AI Workload
Networking challenges

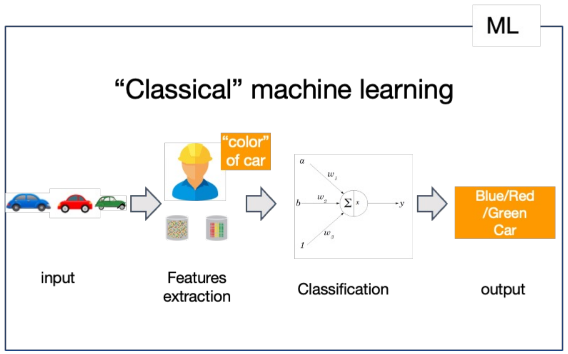
Peter Lundqvist

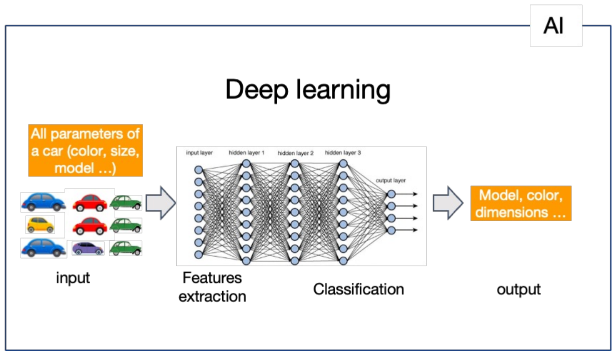
Arista Networks

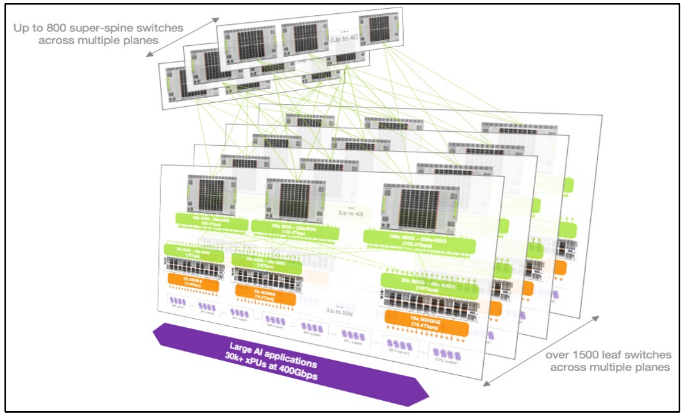
peter@arista.com

“AI”, i do sense audience fatigue on the topic…

- *“Another vendor that rebrand its if-then-else code and call it Deep Learning when it’s same old shit”*
- *“ More Datacenter Fabric crap, i work with THE Internet”*
- *“ Deepseek… that shit that killed my pension plan “*
- *“In the end it will be self-aware and we are all doomed…”*

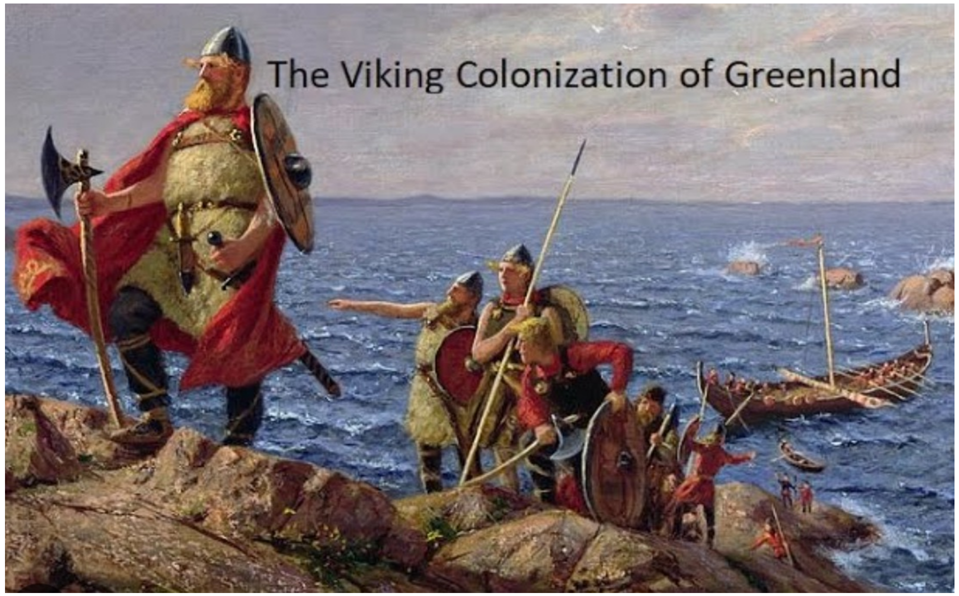






However…

- The train have left the station long time ago… and you all are onboard like it or not !
- You all have learn how to battle the Internet TCP based networks and its mechanisms CWND, SACK, DupACK, Retransmission etc…
- AI Workloads your next battleground
- ChatGPT the new Googling
- Discover new things… be a Viking !



AI workloads main characteristics

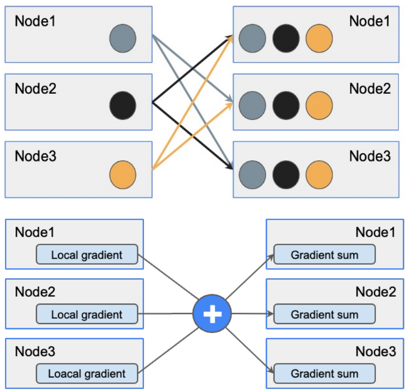
Requires specialized Hardware

- Many names… xPU (GPU, ICU, TPU…)

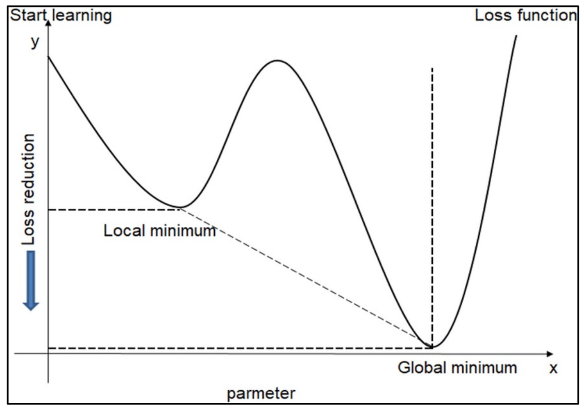
Collective communication/libraries

- The single vs multi-node communication and its data transfer models *CCL (ex NCCL)
- **Process buzzwords**
- LLM, Forwarding vs Backward pass, Weight, Calculate loss, Barrier, Gradient Descent…

AllGather & AllReduce

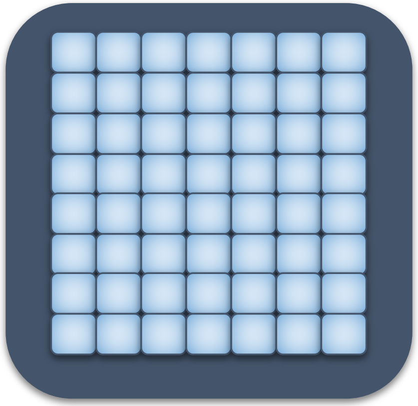


Gradient Descent

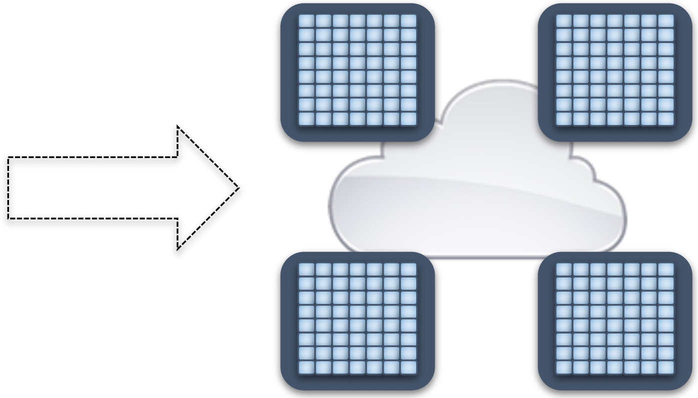


That thing with CPU vs GPU

CPU
1-10+ Cores
Optimized for serial task



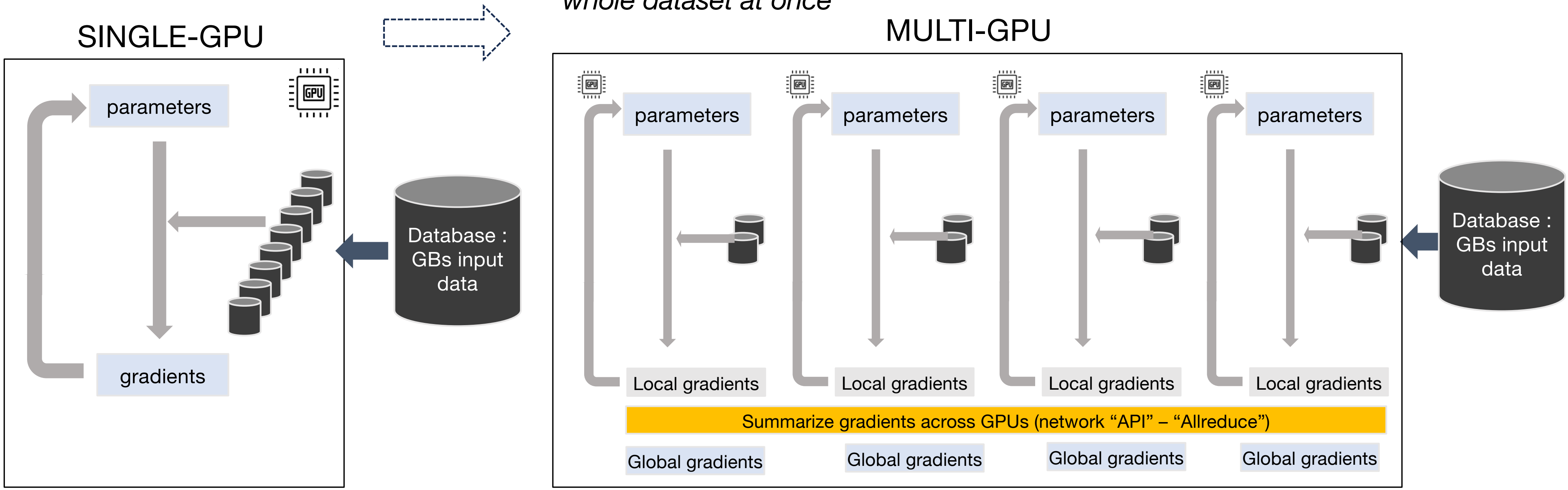
GPU
100-10k+ Cores
Optimized for parallel task



GPU Clusters
x00k GPUs
Multi-GPU Network

Single vs Multi-GPU… makes all the difference

"Computing the gradient for individual data points and then averaging them, is the same as computing the gradient using the whole dataset at once"



- **Data parallelism** allows feeding different GPUs with different parts of the "data" and process the data in parallel
- After each GPU processed its data, it shares the result with all the other GPUs

AI Collective Communication

Responsible for the networking in AI fabrics?

These “patterns” needs to be understood since this steers the behavior(s) of these workloads

- Broadcast
- Allgather
- Allreduce
- Reduce
- Reduce-scatter
- Barrier
- …

Type	Function	Description
Data Movement	Bcast	One to group. One process sends (broadcasts) some data to all the processes in a group.
	Gather	Group to one. If an array is scattered across all processes in the group. And one process (root) collects each piece of the array into a specified array.
	Allgather	All processes, not just the root, receive the result of Gather.
	Scatter	One-To-Group. One process distributes the data into n segments, where the i-th segment is sent to the i-th process in the group which has n processes.
	Alltoall	This is an extension to Allgather.Each process sends distinct data to each receiver. The j-th block from process i is received by process j and stored in the i-th block.
Data Aggregation	Reduce	Group to one. Used to collect data or partial results from multiple processing units and to combine them into a global result by a chosen operator.
	All-Reduce	Sistribute the result of a Reduce operation to all processes in the group.
	Reduce-Scatter	scattering the result of reduction to all processes
	Scan	A Scan operation performs partial reductions on distributed data.
Synchronization	Barrier	A synchronous operation to synchronize all processes within a communicator.

https://www.ietf.org/archive/id/draft-yao-tsvwg-cco-problem-statement-and-usecases-00.html

High level example AI training loop

Step 1: Data propagation

- Distribute the data among GPU

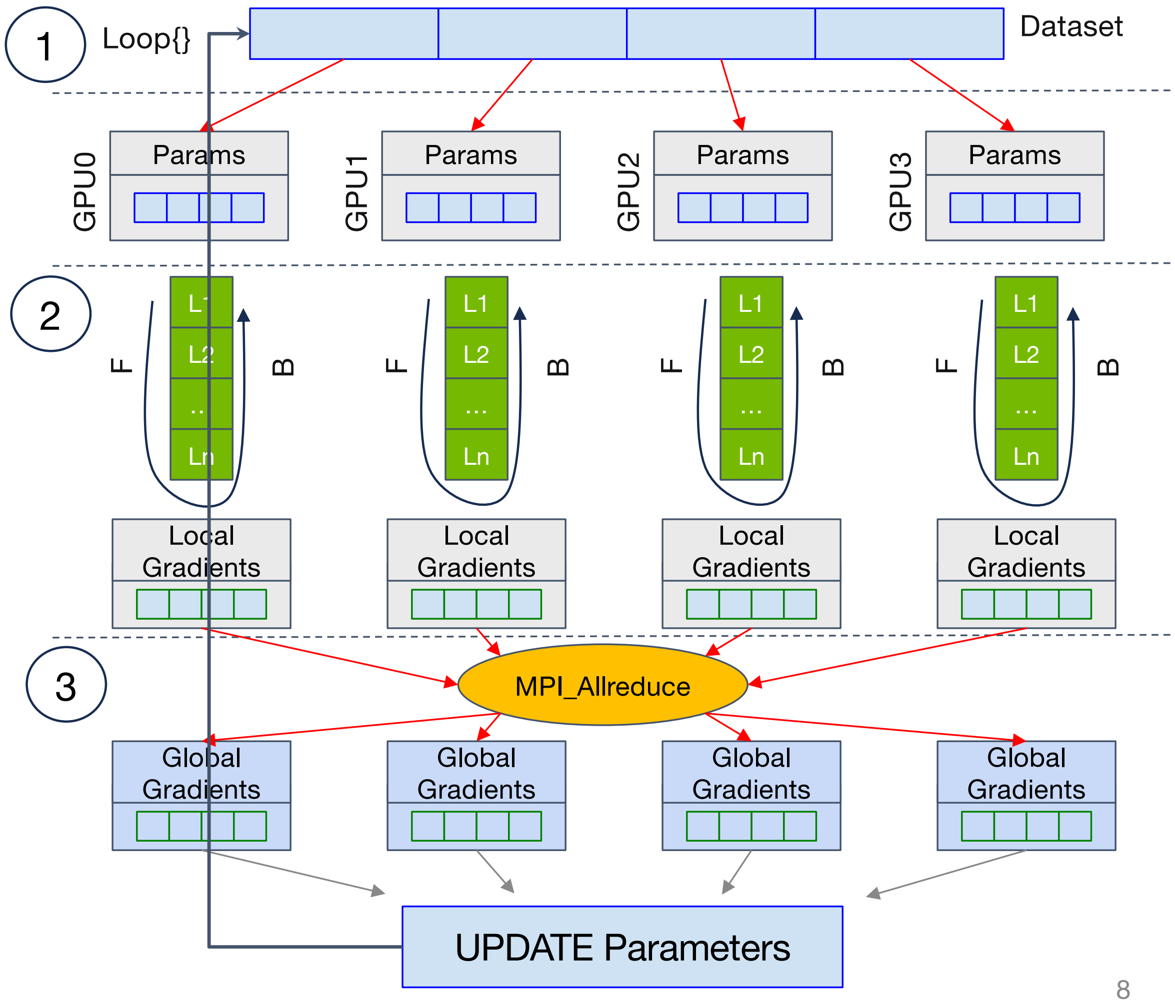
Step 2: Forward and Backward pass

- Perform forward pass and calculate the prediction
- Calculate loss by comparing prediction with actual output
- Perform backward pass: compute the local gradients of the loss function

Step 3: Gradient aggregation

- Call Allreduce to reduce the local gradients
- Update parameters using global gradients

The key metric: Job Completion Time (JCT)

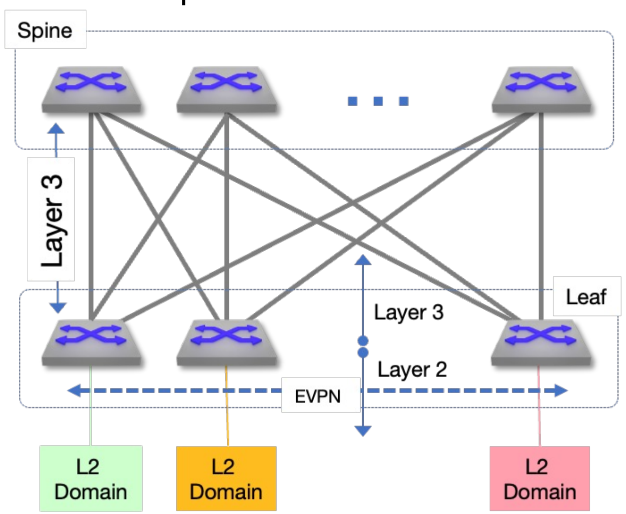


Data Center Fabric Design Principles

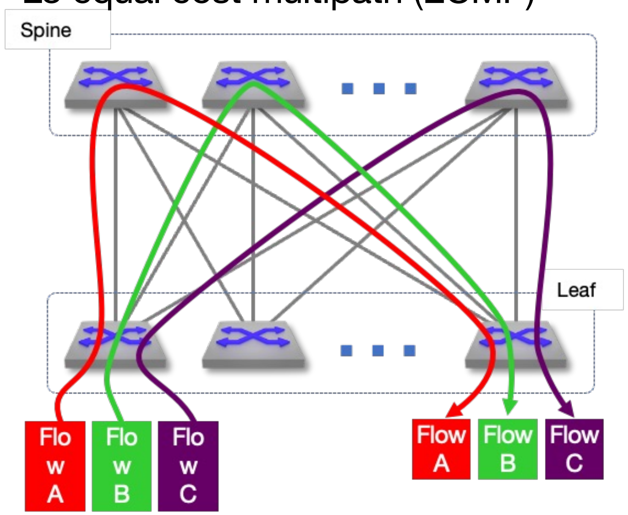
Relax, this session NOT a “DC fabrics for dummies” session

- **However…** AI workloads currently resides in DC fabrics and that have build the experience
- Come on… It’s L3, ECMP and BGP… what’s not to like ☺

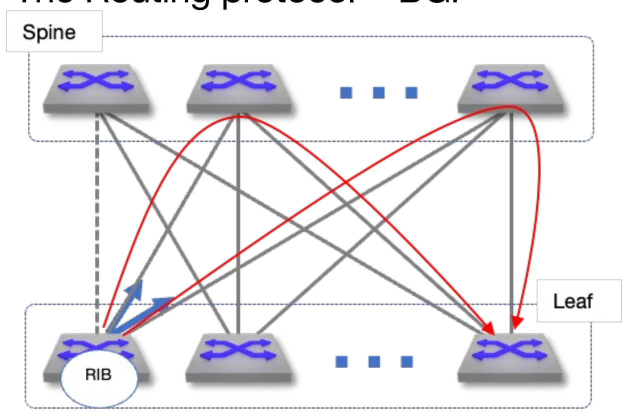
L3 Leaf-Spine Fabrics



L3 equal cost multipath (ECMP)

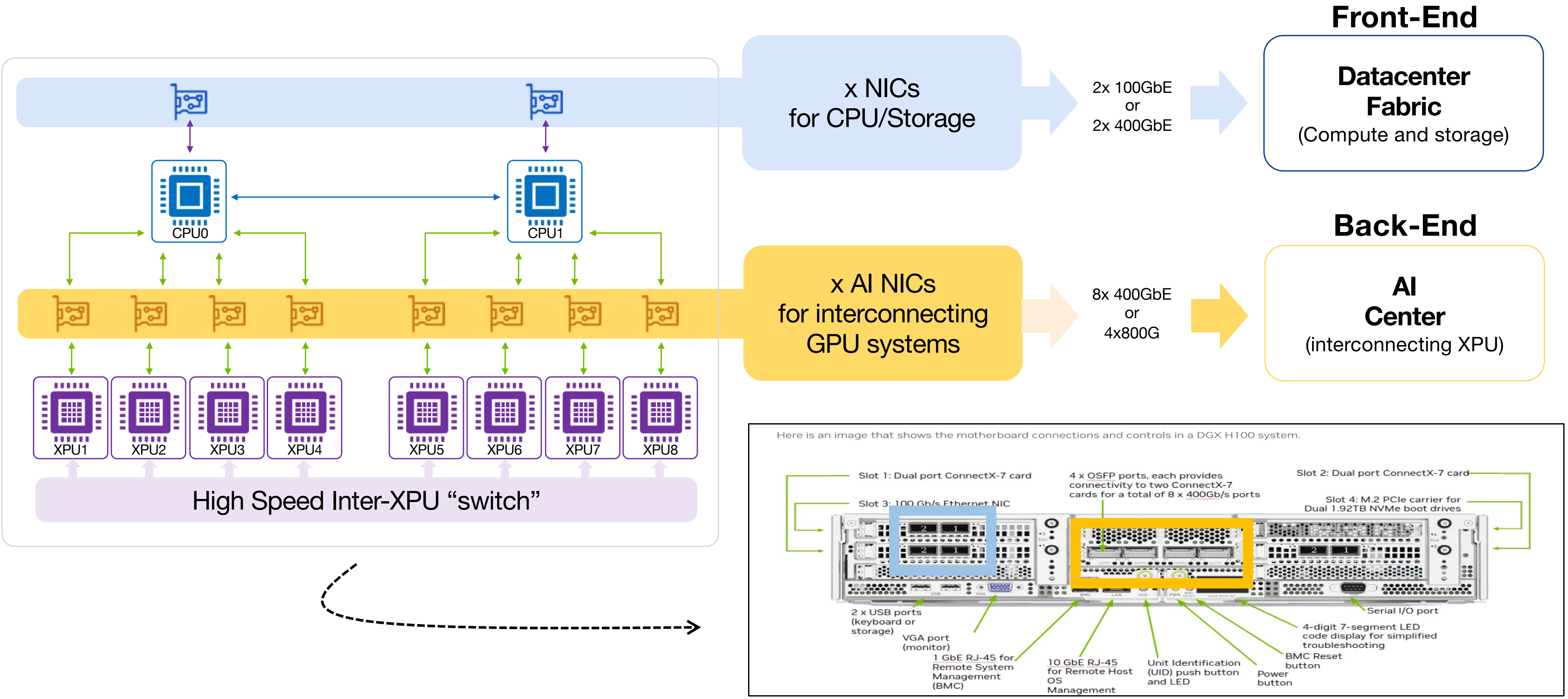


The Routing protocol = BGP



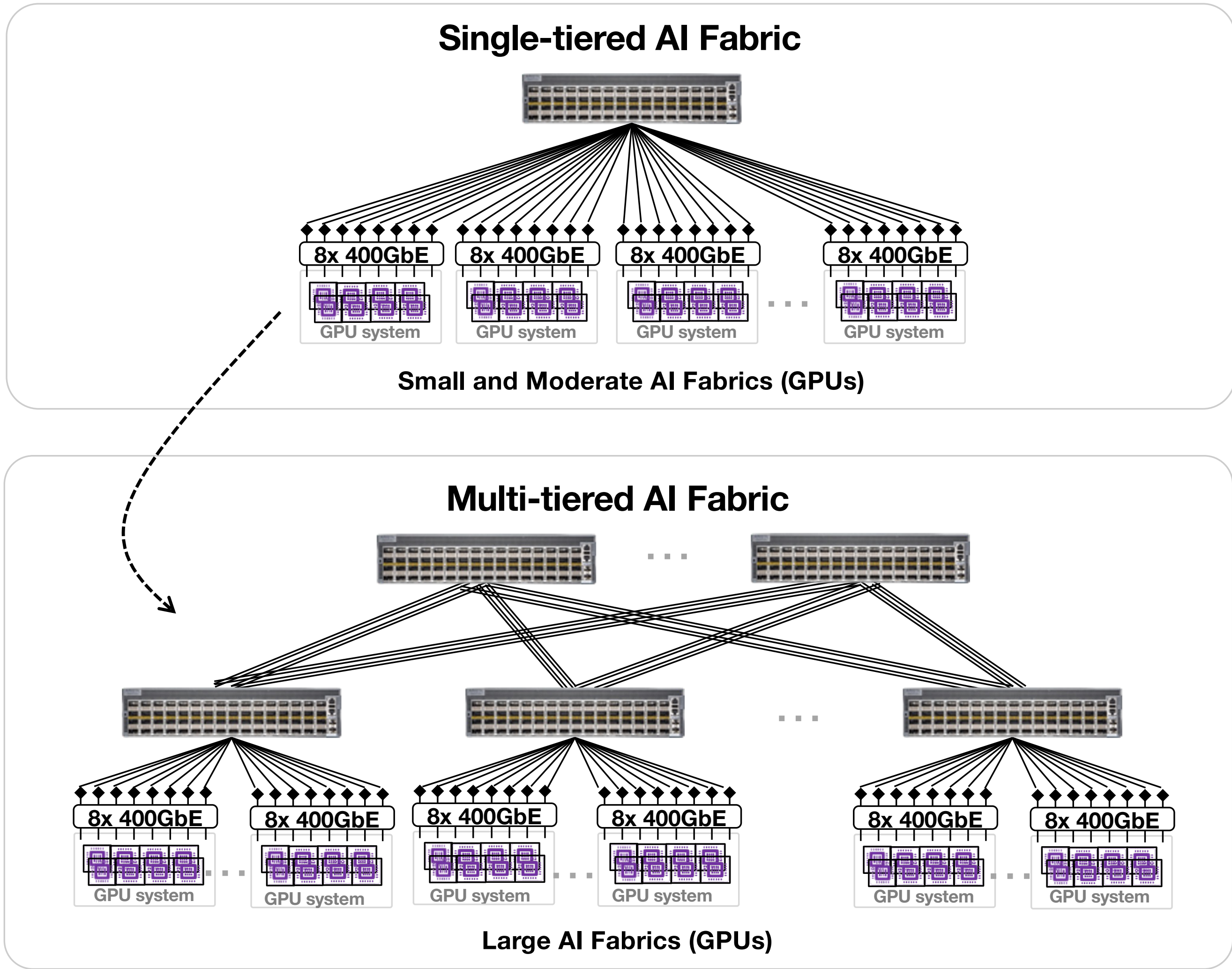
Node maintain its own RIB&adjency state
PATH change deterministic to each RIB

How to connect these GPU systems ?

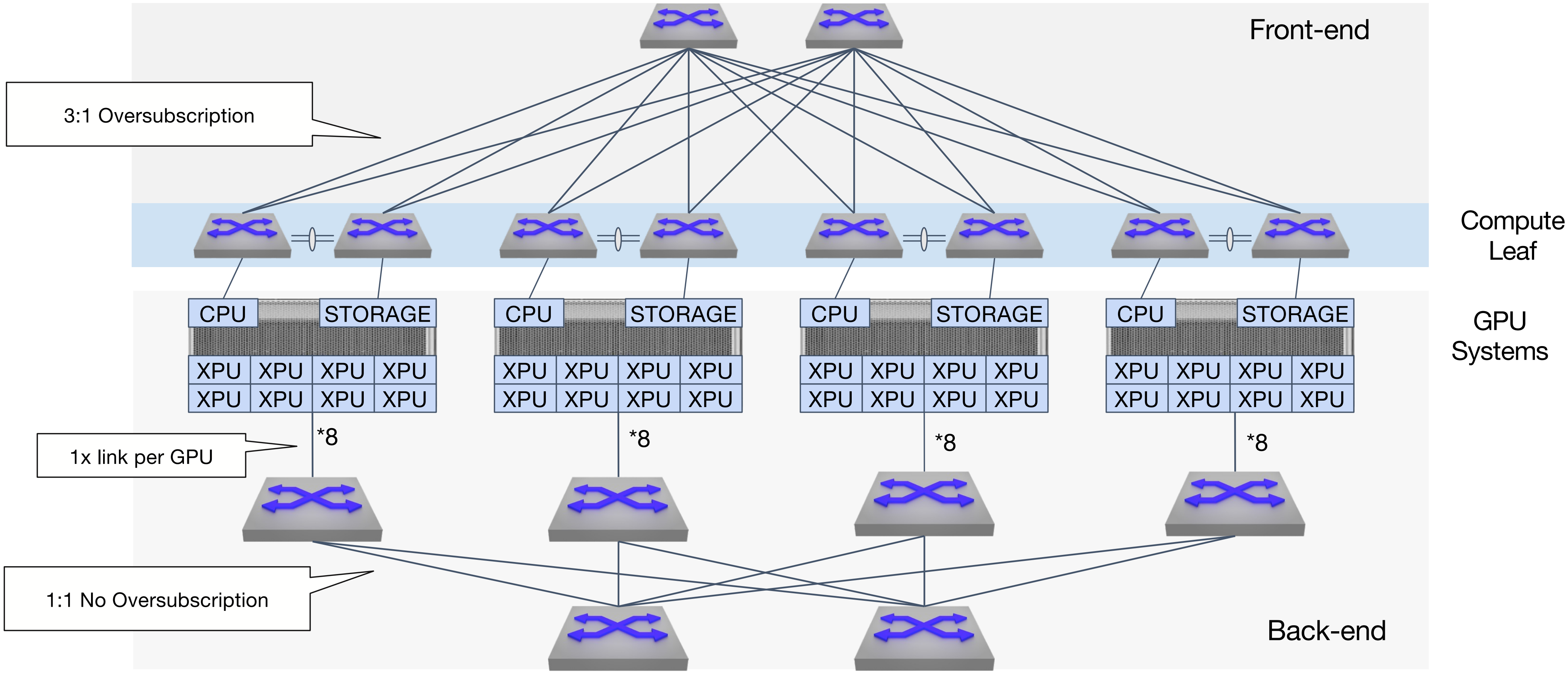


AI Fabric scale Challanges

- Single-tiered or “Spline” less networking challenges, careless multi-GPU
- However when grow from single to 2 or 3-tier…
- Then it’s not that easy with AI Workflows

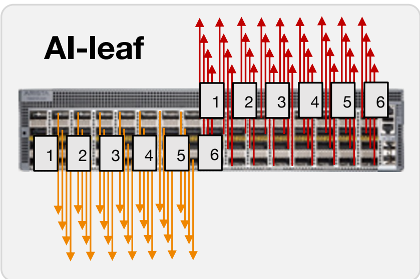


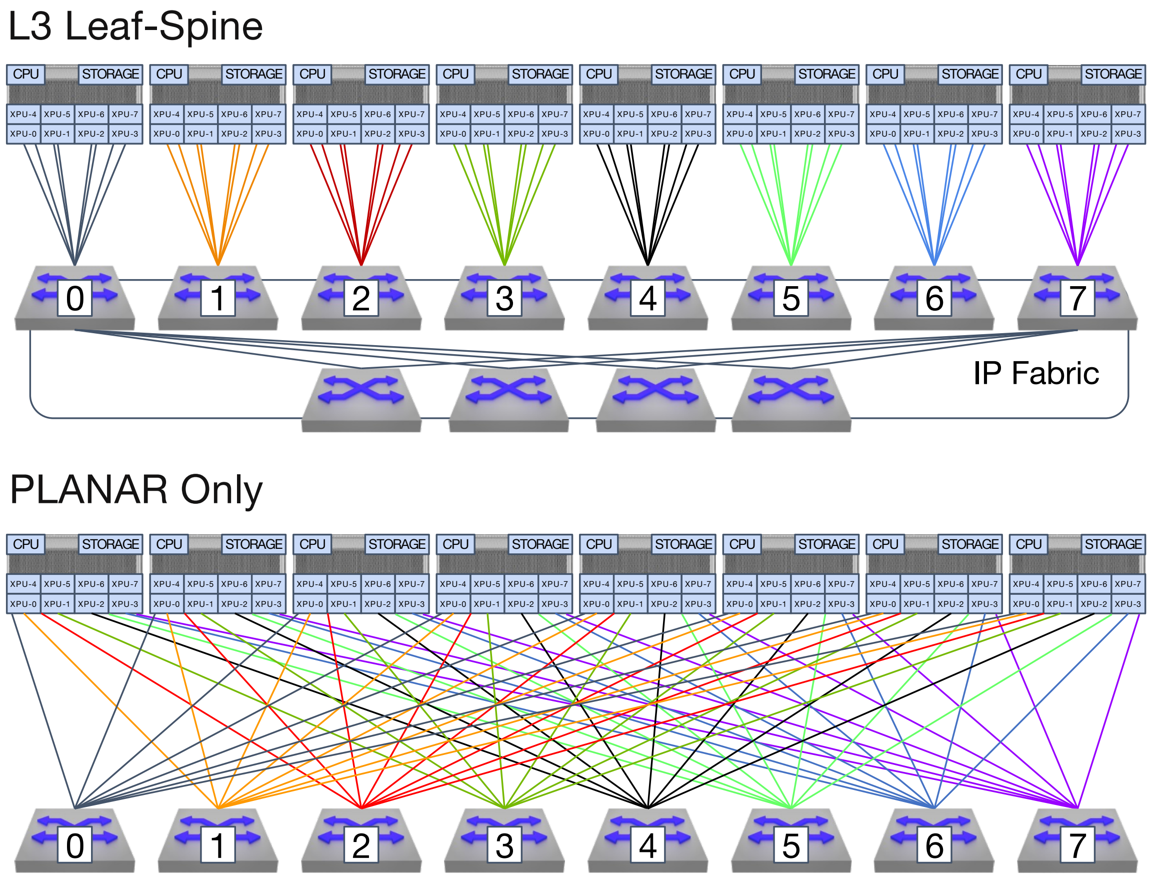
L3 Leaf-Spine Front-end & Back-end



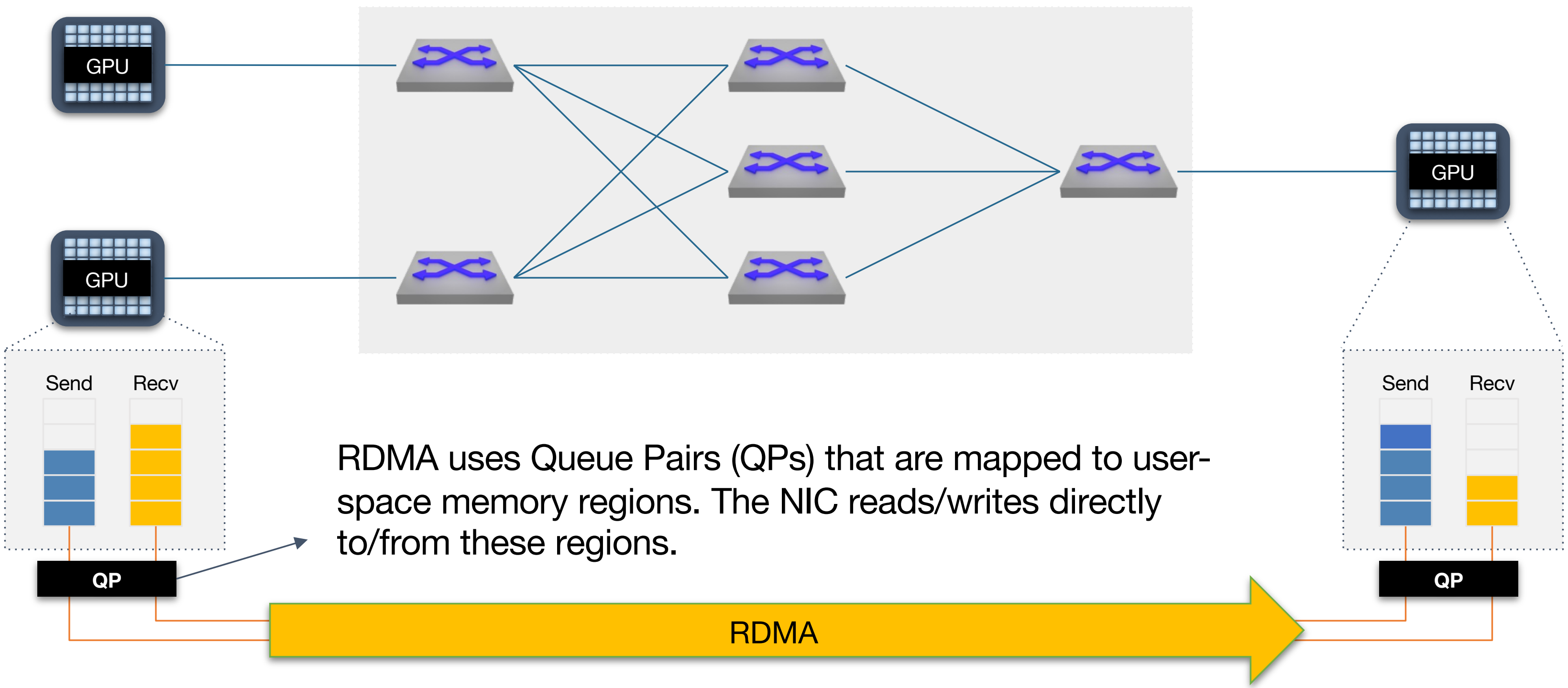
Alternative topologies

- A Planar (Rail) is comprised of GPUs that have the same “rank” and connected to the same network
- Collective communications (ex NCCL library) places flows on each Planar (here 1-7) based on example utilization
- Less networking devices, drawback lots of fibers and complex patching



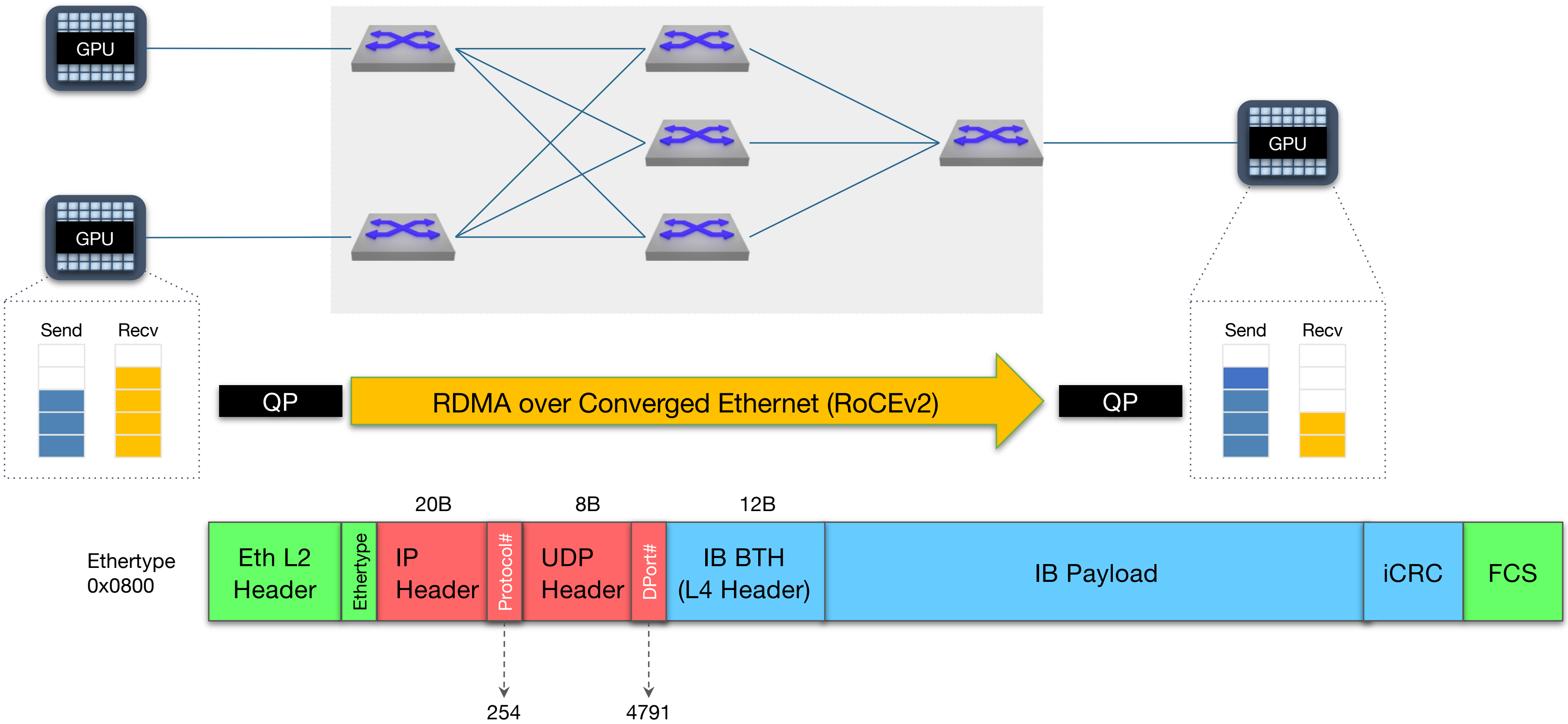


GPU-to-GPU Traffic over Ethernet Fabrics



RDMA = Remote Direct Memory Access

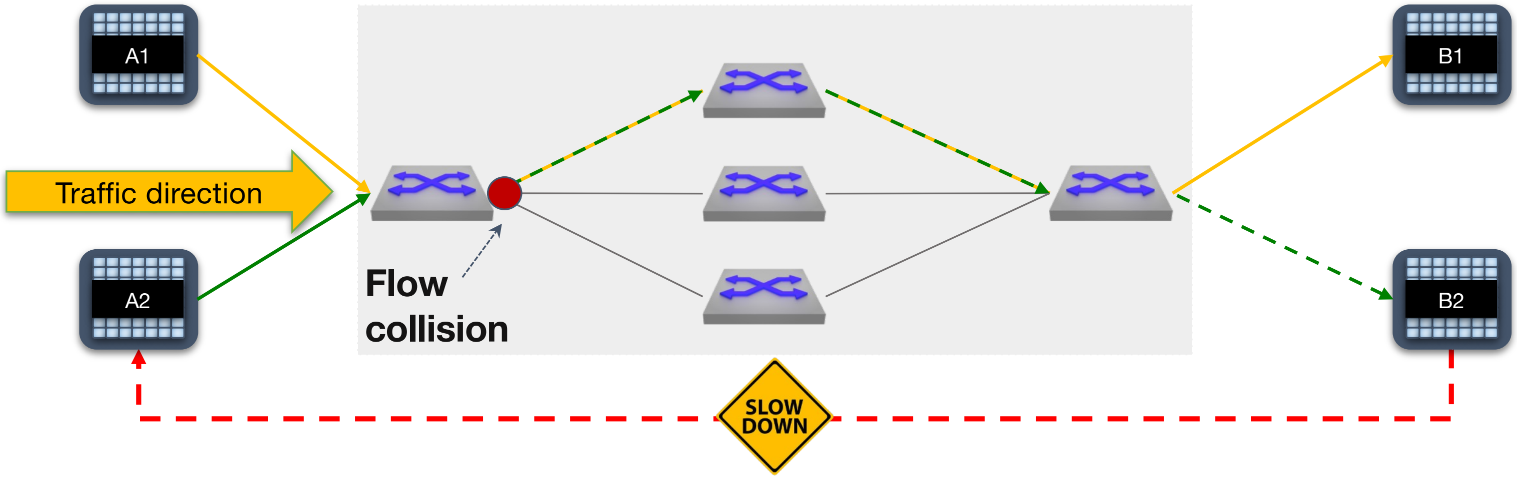
RDMA over Converged Ethernet (RoCEv2)



Now… Challenges with AI Flows

High probability of **flow collisions**

- Low entropy in GPU-to-GPU traffic pattern makes ECMP difficult
- Size and duration of the flows
- The amount of traffic compare to the buffering capability over each ECMP member



No QOS scheduling or queue priority will help since what to drop if *.* important ?

1. Traffic needs to slow down without been dropped => **Pause frames&ECN bits**
2. Traffic needs to be load balance beyond 5-Tuple hashing => **RDMA header**

NO… its NOT a QOS game

- Its not prioritize something at the cost of drop something else…
- **Again what to drop if everything is Important ?**
- Goal: Prevent loss&jitter by avoid burst and incast

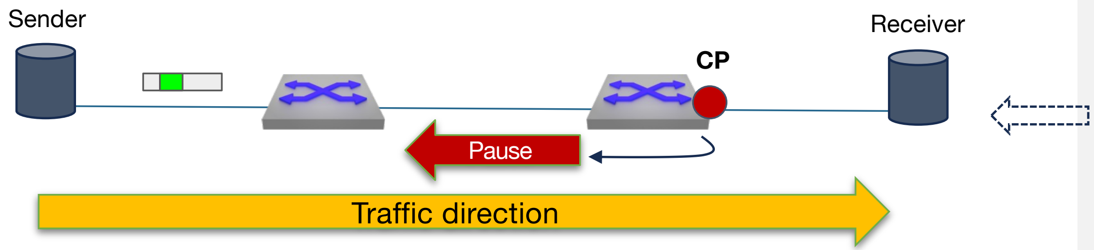
```
Report generated at 2025-02-24 20:33:00
Time                              Interface(TC)  Tx-Latency (usecs)
-------------------------------------------------------------------
0:00:00.29770 ago                 Et1(1)         1367.560
0:00:00.29962 ago                 Et1(1)         545.592
0:00:00.30472 ago                 Et1(1)         1015.288
0:00:00.30977 ago                 Et1(1)         879.248
0:00:00.36439 ago                 Et1(1)         1529.376
0:00:00.36669 ago                 Et1(1)         579.960
0:00:00.37180 ago                 Et1(1)         1049.656
0:00:00.37690 ago                 Et1(1)         1489.280
0:00:00.38194 ago                 Et1(1)         1496.440
(…)
```

```
arista(config)#sh queue-monitor length
Report generated at 2025-02-24 19:23:30
S-Start, U-Update, E-End, TC-Traffic Class
Segment size for S, U and E congestion records is 208 bytes
* Max queue length during period of congestion + Period of congestion exceeded counter
-----------------------------------------------------------------------------------------------------------------------------------
Type    Time                   Absolute                           Interface      Congestion     Queue      Time of Max     Fabric
                               Time                               (TC)           duration       length     Queue length    Peer
                                                                                 (usecs)        (segments) relative to
                                                                                                           congestion
                                                                                                           start
                                                                                                           (usecs)
-----------------------------------------------------------------------------------------------------------------------------------
 E   0:00:00.69912 ago         2025-02-24 19:42:34.66111          Et1(1)         74382          1076*      9797
 U   0:00:00.71738 ago         2025-02-24 19:42:34.64285          Et1(1)         N/A            1066       N/A
 U   0:00:00.72303 ago         2025-02-24 19:42:34.63720          Et1(1)         N/A            1069       N/A
 U   0:00:00.72867 ago         2025-02-24 19:42:34.63156          Et1(1)         N/A            941        N/A
 U   0:00:00.73428 ago         2025-02-24 19:42:34.62595          Et1(1)         N/A            1055       N/A
 U   0:00:00.73989 ago         2025-02-24 19:42:34.62034          Et1(1)         N/A            1071       N/A
 U   0:00:00.74551 ago         2025-02-24 19:42:34.61472          Et1(1)         N/A            1071       N/A
 U   0:00:00.75114 ago         2025-02-24 19:42:34.60909          Et1(1)         N/A            1057       N/A
 U   0:00:00.75665 ago         2025-02-24 19:42:34.60358          Et1(1)         N/A            1065       N/A
 U   0:00:00.76228 ago         2025-02-24 19:42:34.59795          Et1(1)         N/A            1055       N/A
 U   0:00:00.76793 ago         2025-02-24 19:42:34.59230          Et1(1)         N/A            1062       N/A
 S   0:00:00.77350 ago         2025-02-24 19:42:34.58673          Et1(1)         N/A            659        N/A
(…)
```

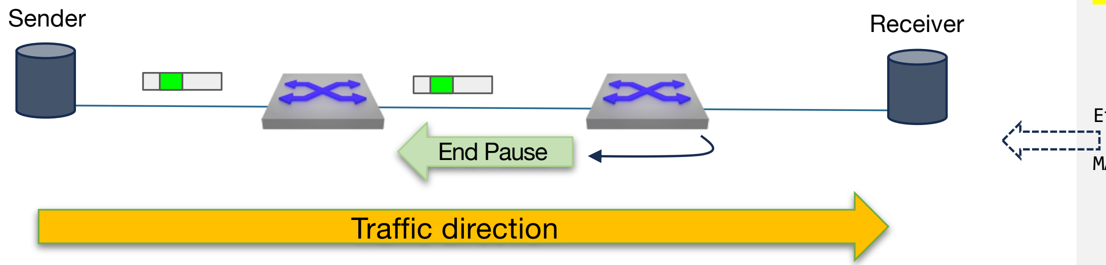
Priority Flow Control (PFC)

```
interface Ethernet15
   dcbx mode ieee
   flowcontrol send on
   flowcontrol receive on
   qos trust cos|dscp
   priority-flow-control on
   priority-flow-control priority 1 no-drop
(…)
```

- When threshold exceeded, a **pause frame** send to halt data transmission for a **specified period of time**
- When the congestion mitigated and rate below the threshold, a pause frame send with time 0 in order to restart data transmission for that specific link

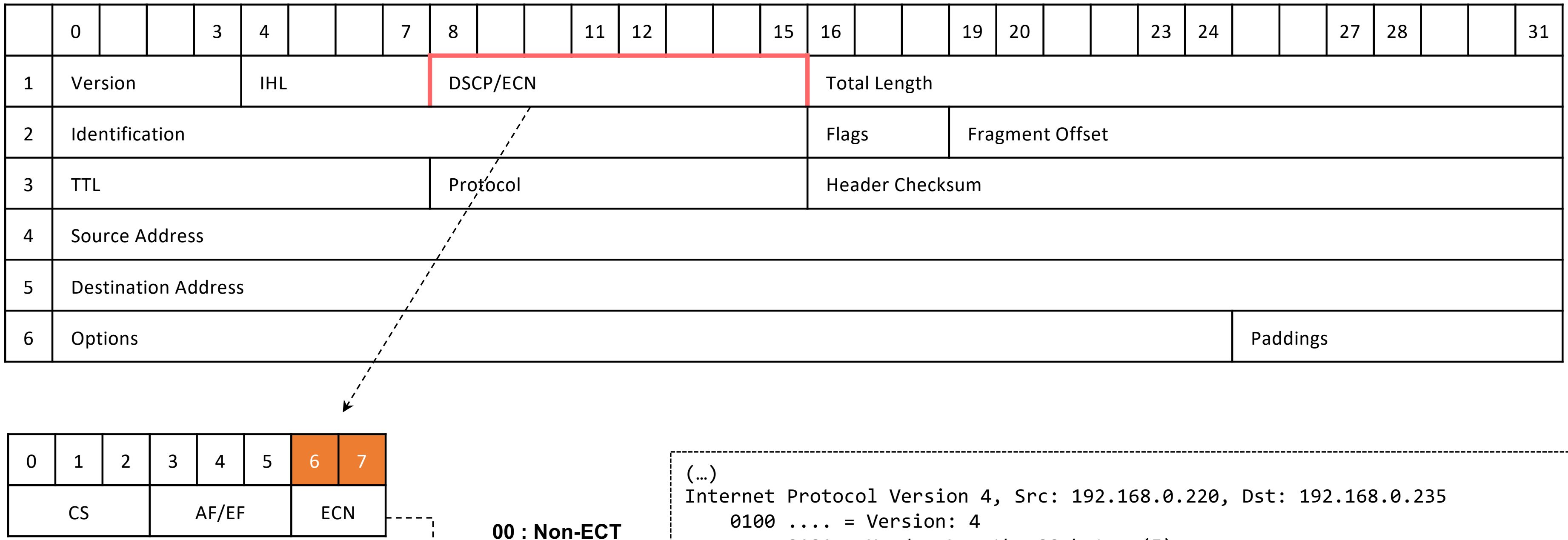



```
Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
    Encapsulation type: Ethernet (1)
    Arrival Time: Feb 20, 2025 22:37:23.527297000 CET
    UTC Arrival Time: Feb 20, 2025 21:37:23.527297000 UTC
    Epoch Arrival Time: 1740087443.527297000
Ethernet II, Src: 00:1c:73:f7:2f:2a, Dst: 01:80:c2:00:00:01
    Type: MAC Control (0x8808)
MAC Control
    Opcode: Class Based Flow Control [CBFC] Pause (0x0101)
    CBFC Class Enable Vector: 0x0003, C0, C1
    CBFC Class Pause Times
        C0: 65535
        C1: 65535
(…)
```




```
Frame 2: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
    Encapsulation type: Ethernet (1)
    Arrival Time: Feb 20, 2025 22:37:23.527520000 CET
    UTC Arrival Time: Feb 20, 2025 21:37:23.527520000 UTC
    Epoch Arrival Time: 1740087443.527520000
Ethernet II, Src: 00:1c:73:f7:2f:2a, Dst: 01:80:c2:00:00:01
    Type: MAC Control (0x8808)
MAC Control
    Opcode: Class Based Flow Control [CBFC] Pause (0x0101)
    CBFC Class Enable Vector: 0x0003, C0, C1
    CBFC Class Pause Times
        C0: 0
        C1: 0
(…)
```

The “lost” bits of the DSCP header = ECN



00 : Non-ECT
01 : ECT(1)
10 : ECT(0)
11 : CE

CS = Class Selector [CS0 to CS7]
AF = Assured Forwarding
EF = Expedited Forwarding
ECN = Explicit Congestion Notification

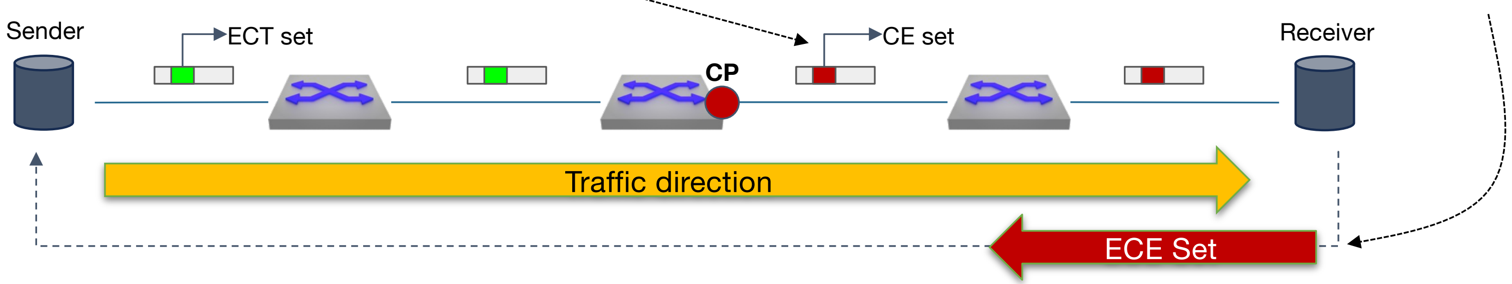
```
(…)
Internet Protocol Version 4, Src: 192.168.0.220, Dst: 192.168.0.235
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    Differentiated Services Field: 0x03 (DSCP: CS0, ECN: CE)
        0000 00.. = Differentiated Services Codepoint: Default (0)
        .... ..11 = Explicit Congestion Notification: Congestion Experienced (3)
(…)
```

ECN with TCP (DCTCP)

- If traffic rate exceed threshold => switch set Congestion experienced (CE set)
- The receiver set the ECE flag in the TCP header in the acknowledge packet
- Servers needs to support ECN (`net.ipv*.tcp_ecn=1`)

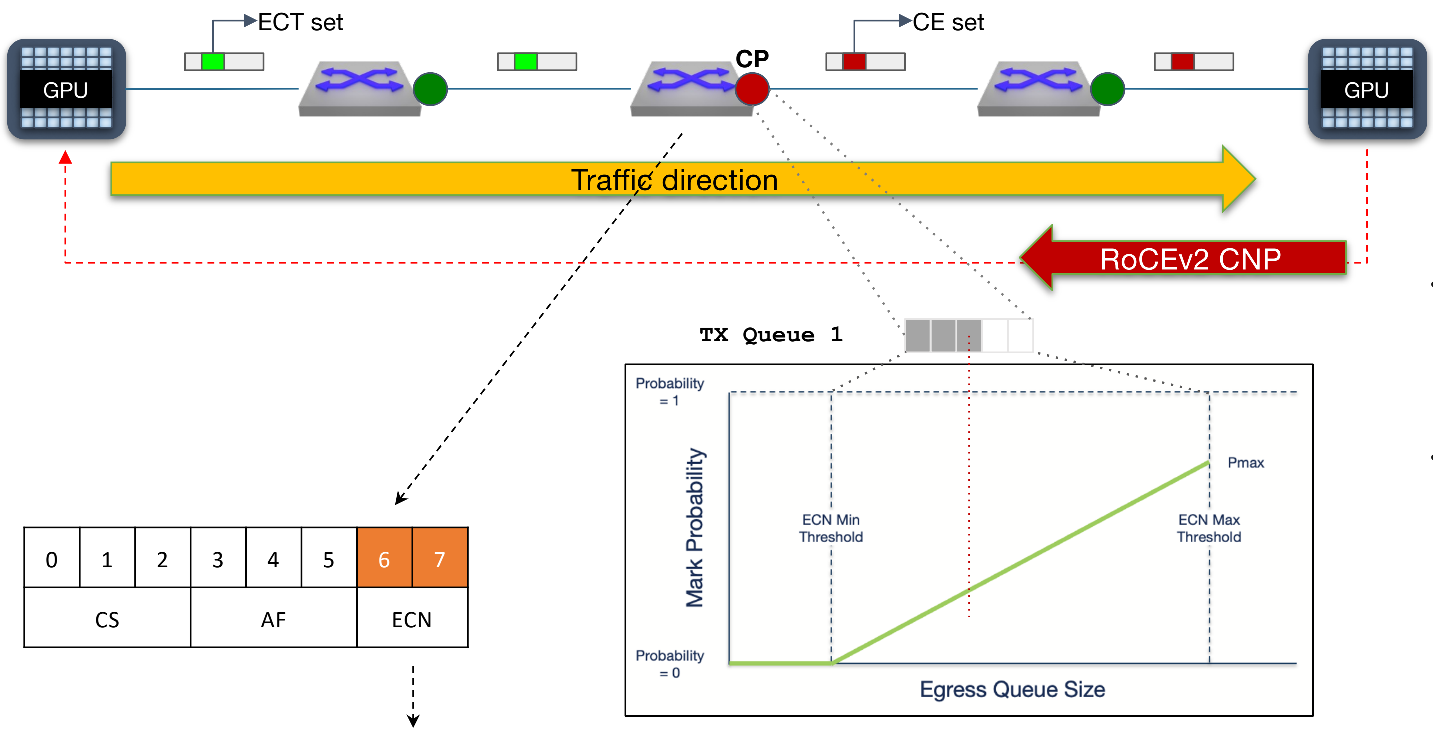
```
Internet Protocol Version 4, Src: 192.168.0.235, Dst: 192.168.0.220
(…)
    Differentiated Services Field: 0x03 (DSCP: CS0, ECN: CE)
        0000 00.. = Differentiated Services Codepoint: Default (0)
        .... ..11 = Explicit Congestion Notification: Congestion Experienced (3)
(…)
Transmission Control Protocol, Src Port: 48802, Dst Port: 5555, Seq: 357657, Ack: 1, Len: 1448
(…)
```

```
Internet Protocol Version 4, Src: 192.168.0.220, Dst: 192.168.0.235
(…)
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
(…)
    Transmission Control Protocol, Src Port: 5555, Dst Port: 48802, Seq: 1, Ack: 359105, Len: 0
    Flags: 0x050 (ACK, ECE)
        000. .... .... = Reserved: Not set
        ...0 .... .... = Accurate ECN: Not set
        .... 0... .... = Congestion Window Reduced: Not set
        .... .1.. .... = ECN-Echo: Set
        .... ..0. .... = Urgent: Not set
        .... ...1 .... = Acknowledgment: Set
        .... .... 0... = Push: Not set
        .... .... .0.. = Reset: Not set
        .... .... ..0. = Syn: Not set
        .... .... ...0 = Fin: Not set
(…)
```



The Addition of Explicit Congestion Notification (ECN) to IP
https://datatracker.ietf.org/doc/rfc3168/

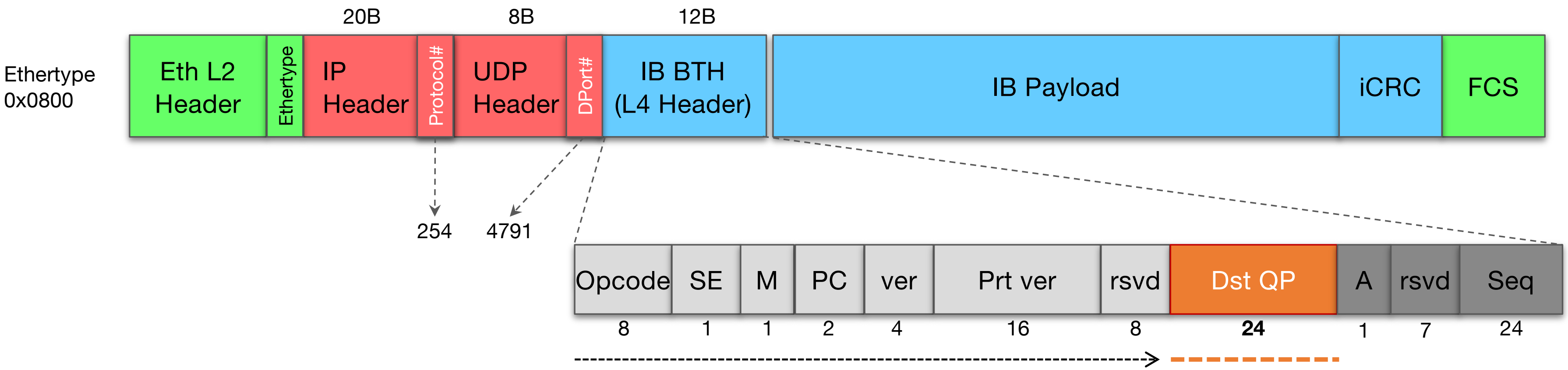
RDMA = UDP, CNP acting for TCP ECE



- Congestion Experienced (CE) Detected set by the Congestion Point (CP)
- Inform the Sender via RoCEv2 Congestion Notification Packet (CNP)

```
qos profile rocev2-wred
 tx-queue 1
  no priority
  random-detect ecn minimum-threshold 256 kbytes maximum-threshold 512 kbytes max-mark-probability 100 weight 0
```

RDMA Aware Load Balancing

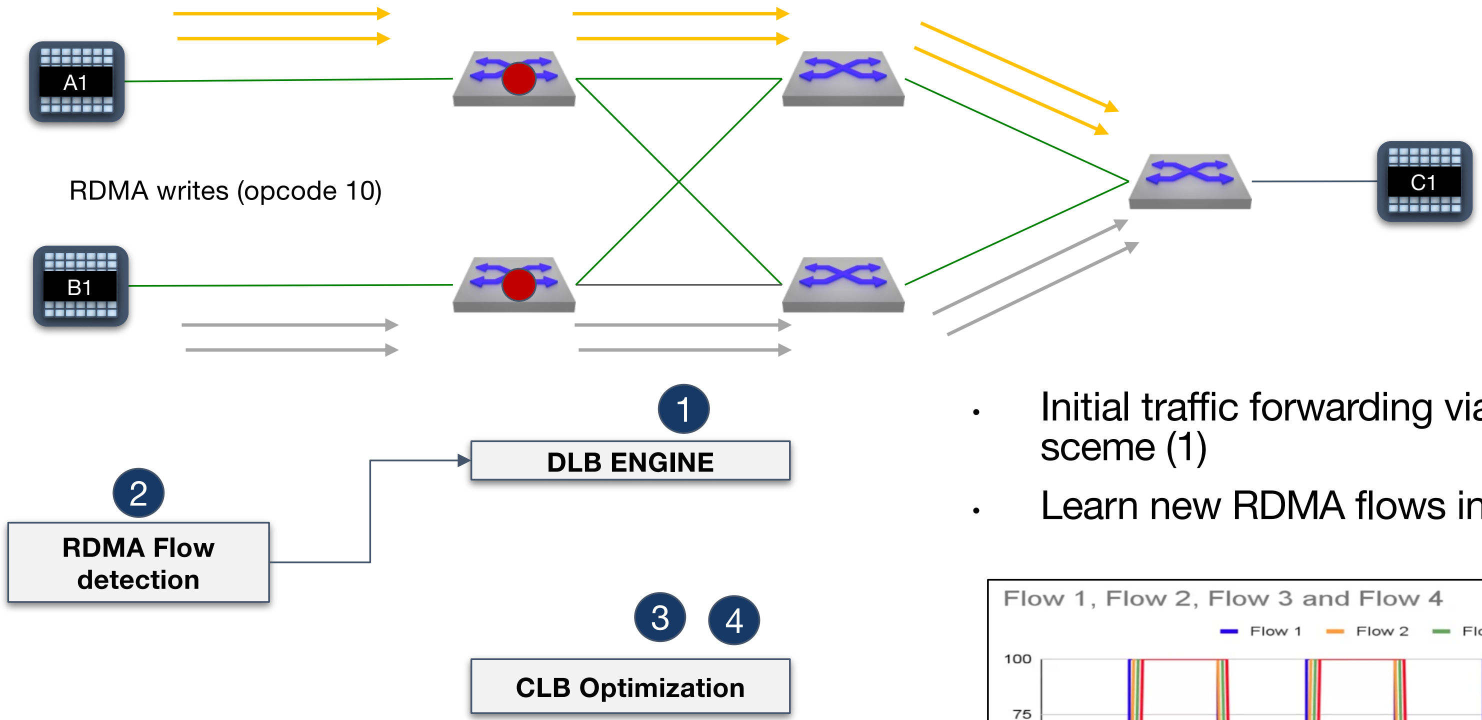


Hash based on destination Queue Pair field

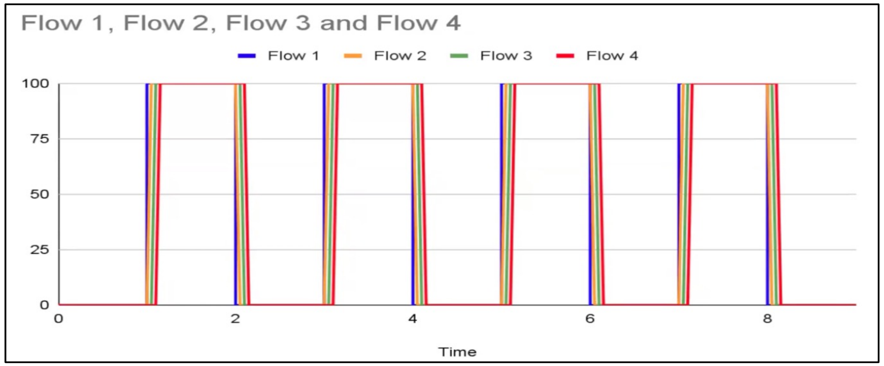
- Jobs run typically with 4… 64 queue-pairs (QP)
- Number of flows is roughly twice the number of QPs => 4*QPs at 400G translates to 8x50G flows
- The Dst QP starts at offset 0x5B (40bit) and has length of 0x3B (24bit)

```
load-balance policies
   load-balance sand profile default
        fields udp dst-port 4791 payload bytes 5-7
  (…)
```

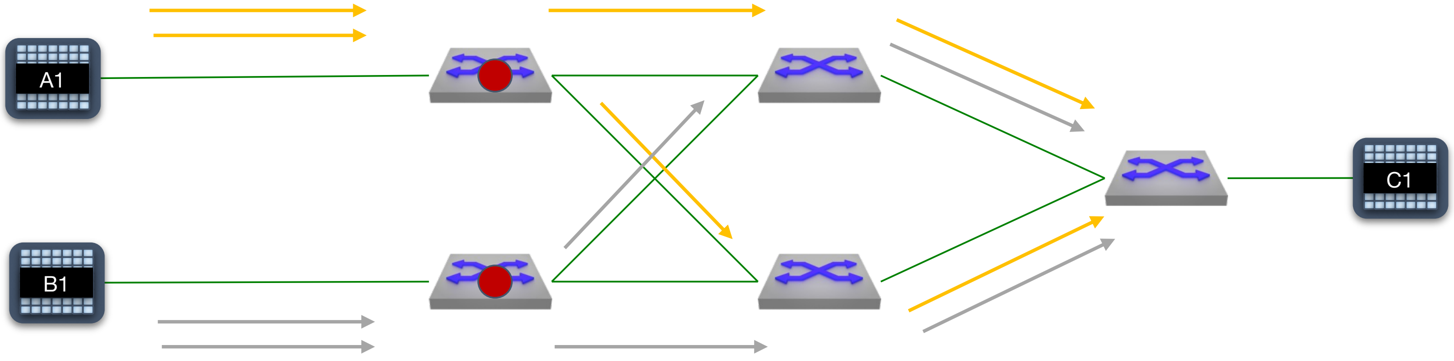
Cluster Load balancing, learn the flow(s)

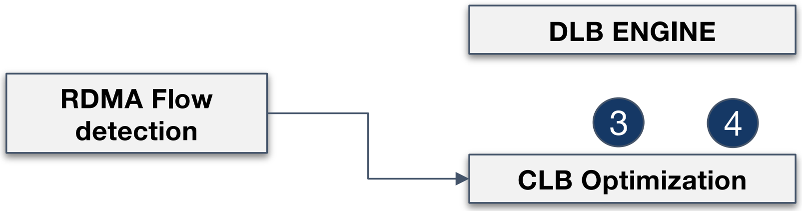


- Initial traffic forwarding via default load-balance sceme (1)
- Learn new RDMA flows in the network (2)



Cluster Load balancing…



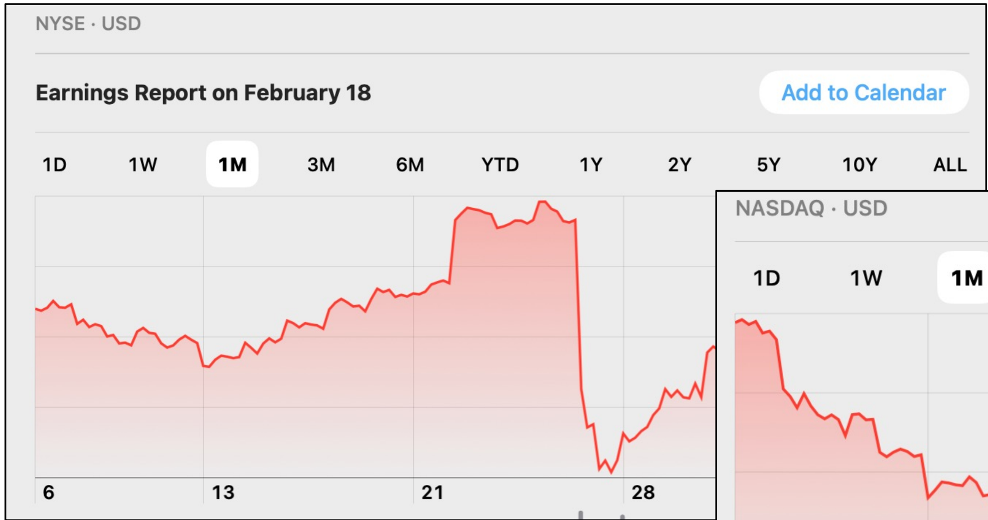


- Identify flows that are part of the same collective (3)
- Allocate these flows to evenly links (4)

```
switch(conf)#load-balance cluster
switch(conf-clb)#forwarding type bridged encapsulation vxlan ipv4
switch(conf-clb)#load-balance method flow round-robin
switch(conf-clb)#flow source learning
switch(conf-clb-flow-learning)#aging timeout 60 seconds
switch(conf-clb)#port group host server1
switch(conf-clb-portgroup-server1)# interface Et15/1, Et16/1…
switch(conf-clb-portgroup-server1)#flow limit 800
```

However… its expensive

- Multi-GPU Fabric(s) expensive…
- Market looking for alternative models running over less expensive hardware
- Alt ways of deploy the workloads



AI capacity from AI Provider

AIaaS

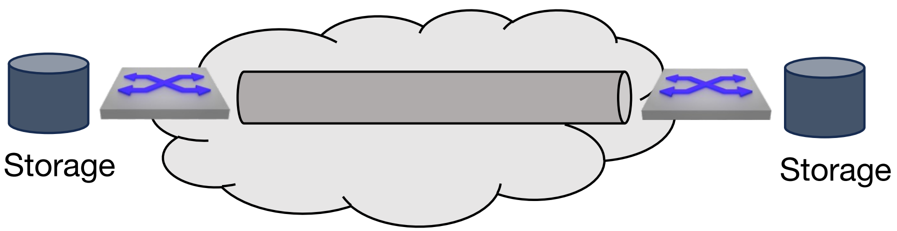
- Running AI in the Cloud, Chatbots ?

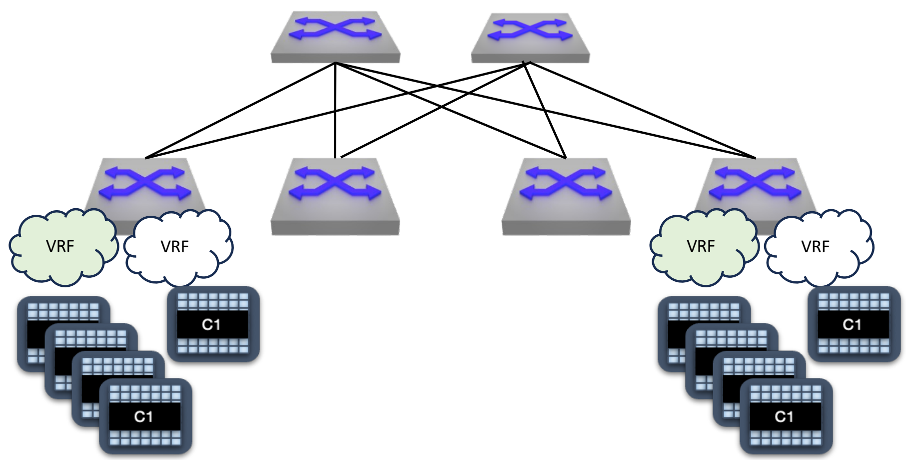
"Without Data, AI Means Nothing”

- How to secure transport “Data” over WAN, IPSEC, TunnelSEC ?

Segment customers from each other

- Segmentation ? EVPN, that is segment storage with VMs, GPUs with VLAN/VRF…





Running AI Workloads outside or between DC ?

Is it even doable (with current) Collective Communication behavior ?

- Which none DC Back-end network can handle 400Gbps flow(s) GPU<>GPU ?
  - DC have much more BW than Cores (No shit)… thereby todays Multi-GPU solution adapted to high-throughput, low-latency and packets arrive in the right order

- MPLS/VXLAN encapsulation and throttling feedback ?
  - ECN/Pause frames pointless since encapsulated end-to-end, temporary networking challenges needs to be address with other means than lossless features like PFC/ECN

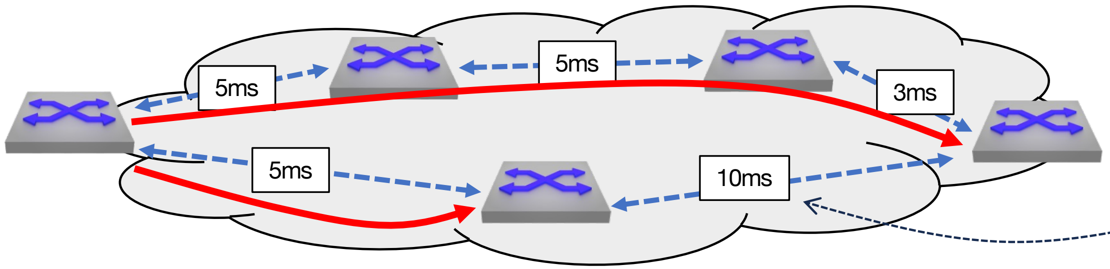
- SRv6 and micro-SID that seems to fix everything from hangover to networking ?
  - Can't see any golden nugget compare to example MPLS headers, however more fields to play with example the flow label field…

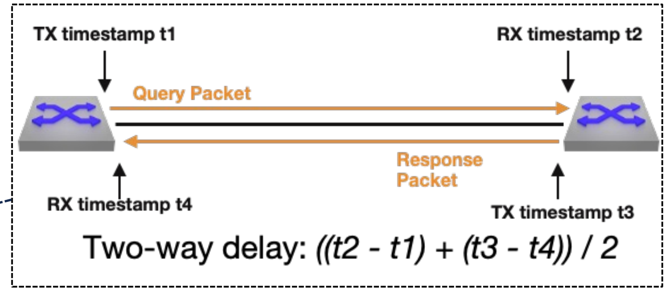
```
RFC 1925 The Twelve Networking Truths
(…)
(5) It is always possible to aglutenate multiple separate problems into a
single complex interdependent solution. In most cases this is a bad idea.
(…)
```

20th Cores dumb ?

Not really, example both QOS and delay can be handle pretty good with todays features

- Example MPLS-SR Flexalgo&TWAMP can steer the path based on the current delay/load on each on the transit link(s) end-to-end by update TE Database using IS-IS TLV





However… the problem are physics and "spoiled childs behavior"

Whole loop time	158.57ms	158.96ms	158.52ms	160.63ms	189.38ms	161.10ms
CPU <> GPU transfer time	1.37ms	1.29ms	1.32ms	1.34ms	1.37ms	1.33ms
Forward/backward time	34.49ms	34.50ms	34.48ms	34.54ms	34.50ms	34.52ms
Grad sync time	119.97ms	120.81ms	120.11ms	122.02ms	150.95ms	122.55ms
Whole model time	155.20ms	155.94ms	155.27ms	157.21ms	186.12ms	157.70ms

Change communication model(s)

Alternative Collective Communication models over WAN

- Lower the speed(s) ?
  - Works, but “takes the air out” of Multi-GPU design
- More Banner allow more time for synchronization (and write)
  - Probably the same result as above, much tuning needed
- Move to other protocols example NWMe, iWARP or move to QUIC ?
  - TCP slow&complex state machine… even with SACK and Fast-recovery
  - QUIC have easier flow control and could support ECN similar to RoCEv2 CNP
- Single GPU communication design
  - Only distribute “data” from Storage to each remote GPU and result write(s) back over WAN ?
  - In theory work as it would be in a DC, however this is a goodbye kiss to the parallelism

The future is yesterday

- Neural networking is not something new,,, neither linear algebra
- Things just happens... Example the breakthrough introduction of the backpropagation and gradient descent algorithm (1986). Suddenly a company most famous for it's graphical cards in PCs, introduced the usage of GPU for none-graphical applications (2006).
- New open-source models like Deepseek, With less needs of expensive hardware or even other ways of Collective Communication ? Of course... I mean we are far from Metcalfe's half-duplex ethernet with todays 800Gbps ethernet
- **However…** parallelism here to stay, and *any* model totally useless without good "data" to be trained on/with. Where there is available capacity regards to compute&storage… workloads and movement of "data" will most likely follow