In the best case situation, it will take multiple hours for a datacenter technician to fix a broken GPU server, but in many cases it can take days before a broken node can be brought back into to the training run. Broken nodes and spare hot nodes are GPUs that are not actively contributing to the model, despite theoretically having FLOPS to deliver.

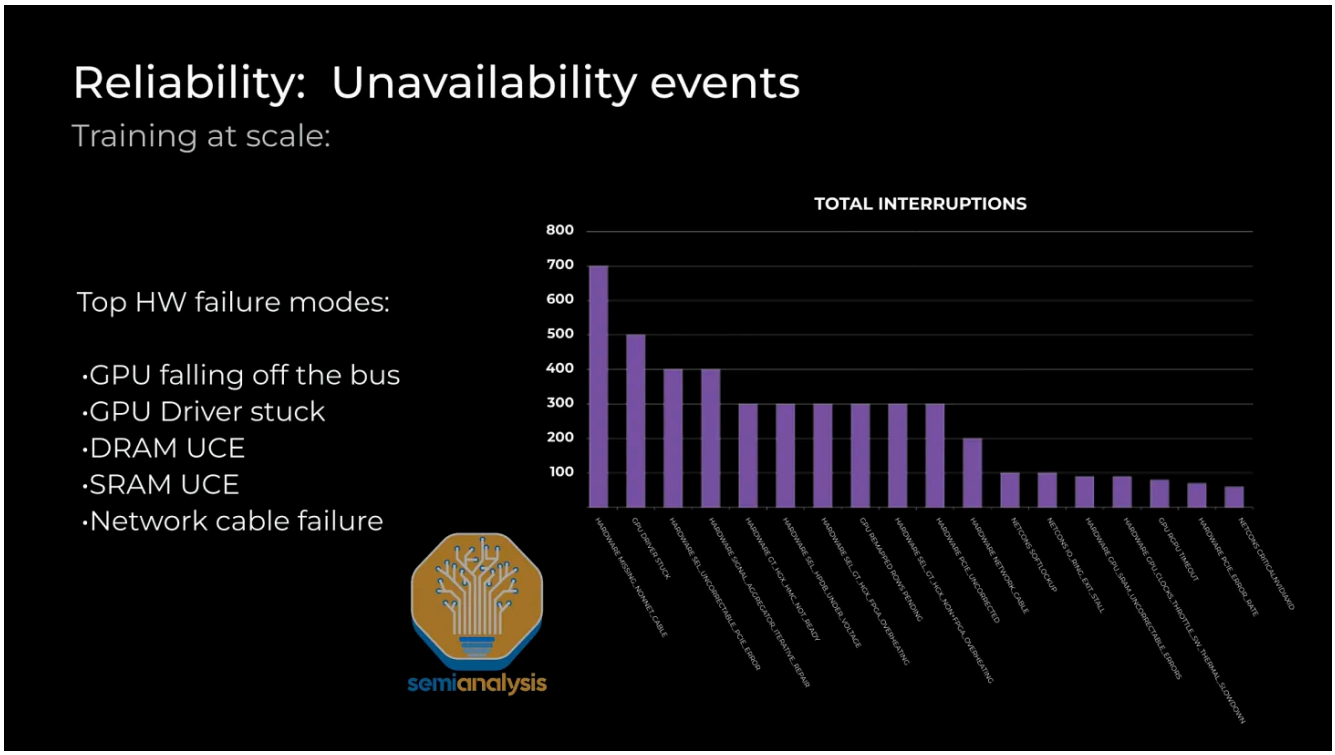
As a model is trained, frequent checkpointing of the model to CPU memory or NAND SSDs is needed in case errors such as HBM ECC happen. When an error occurs, you must reload the weights of the model and the optimizer from the slower tier of memory and restart training. Fault tolerance training techniques, such as Oobleck, can be used to provide user level application driven approaches to deal with GPU and network failures.

Unfortunately, frequent checkpointing and fault tolerance training techniques hurt the overall MFU of the system. The cluster needs to constantly pause to save its current weights to persistent memory or CPU memory. Furthermore, when you reload from checkpoints, you usually only save once every 100 iterations. This means you can only losing at max 99 steps of useful work. On an 100k cluster, if each iteration took 2 seconds, you lose up to 229 GPU days of work from a failure at iteration 99.

The other approach to fault recovery is to have your spare nodes just RDMA copy from other GPUs over the backend fabric. Since the backend GPU fabric is approximately

400Gbps and there are 80GB of HBM memory per GPU, it will take approximately ~1.6 seconds to copy the weights. With this approach, at max you will only lose 1 step (since more GPU HBM will have the most recent copy of the weights), so that would be only 2.3 GPU days' worth of computation + another 1.85 GPU days to RDMA copy the weights from other GPUs HBM memory.

Most leading AI labs have implemented this, but many smaller firms still stick to using the heavy, slow, inefficient technique of restarting from checkpoints for all failures due to simplicity. Implementing fault recovery through memory reconstruction can add multiple percentage points to MFU for a large training run.



Source: Meta

One of the most common problems encountered is Infiniband/RoCE link failure. Even if each NIC to leaf switch link had a mean time to failure rate of 5 years, due to the high number of transceivers, it would only take 26.28 minutes for the first job failure on a brand new, working cluster. Without fault recovery through memory reconstruction, more time would be spent restarting the training run in 100,000 GPU clusters due to optics failures, than it would be advancing the model forward.

Estimated Time to First Job Failure (Minutes)				
Mean Time to Failure Per Link	3 years	4 years	5 years	10 years
Number of GPUs				
10,000	157.7	210.2	262.8	525.6
20,000	78.8	105.1	131.4	262.8
50,000	31.5	42.0	52.6	105.1
100,000	15.8	21.0	26.3	52.6

Source: SemiAnalysis

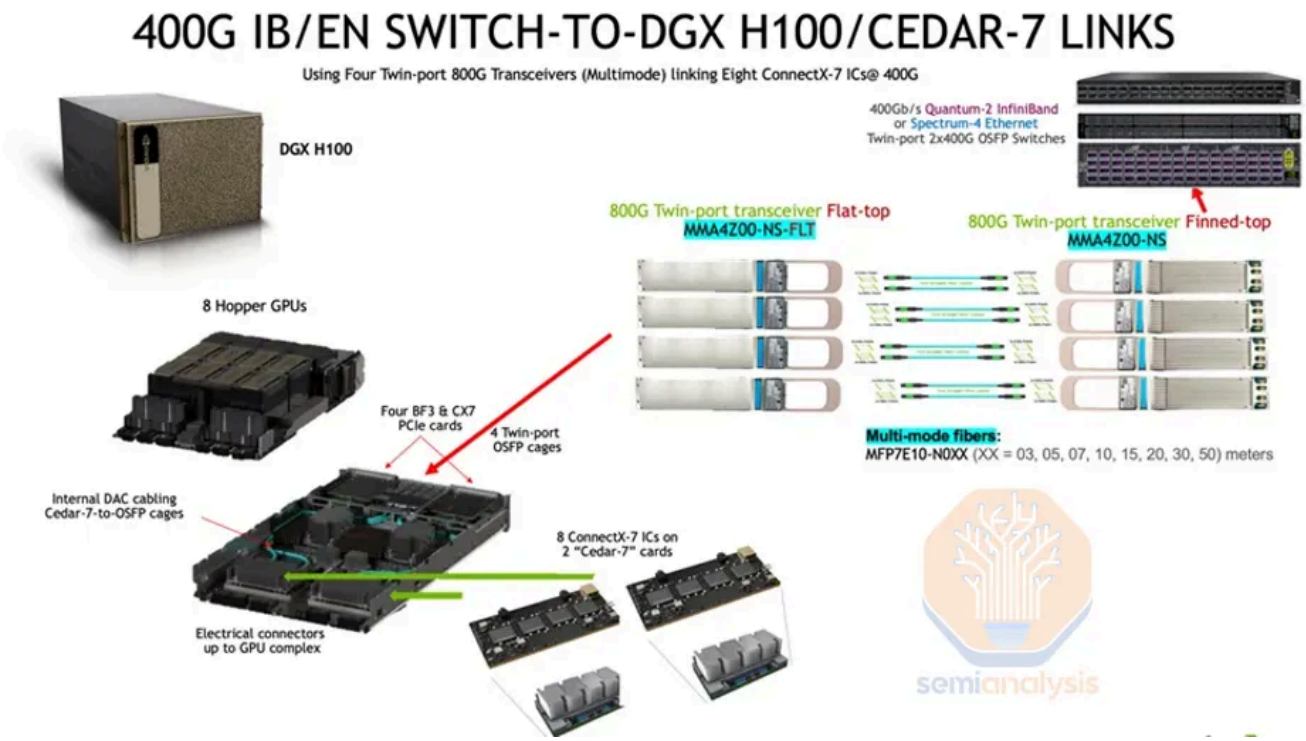
Since each GPU is directly attached to a ConnectX-7 NIC (through a PCIe switch), there is no fault tolerance at the network architecture level thus the fault must be handled at the user training code, directly adding to complexity of the codebase. This is one of the main challenges of current GPU network fabrics from NVIDIA and AMD, where even if one NIC fails, that GPU then has no alternative paths to talk to other GPUs. Due to the way current LLMs use tensor parallelism within the node, if even one NIC, one transceiver or one GPU fails, the whole server is considered down.

There is a lot of work being done to make networks reconfigurable and not have nodes be so fragile. This work is critical as the status quo means an entire GB200 NVL72 would go down with just 1 GPU failure or 1 optical failure. A multi-million-dollar 72 GPU rack going down is far more catastrophic than an 8 GPU server worth a few hundred thousand dollars.

Nvidia has noticed this major problem and added a dedicated engine for reliability, availability and serviceability (RAS). We believe RAS engine analyzes chip level data such temperature, number of recovered ECC retries, clocks speed, voltages to predict when the chip will likely failure and alert the datacenter technician. This will allow them to do proactive maintenance such as using a higher fan speed profile to maintain reliability, take the server out of service for furthermore physical inspection at a later maintenance window. Furthermore, before starting a training job, each chip's RAS engine will perform a comprehensive self-check such as running matrix multiplication with known results to detect silence data corruptions (SDC).

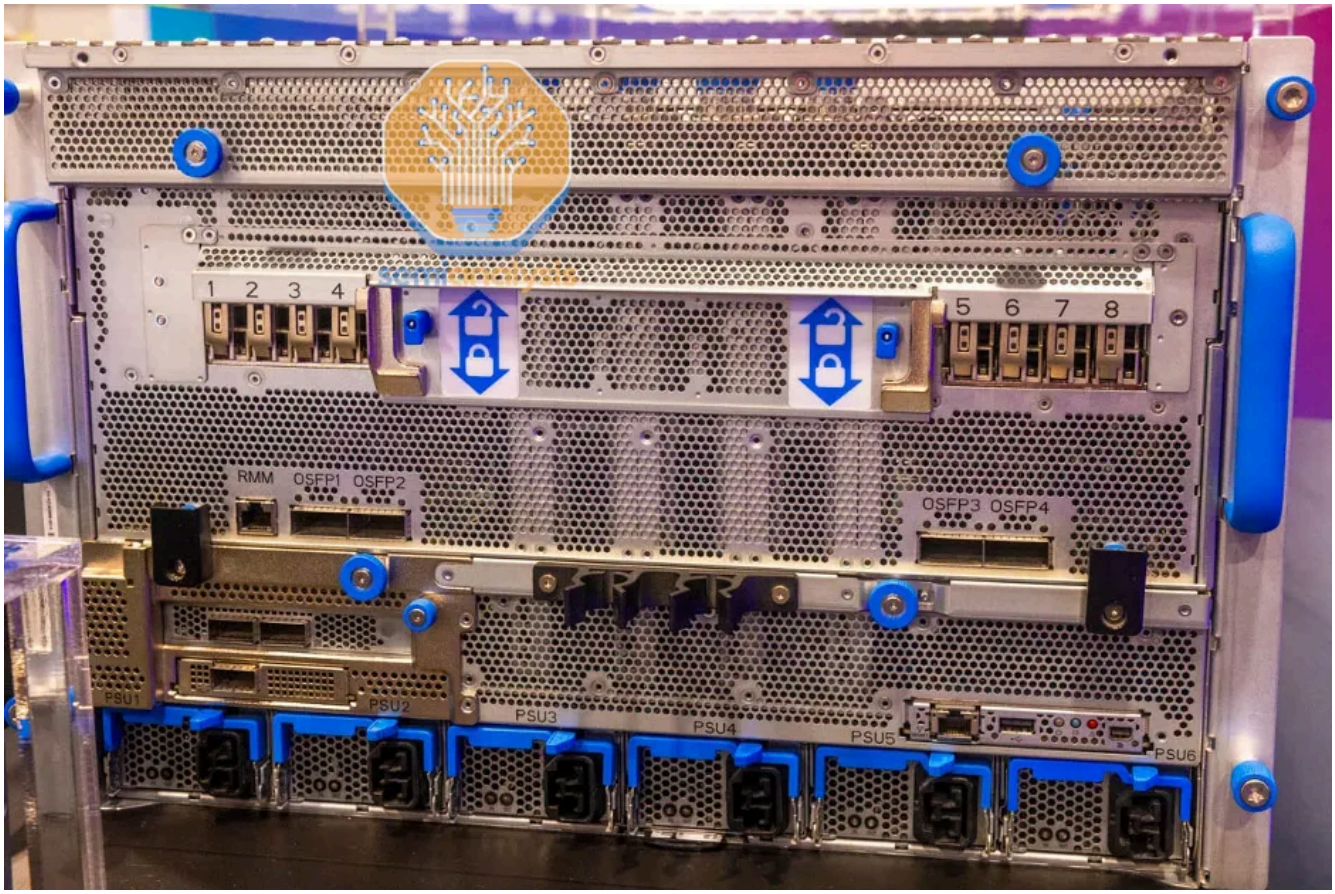
Cedar-7

Another cost optimization that some customers like Microsoft/Openai are doing is by using the Cedar Fever-7 networking module per server instead of using 8 PCIe form factor ConnectX-7 networking cards. One of the main benefits of using a Cedar Fever module is that it allows for just using 4 OSFP cages instead of 8 OSFP cages thus allowing for the use of twin port 2x400G transceivers on the compute node end too, instead of just the switch end. This reduces the transceiver count to connect to the leaf switches from 8 transceivers to 4 transceivers per H100 node. The total compute node end transceiver counts to connect GPUs to leaf switch reduces from 98,304 to 49,152.



Source: Nvidia

Since the GPU to leaf switch links are cut in half, this also helps with the estimated time to first job failure. We estimate that each twin port 2x400G link has a mean time to failure per link of 4 years (vs. 5 years of single port 400G link) and this will bring the estimated time to first job failure to 42.05 minutes, this is way better than the 26.28 minutes without Cedar-7 modules.



Source: ServeTheHome

Spectrum-X NVIDIA

There is currently a 100k H100 cluster being deployed and will be operational by end of the year that is using NVIDIA Spectrum-X Ethernet.

Nvidia's InfiniBand Problem - Spectrum-X AI Fabric, Tomahawk-5, Jericho-3AI, Quantum-2

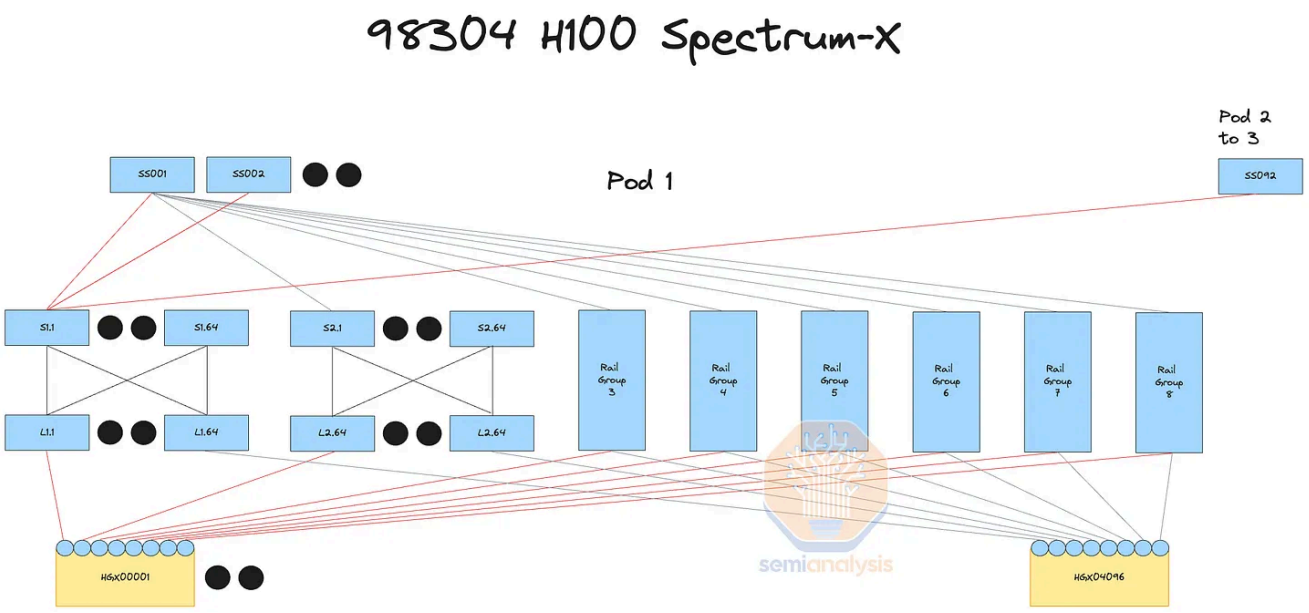
DYLAN PATEL • MAY 28, 2023

Read full story →

Last year we covered a variety of advantages Spectrum-X has over InfiniBand in large networks. Even outside the performance and reliability advantages, Spectrum-X also has a huge cost advantage. Spectrum-X Ethernet is that each SN5600 switch has 128 ports of 400G while the InfiniBand NDR Quantum-2 switch only has 64 ports of 400G. Note that Broadcom's Tomahawk 5 switch ASIC also supports 128 400G ports, putting the current generation of InfiniBand at a big disadvantage.

A fully interconnected 100k cluster can be 3 tiers instead of 4 tiers. Having 4 tiers instead of 3 tiers means that there are 1.33x more transceivers required. Due to the lower radix of the Quantum-2 switch, the maximum number of fully interconnected GPUs on a 100k cluster is limited to 65,536 H100s. The next generation of InfiniBand switches called Quantum-X800 solves this by having 144 800G ports, though as one can tell from the number "144", this is designed for use with the NVL72 and NVL36 systems and is not expected to be used much in B200 or B100 clusters. Even though there is cost savings of not having to do 4 tiers with Spectrum-X, the unfortunate downside is that you still need to buy highly marked up transceivers from Nvidia LinkX product line as other transceivers may not work or be validated by Nvidia.

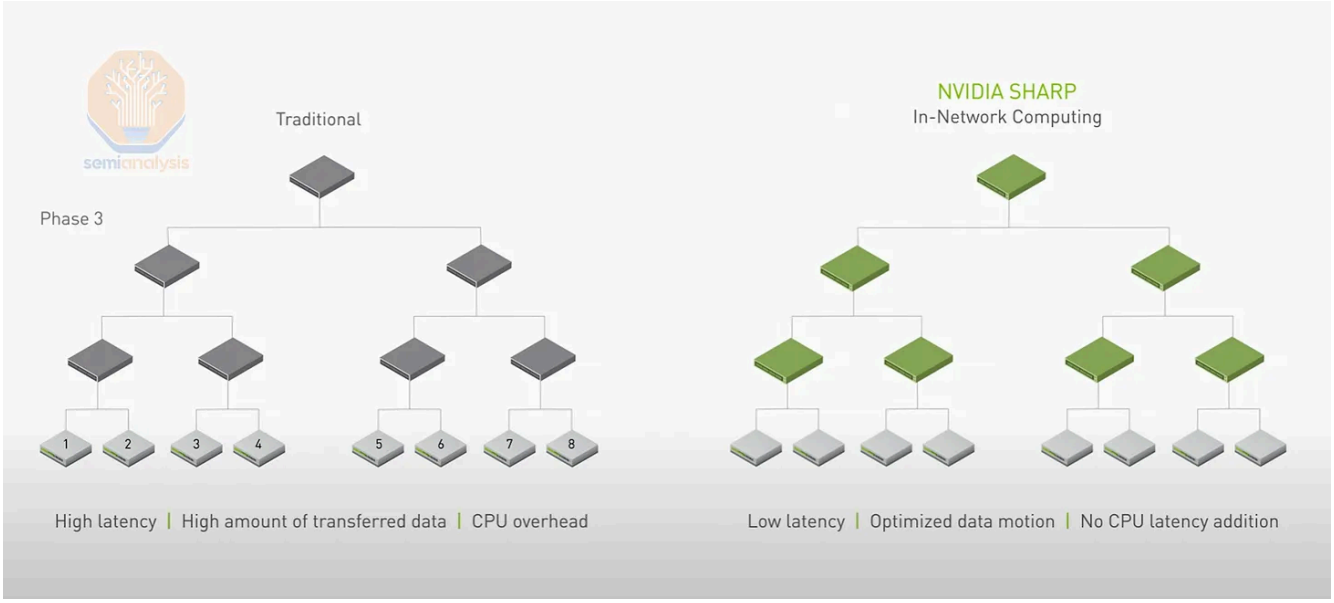
The main advantage of Spectrum-X over other vendors is that Spectrum-X is supported first class by NVIDIA libraries such as NCCL and Jensen pushes you up the allocation queue to being one of the first customers for their new product line vs. with the Tomahawk 5 chips, you need a lot of in house engineering effort to optimize your network with NCCL in order to achieve max throughput.



Source: SemiAnalysis

An unfortunate downside of using Ethernet instead of InfiniBand for your GPU fabric is that Ethernet does not currently support SHARP in network reductions. In-network reductions work by performing the summation of each GPU by having the network

switch run those computations. The theoretical network bandwidth increase of SHARP is 2x as it reduces the number of sends and writes each GPU has to do by 2x.



Source: Nvidia

Another downside of Spectrum-X is that for the first generation of 400G Spectrum-X, Nvidia uses Bluefield3 instead of ConnectX-7 as a bandaid solution. For the future generations, we expect ConnectX-8 to work perfectly fine with 800G Spectrum-X though. The price delta between a Bluefield-3 and a ConnectX-7 card is around $300 ASP for hyperscaler volumes, with the other downside being that card uses 50 watts more compared to ConnectX-7. Thus, for each node, 400W of additional power is needed, lowering the “intelligence per picojoule” of the overall training server. The datacenter that you put Spectrum X in now requires an additional 5MW for a 100,000 GPU deployment versus a Broadcom Tomahawk 5 deployment with the exact same network architecture.

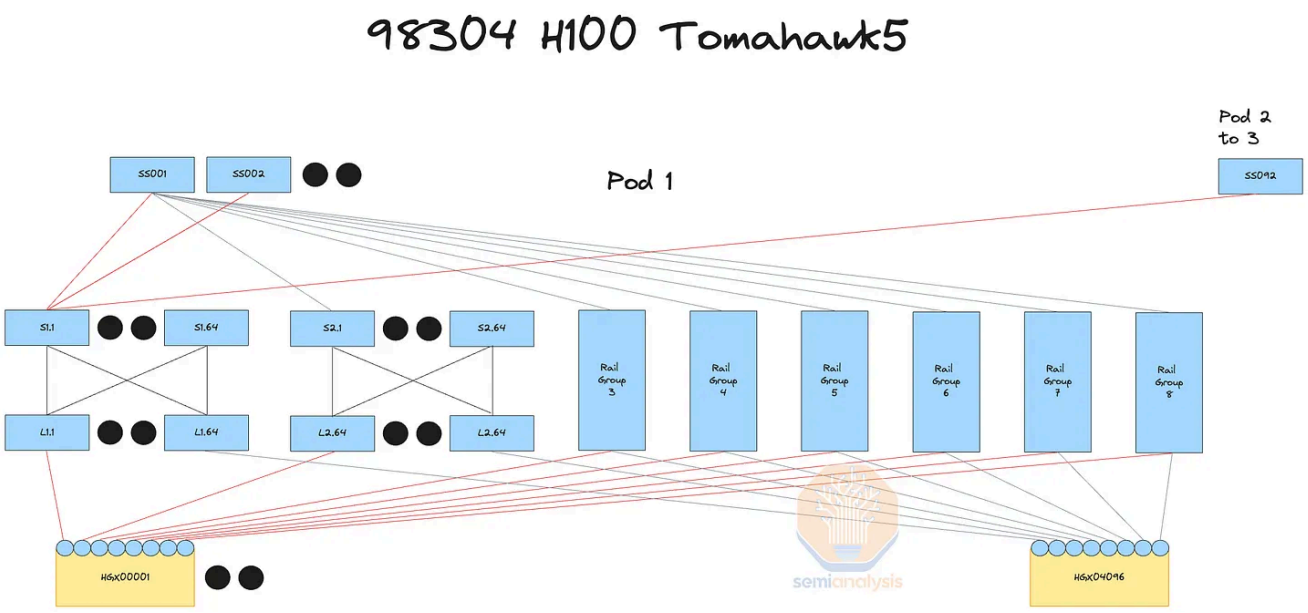
Broadcom Tomahawk 5

In order to avoid paying the massive Nvidia tax, a lot of customers are deploying with Broadcom Tomahawk 5 based switches. Each Tomahawk 5 based switch has the same number of ports as the Spectrum-X SN5600 switch at 128 400G ports and achieves similar performance if your firm has good network engineers. Furthermore, you can buy

any generic transceivers and copper cables from any vendor in the world and mix and match.

Most customers are partnering directly with ODMs such as Celestica for switches using Broadcom based switch ASICs and firms like Innolight and Eoptolink for transceivers. Based on the switch cost and the generic transceiver cost, Tomahawk 5 is way cheaper compared to Nvidia InfiniBand and is also cheaper when compared to Nvidia Spectrum-X.

The unfortunate downside of this is that you need to have enough engineering capacity to patch and optimize NCCL communication collectives for the Tomahawk 5. Out of the box, NCCL communication collectives is only optimized for Nvidia Spectrum-X and Nvidia InfiniBand. The good news is that if you have 4 billion dollars for a 100k cluster, you have enough engineering capacity to patch NCCL and write the optimizations. Of course software is hard, and Nvidia is always on the bleeding edge, but generally we expect every hyperscaler to make these optimizations and switch away from InfiniBand.



Source: SemiAnalysis

We will now talk through the bill of materials for 4 different 100k GPU cluster network designs, the switch and transceiver costs associated with them showing the advantages of different network designs, and a physical floorplan of a GPU cluster optimized for reduced optics.

← Previous