Networks for Machine Learning Jobs

Manya Ghobadi
MIT CSAIL

ghobadi@csail.mit.edu

The machine learning storm

ChatGPT: fastest growing online application ever

Just one month:

- 600 million live inference queries
- 100 million unique visitors (Instagram took 2 years)

- Disruption and innovation in search, content creation, code/SQL generation, DevOps assistance, tutoring, translation, …

Massive implications for systems and networks

https://www.theguardian.com/technology/2023/feb/02/chatgpt-100-million-users-open-ai-fastest-growing-app

Two important metrics for ML systems

- Latency:
  - Training ChatGPT took ~2 million GPU hours (200 years with one GPU)
  - Live inference query response time should be less than 100 ms

- Energy consumption:
  - ChatGPT's monthly electricity consumption is in the millions of KWh
  - Energy of serving inference queries for a month is larger than training

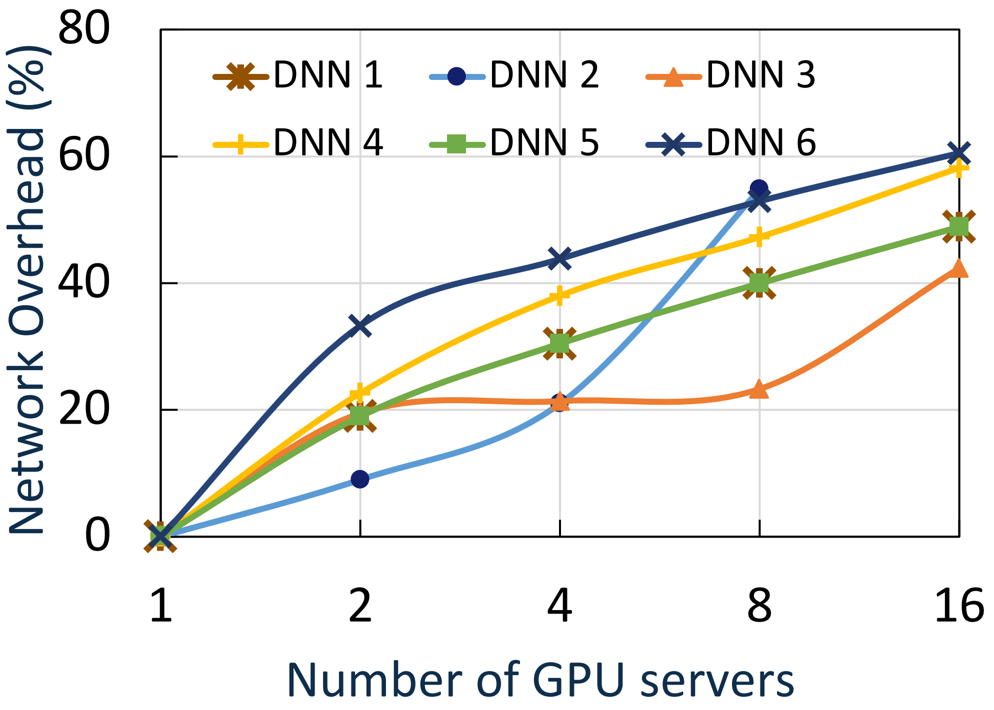
https://towardsdatascience.com/chatgpts-electricity-consumption-7873483feac4

We need

fast and energy-efficient ML systems.

State-of-the-art: application-agnostic datacenters

- Congestion control protocols
- Scheduling algorithms
- Network topology
- End-host capabilities

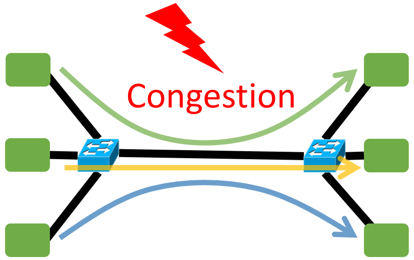


Implication: existing datacenter networks are becoming bottlenecks for ML training and inference jobs.

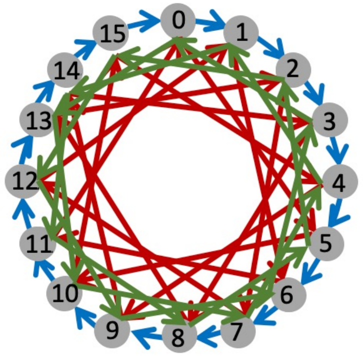
High-level message of the talk

Networking techniques to build high-performance *ML-centric* datacenters.

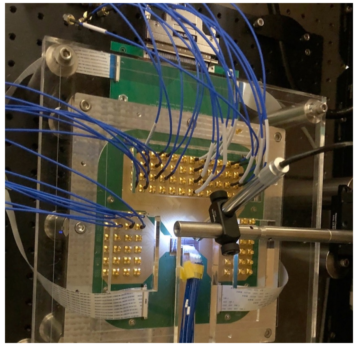
Talk outline: three key lessons



Fair congestion control is sometimes inefficient [HotNets'22, NSDI'24].



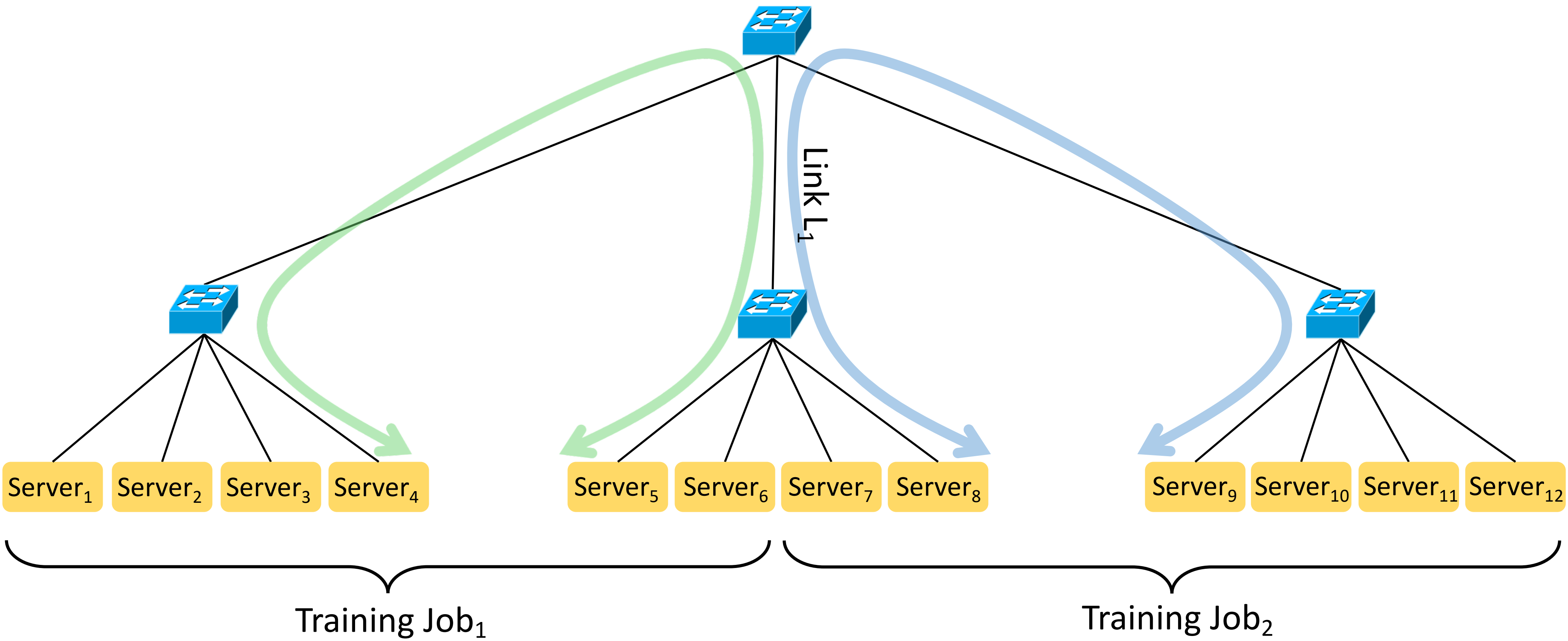
Reconfigurable networks for ML training [SIGCOMM'21, NSDI'23].



Analog computing for ML inference [SIGCOMM'23, Science'22, OFC'22].

Network congestion in ML datacenters

- TCP or RDMA congestion control protocols.
- DNN schedulers place workers based on topological proximity.
- In large datacenters cross-job network contention is inevitable.

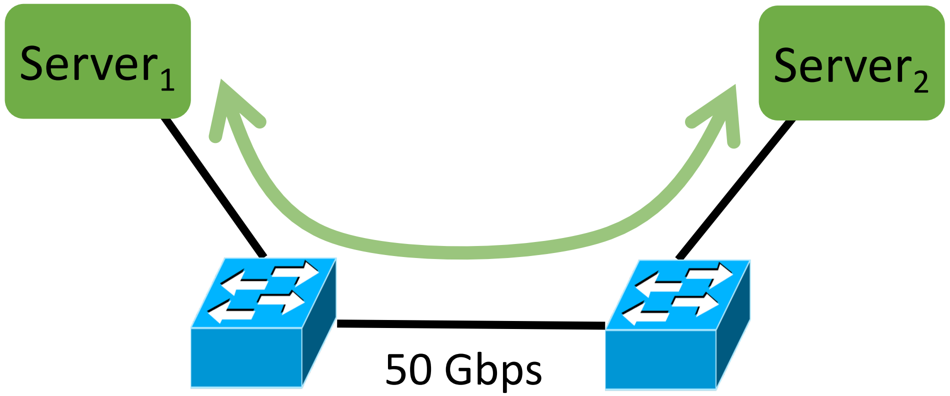


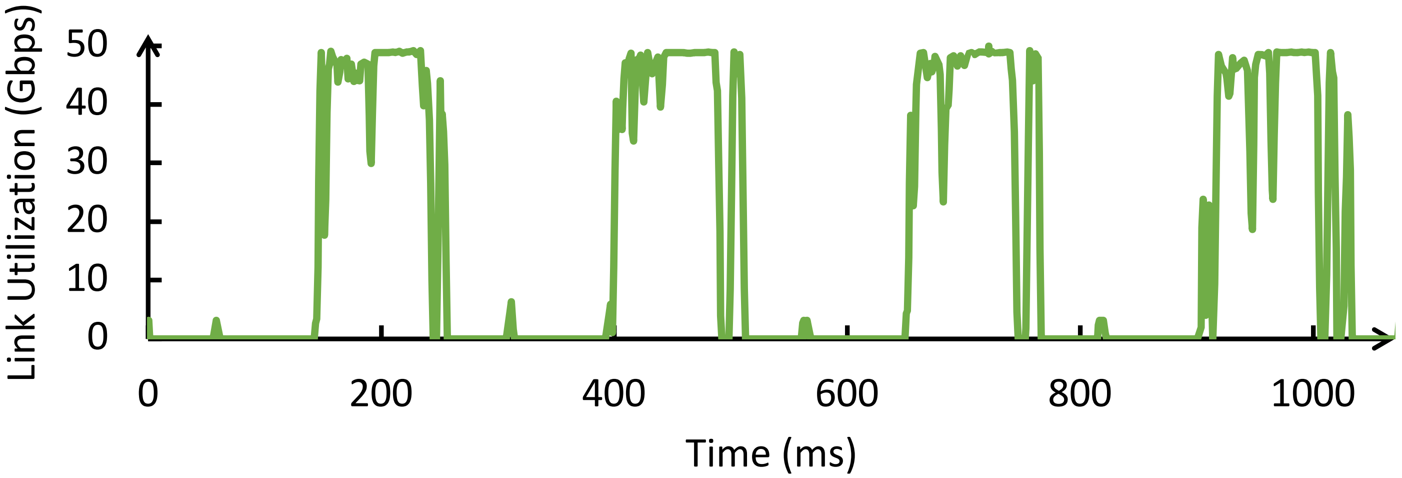
What is the impact of congestion control algorithms when ML jobs share network links?

Fair congestion control protocols are not necessarily beneficial for ML workloads!

[HotNets'22] Congestion Control in Machine Learning Clusters, S. Rajasekaran, M. Ghobadi, G. Kumar, A. Akella

Communication pattern of DNN training

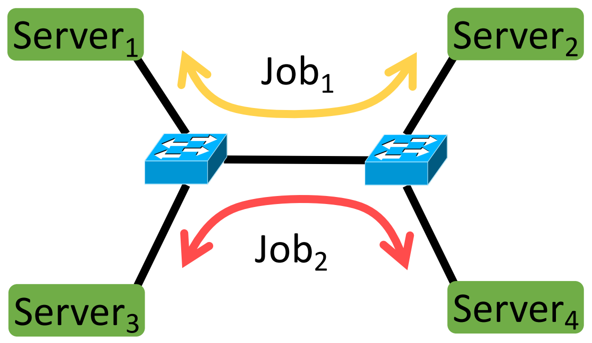


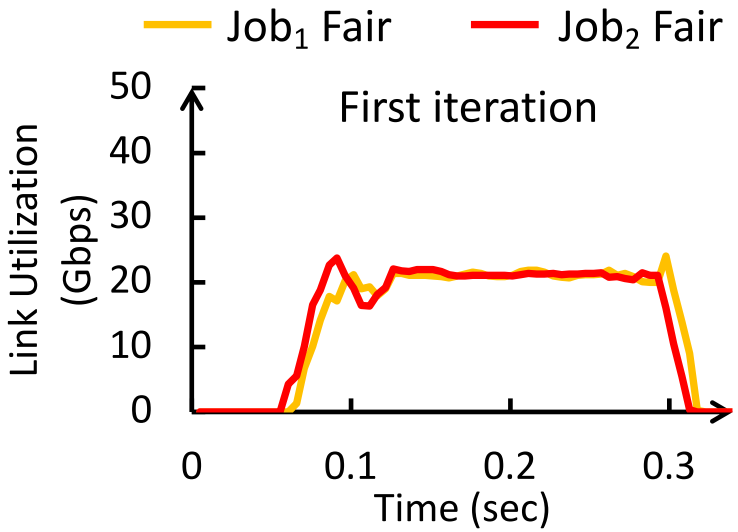


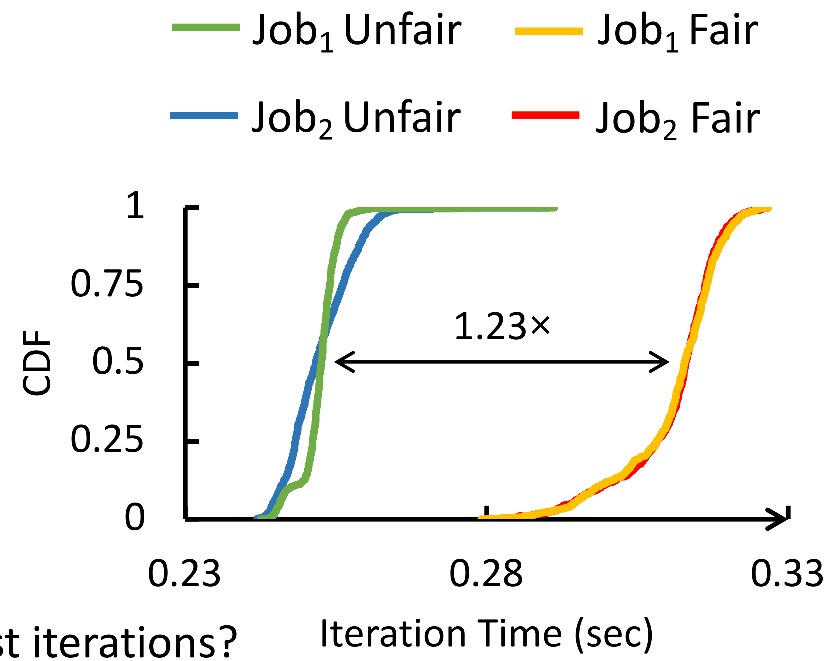
DNN training has a periodic up-down pattern of network demand.

Surprising payoff of unfairness

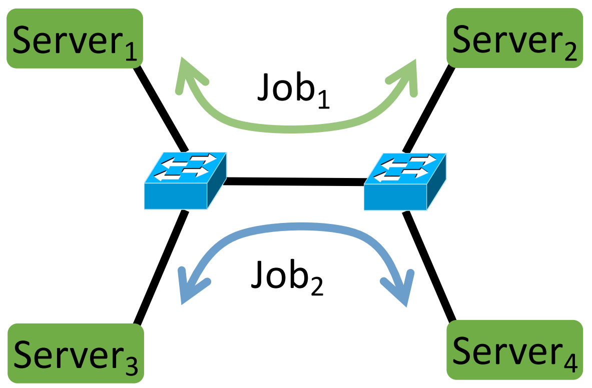
50% – 50% bandwidth sharing

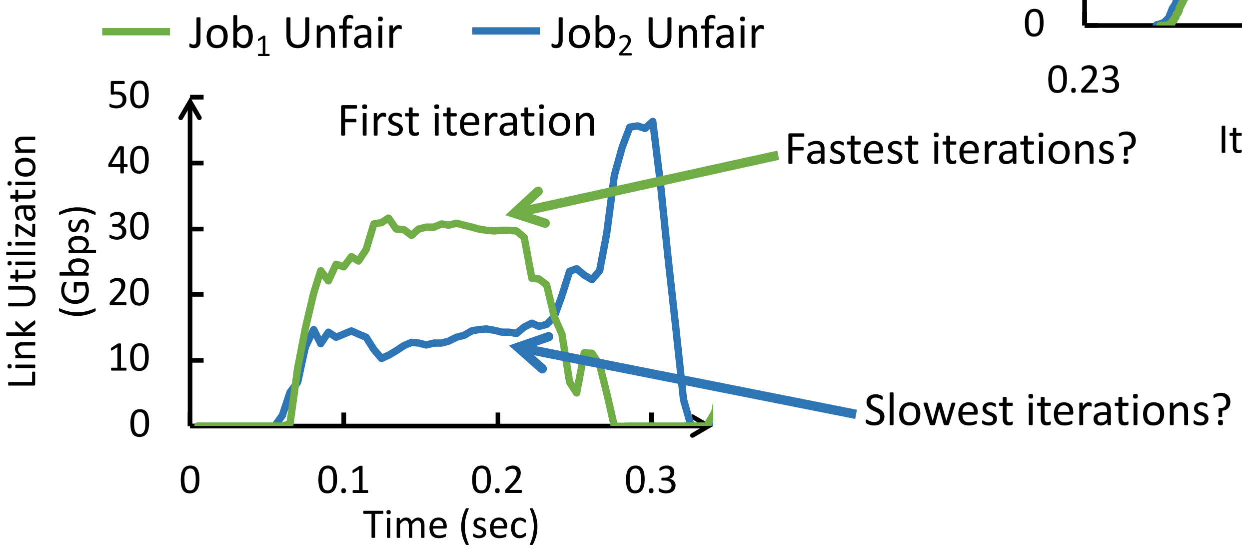




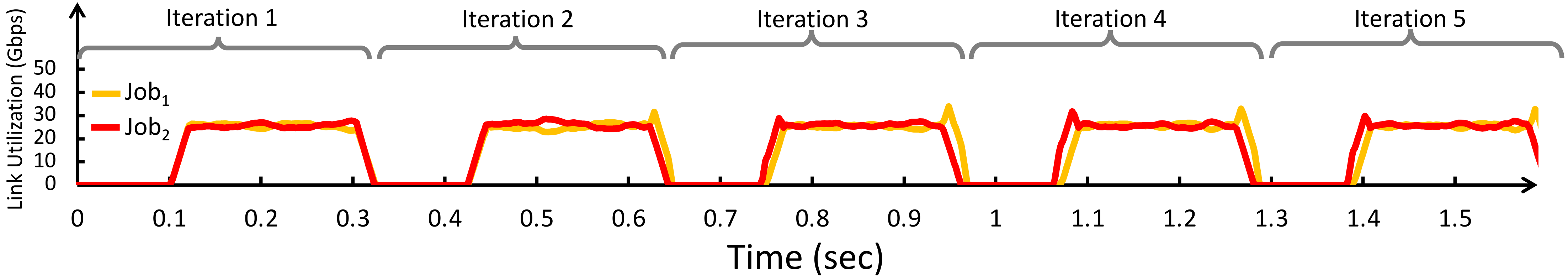


60% – 40% bandwidth sharing

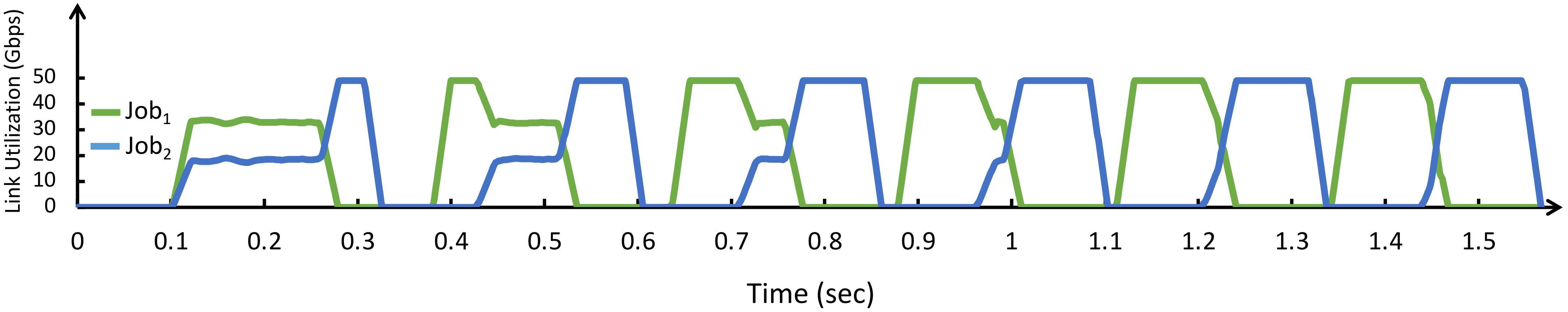




Why does unfairness help ML training?



Fair bandwidth sharing



Unfair bandwidth sharing

Can unfairness interleave all DNN training jobs?

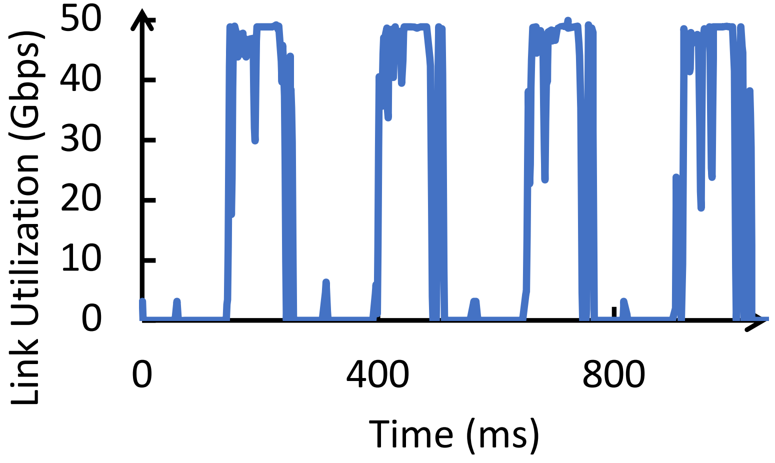
Unfairness doesn't always help

Job combination	Speed-up from Unfairness	Compatible
VGG11 (image recognition) VGG11 (image recognition)	1.05x 0.86x	✗
DLRM (recommendation) DLRM (recommendation)	1.3x 1.28x	✓
BERT (language) VGG19 (image recognition)	1.17x 0.94x	✗
VGG19 (image recognition) VGG16 (image recognition) ResNet50 (image recognition)	1.18x 1.18x 1.01x	✓

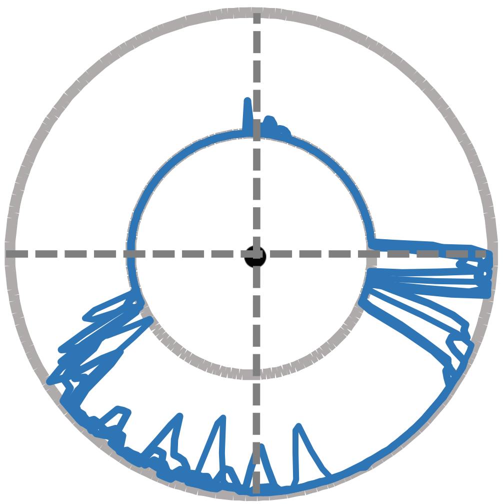
Compatible jobs are a group of jobs for which unfairness results in faster iteration times for all the jobs in the group.

Which job are compatible?

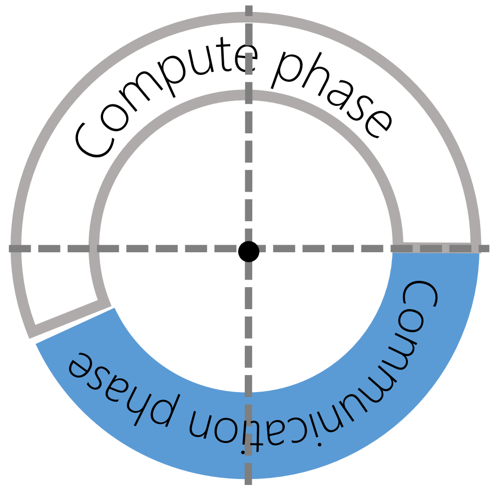
- Challenges:
  - Interleaving must be checked across thousands of iterations across many jobs
  - Different jobs have different iteration times and communication durations
- Our solution: a geometric abstraction



Network demand



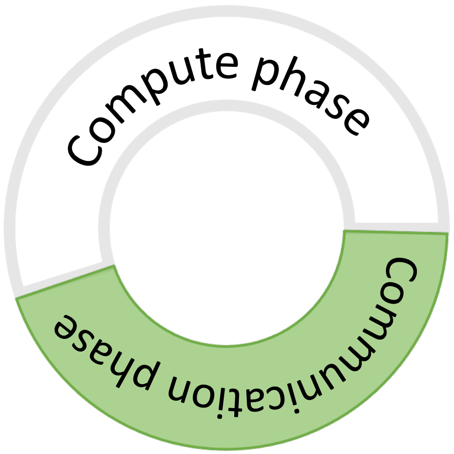
Network demand rolled around a circle



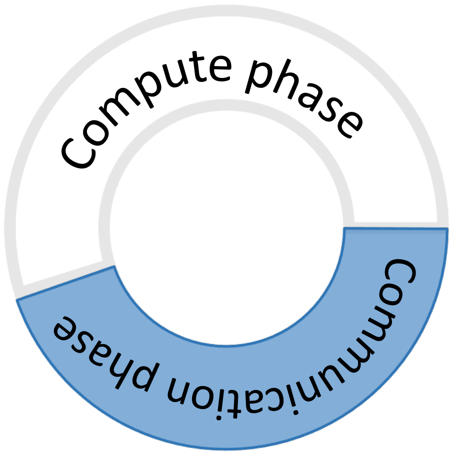
Geometric representation

Determining job compatibility

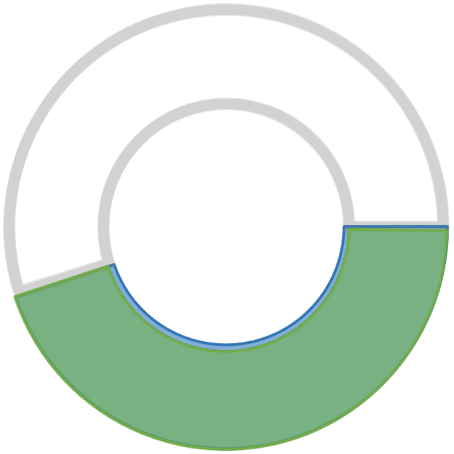
- Fully compatible jobs: Two DLRM models



Job$_1$: DLRM

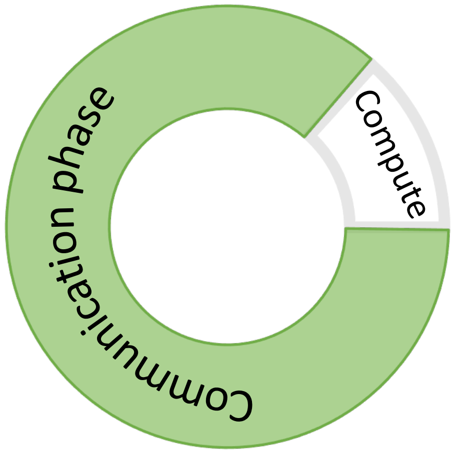


Job$_2$: DLRM

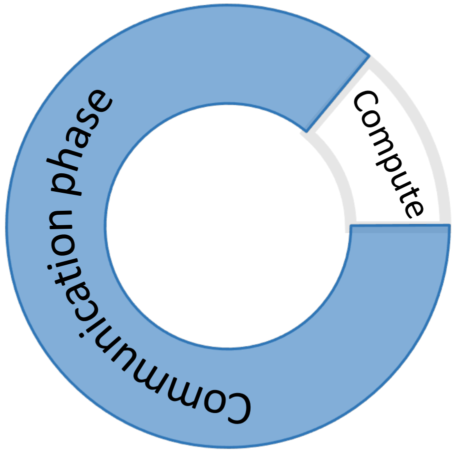


Fully compatible

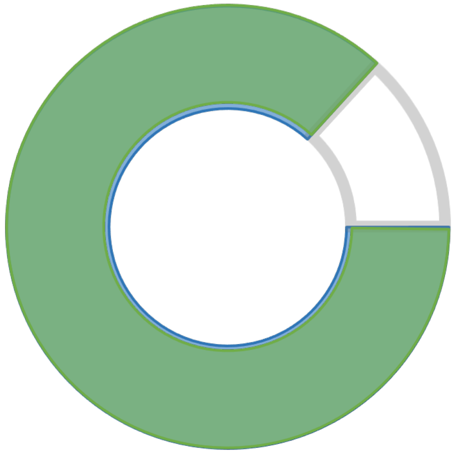
- Partially compatible: Two VGG11 models



Job$_1$: VGG11



Job$_2$: VGG11

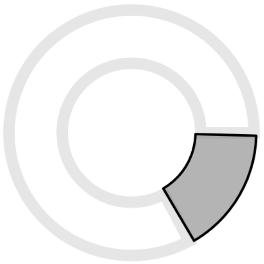


Partially compatible

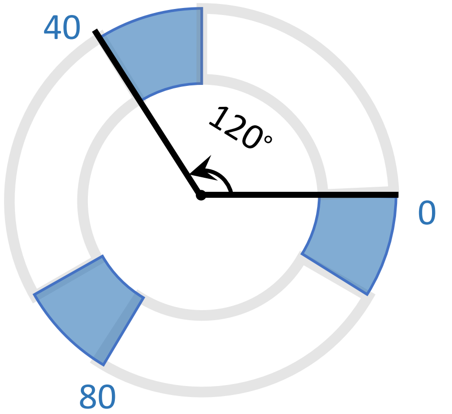
Challenge:
Jobs with different iteration times sharing a link

Solution: Use Least Common Multiple of iteration times to create unified circle

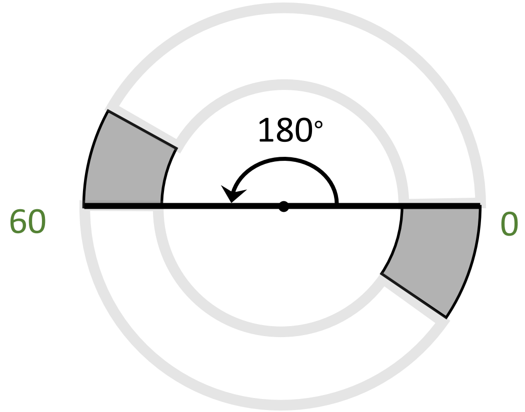
Job$_1$: 40ms
Iteration time



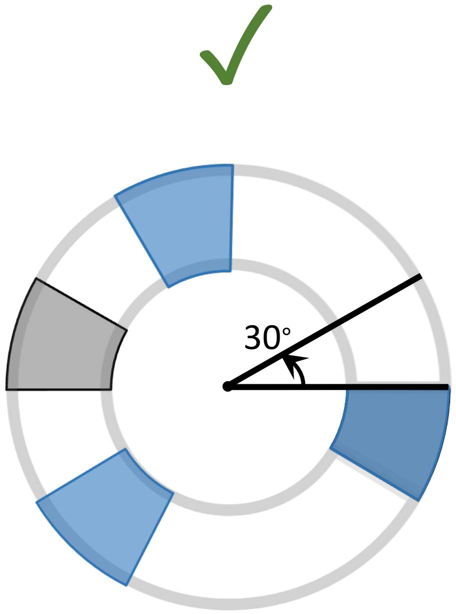
Job$_2$: 60ms
iteration time



Job$_1$ on unified circle



Job$_2$ on unified circle



- We translate the problem of compatibility to an optimization formulation to find rotation angles

Computing rotation angles

Input	$J^l = \{j\}$	Set of ML jobs $j \in J^l$ competing on link l.
	$\{\text{unified\_circle}_j\}$	Set of unified circles for $\forall j \in J$. Each circle is a data structure that contains the angles and bandwidth demand of Up or Down phases.
	$bw\_circle_j(\alpha)$	Bandwidth demand at angle α on unified_circle$_j$
	r_j	Number of iterations of j in its unified_circle$_j$.
	$A = \{\alpha\}$	Set of discrete angles $\alpha \in [0,2\pi]$. $\lvert A\rvert$ denotes the number of discrete angles.
	C^l	Total link capacity of link l.
Output	$demand_\alpha$	Total bandwidth demand at angle α when considering the demand of all jobs $j \in J$.
	Δ_j^l	Rotation angle of $j \in J$ on link l, in radians.
	$score$	Compatibility score of jobs sharing link l.

Set of jobs and their compute/communication phases

Compatibility score and rotation angle for each job

Auxiliary definitions:

$$Excess(demand_\alpha) = \begin{cases} demand_\alpha - C^l & if demand_\alpha > C^l \\ 0 & otherwise \end{cases} \tag{1}$$

$$\textbf{Maximize: } score = 1 - \frac{\sum_\alpha Excess(demand_\alpha)}{|A|C} \tag{2}$$

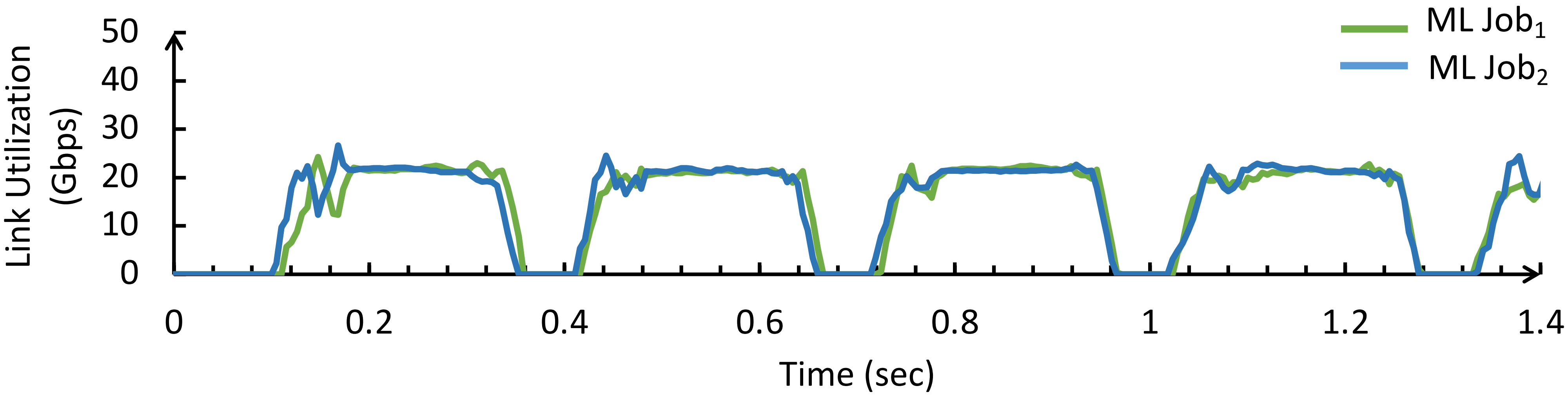
Subject to:

$$\forall \alpha : \sum_j bw\_circle_j(\alpha - \Delta_j^l) \leq demand_\alpha \tag{3}$$

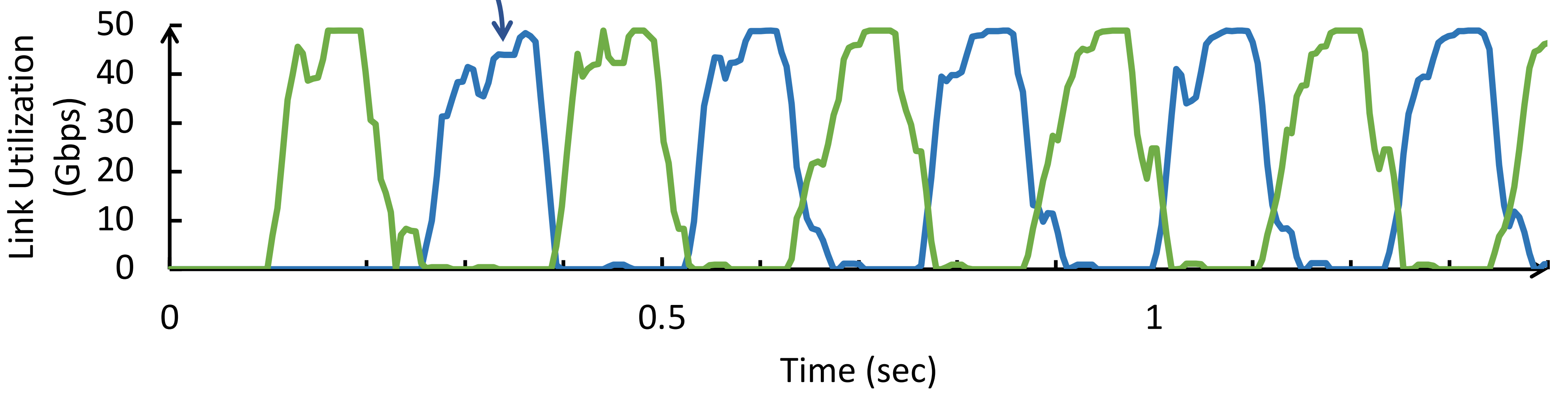
$$\forall \Delta_j^l : 0 \leq \Delta_j^l \leq \frac{2\pi}{r_j} \tag{4}$$

Minimize the overlapping region on geometric circle

Translating rotation angles to time-shifts



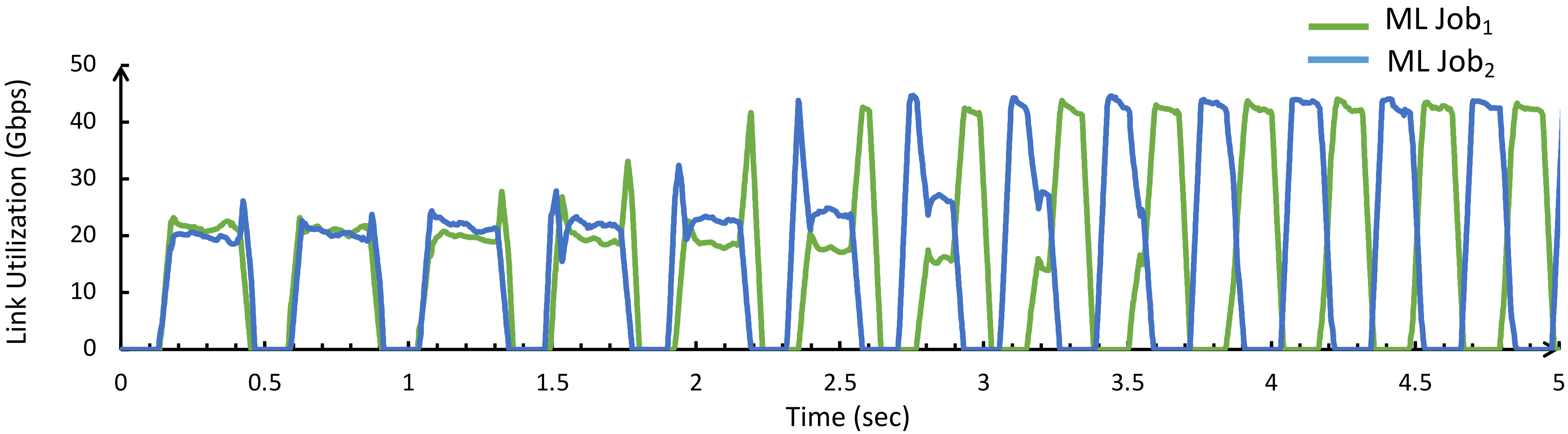
Start time of the first iteration is shifted



[NSDI'24] Cassini: Network-Aware Job Scheduling in Machine Learning Clusters, S. Rajasekaran, M. Ghobadi, A. Akella

Is there congestion control algorithm that can *automatically* stabilize to an interleaved state?

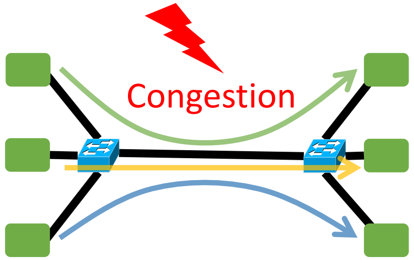
MLTCP: A congestion control scheme for ML



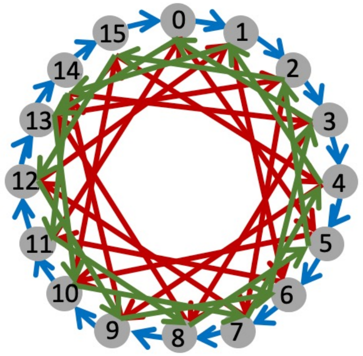
- MLTCP: A novel congestion control scheme for automatic interleaving of ML jobs

We are looking for partners from the Netdev community
(Email: ghobadi@mit.edu)

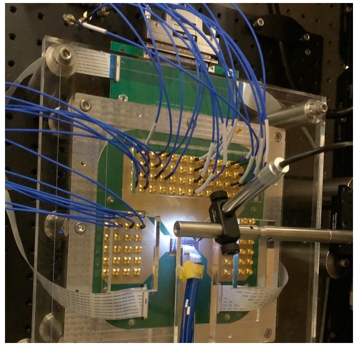
Talk outline: three key lessons



Fair congestion control is sometimes inefficient [HotNets'22, NSDI'24].



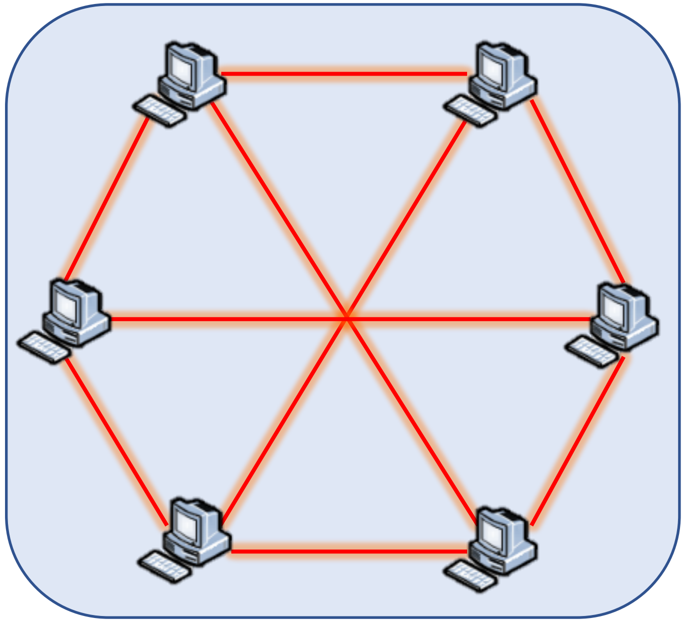
Reconfigurable networks for ML training [SIGCOMM'21, NSDI'23].



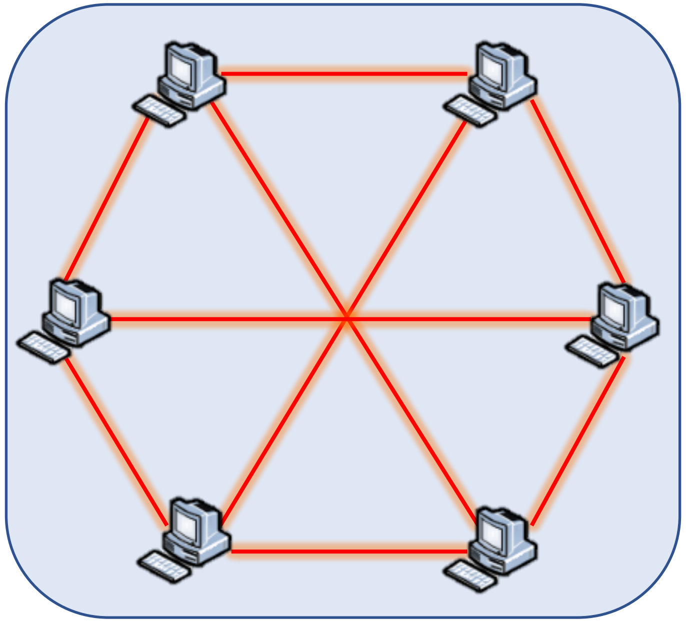
Analog computing for ML inference [SIGCOMM'23, Science'22, OFC'22].

Can we avoid cross job congestion all together with a clean-slate ML-centric optical datacenter?

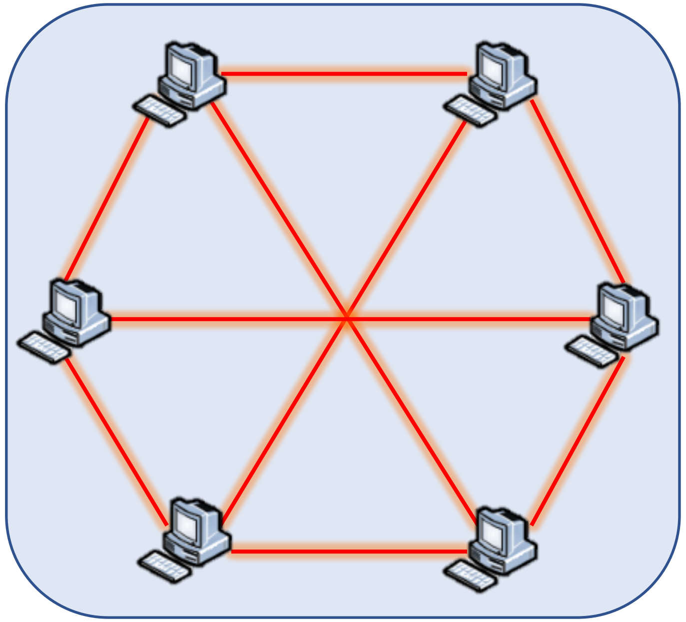
Reconfiguring physical network topology



Topology A

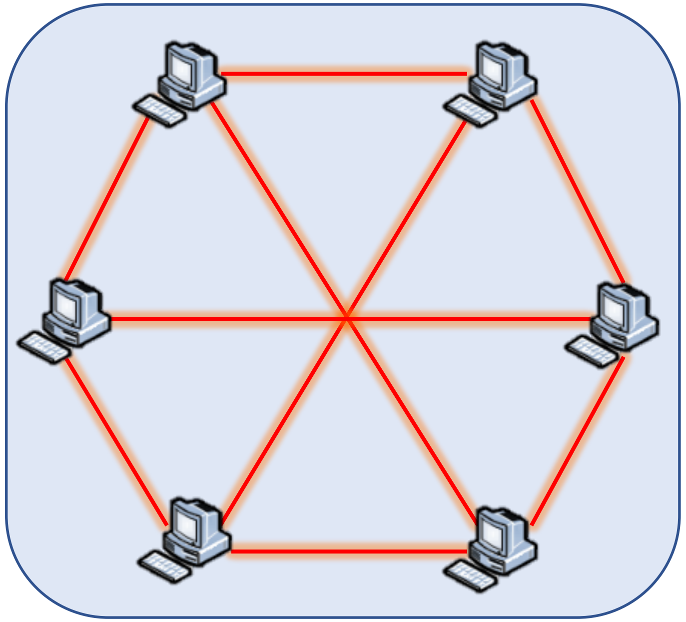


Topology A

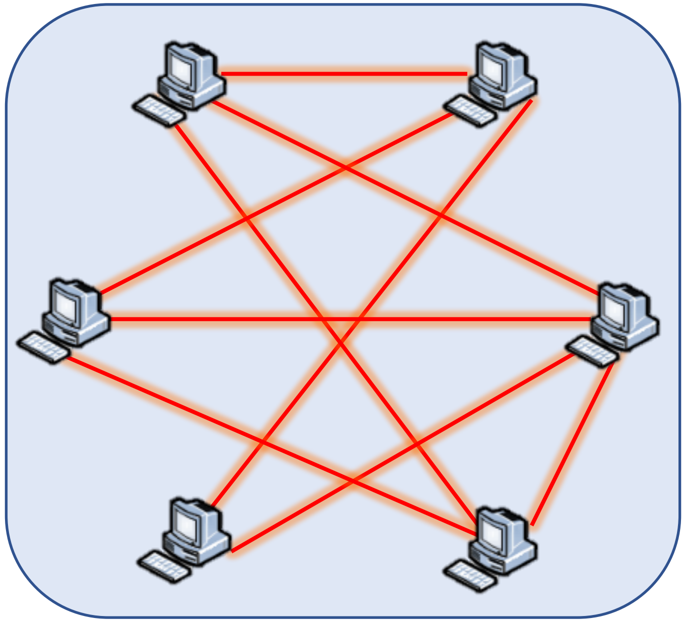


Topology A

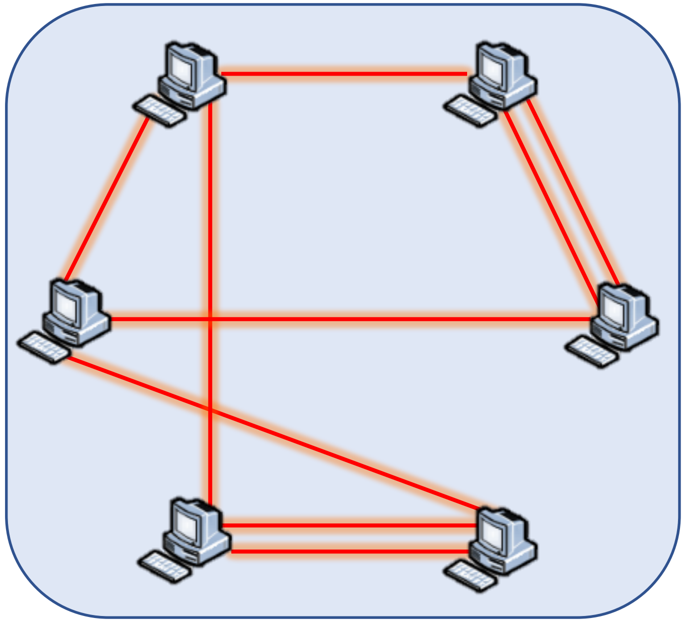
Reconfiguring physical network topology



Topology A

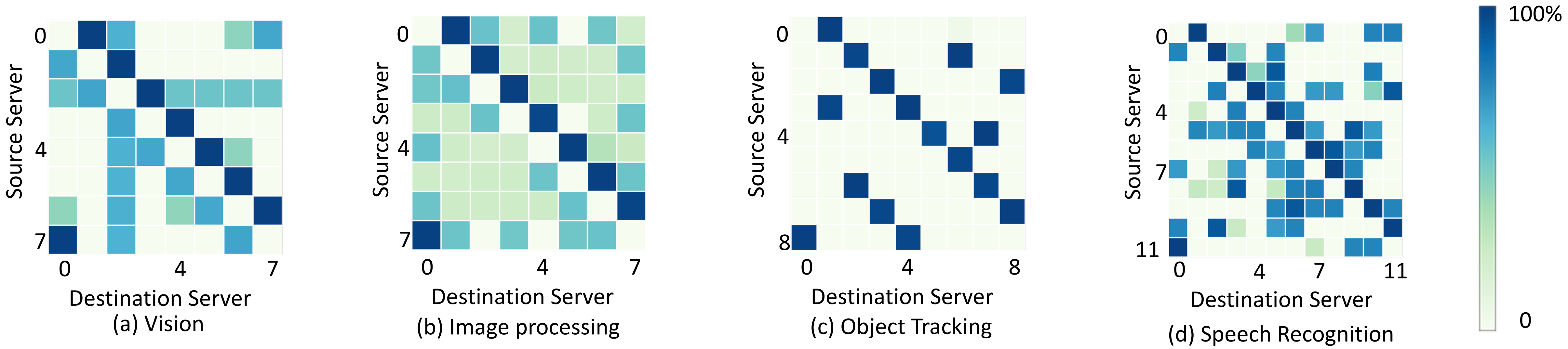


Topology B



Topology C

DNNs training jobs exhibit different traffic patterns



Traffic pattern for a job is predictable but different jobs have different traffic patterns.

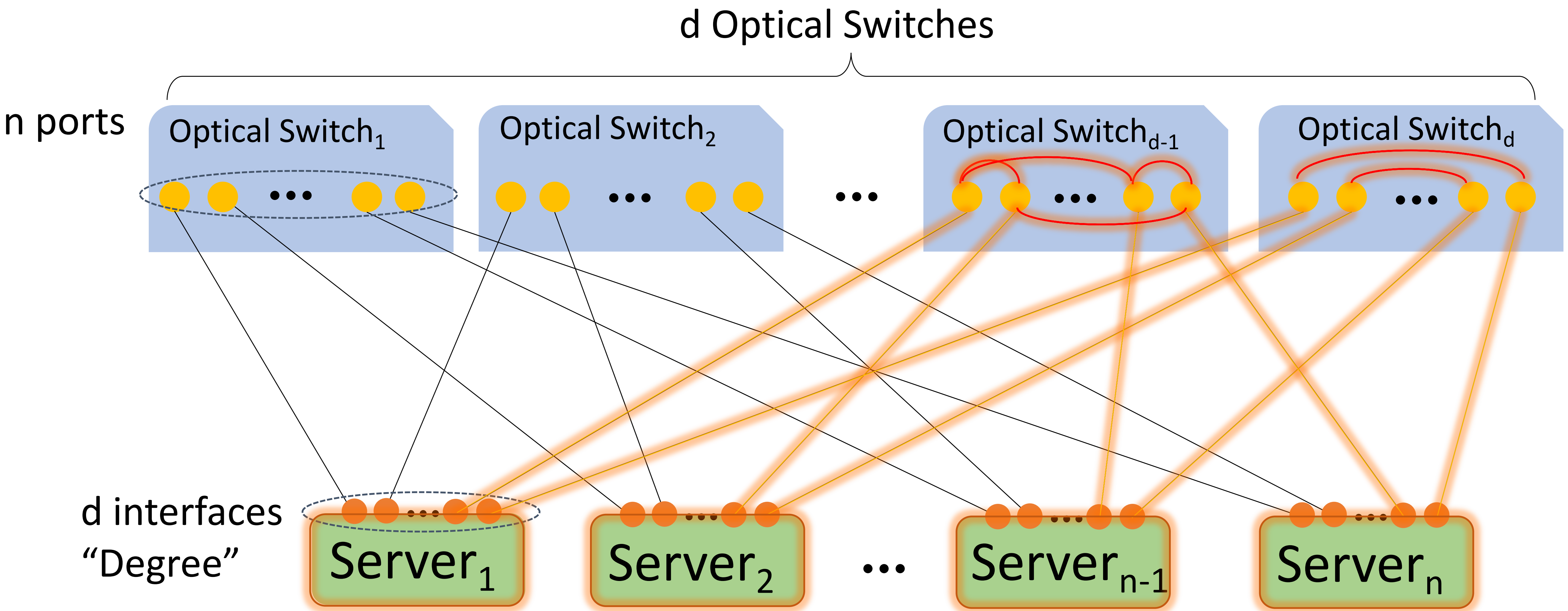
ML training workloads and optical interconnects: match made in heaven

- Building full-bisection bandwidth networks is expensive and unnecessary
  - Training traffic pattern repeats for the entire duration of a job (several hours to days)
  - DNN training jobs have widely different traffic patterns

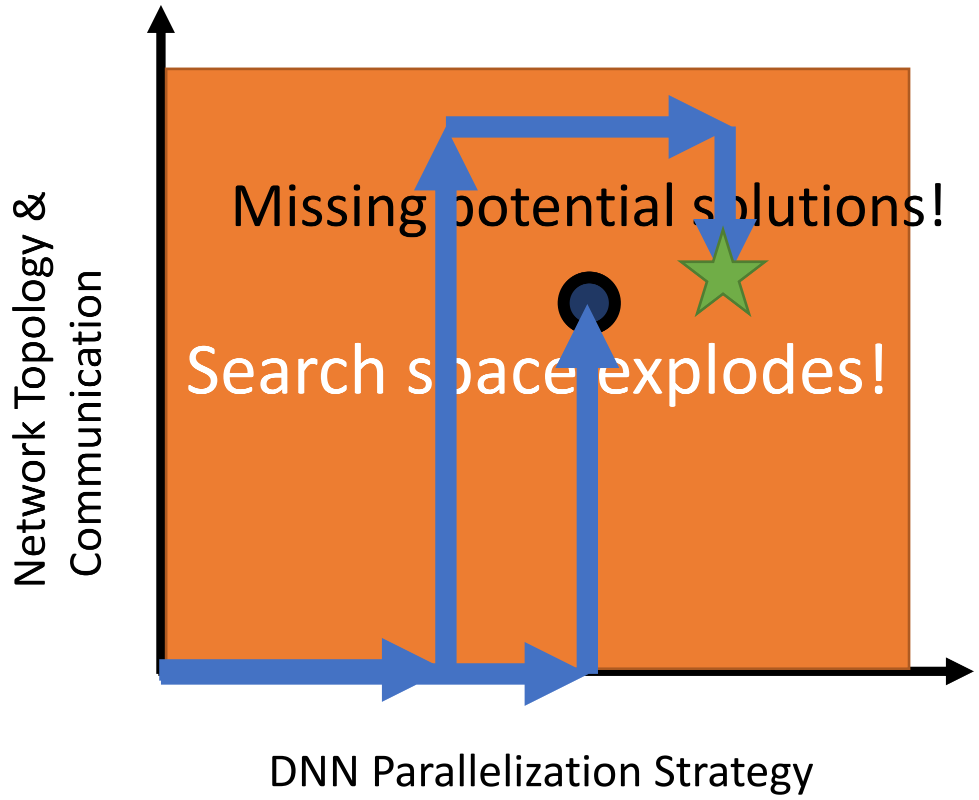
Key idea: a one-shot reconfigurable optical datacenter that partitions the network for each ML job.

[SIGCOMM'21] SiP-ML: High-Bandwidth Optical Network Interconnects for Machine Learning Training
M. Khani, M. Ghobadi, M. Alizadeh, Z. Zhu, M. Glick, K. Bergman, A. Vahdat, B. Klenk, E. Ebrahimi

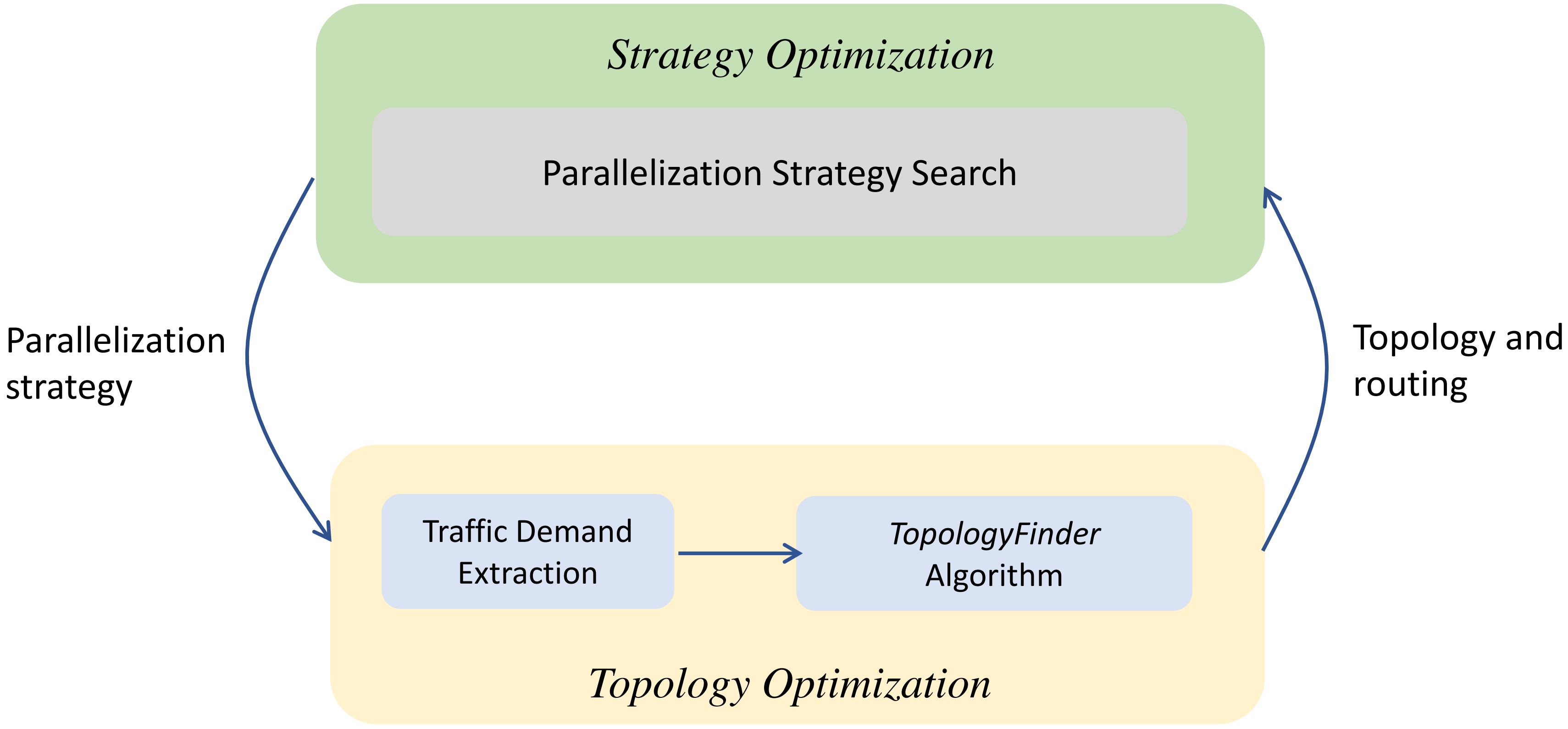
A reconfigurable interconnect for DNN training



Challenge: Huge search space

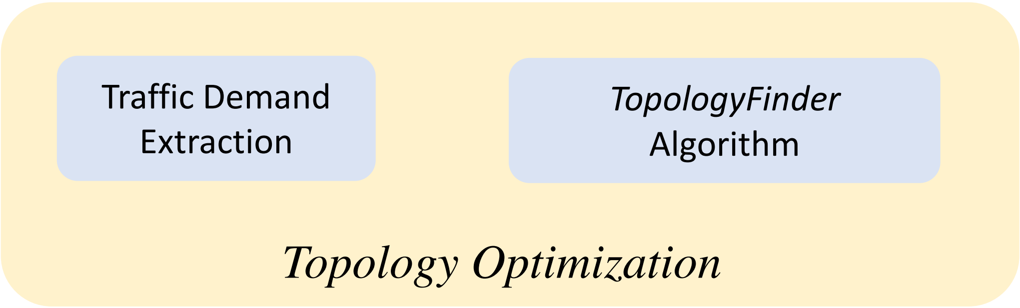


TopoOpt: alternating optimization framework



[NSDI'23] TopoOpt: Optimizing the Network Topology for Distributed DNN Training, W. Wang, M. Khazraee, Z. Zhong, M. Ghobadi, Z. Jia, D. Mudigere, Y. Zhang, A. Kewitsch

Alternating optimization framework

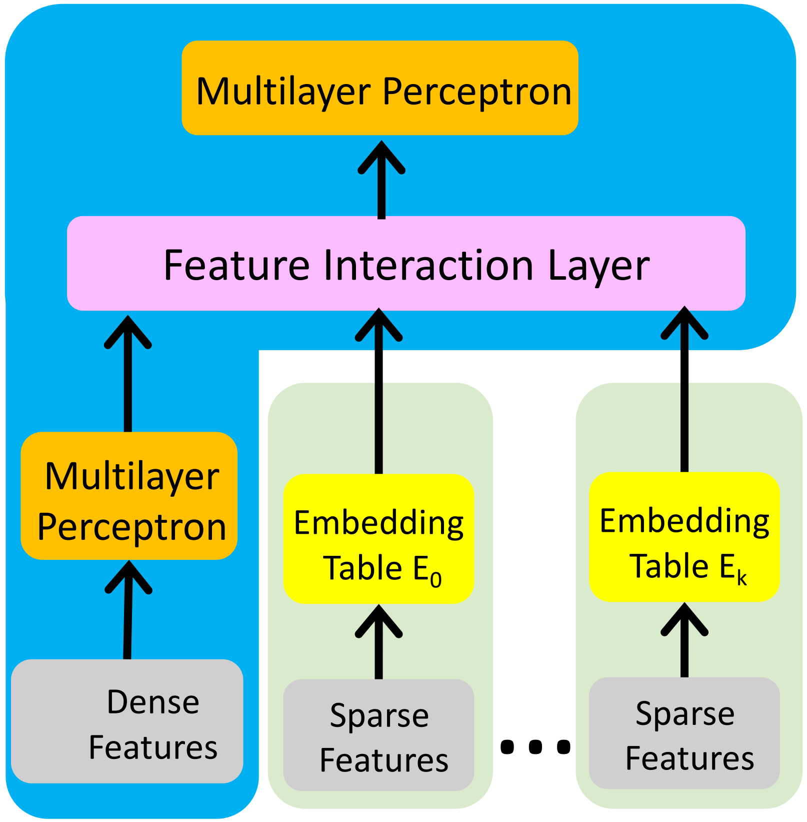


What is an ideal network topology for a given DNN training job?

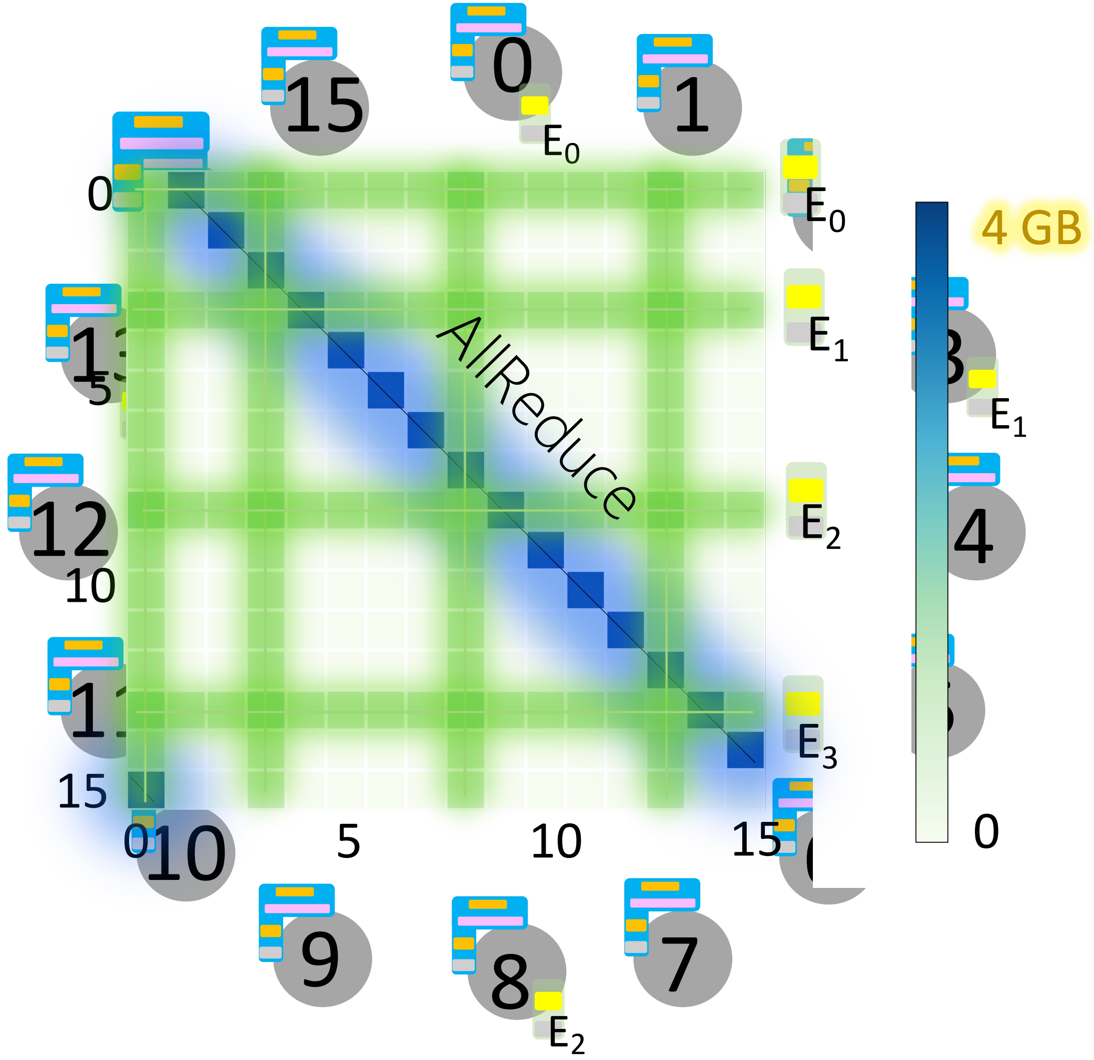
What is an ideal network topology for a given DNN training job?

Traffic heatmap of hybrid data/model parallelism

- Hybrid parallelism: data + model

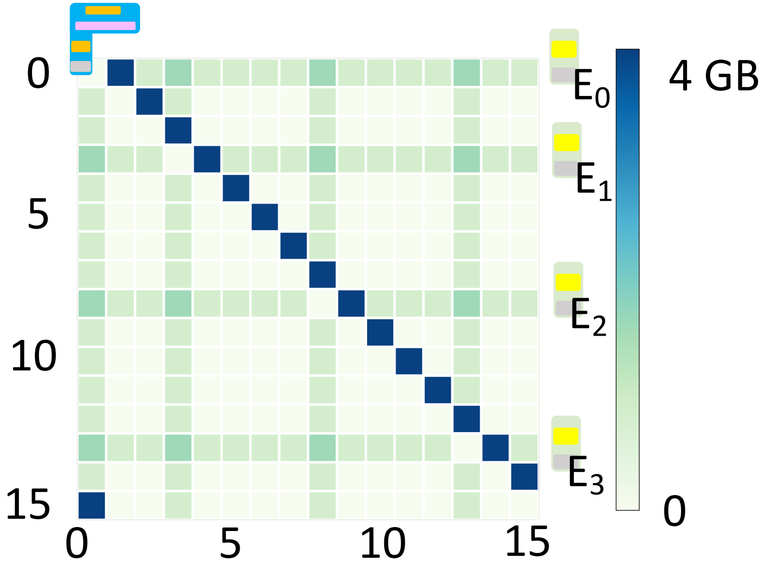


Deep Learning Recommendation Model (DLRM)

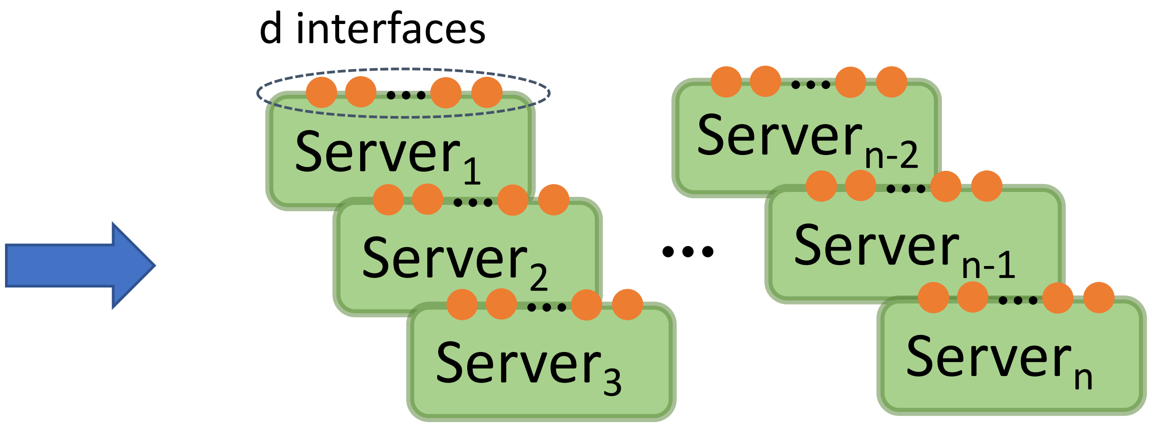


What is a good topology for a training job?

- Ideal solution: create a shard that exactly matches the traffic matrix
- Challenging:



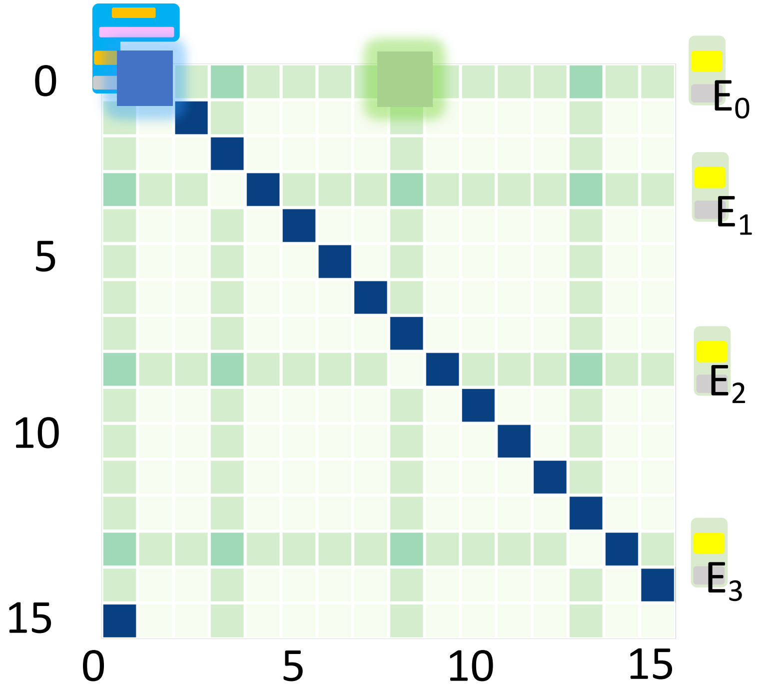
Non-uniform traffic distribution

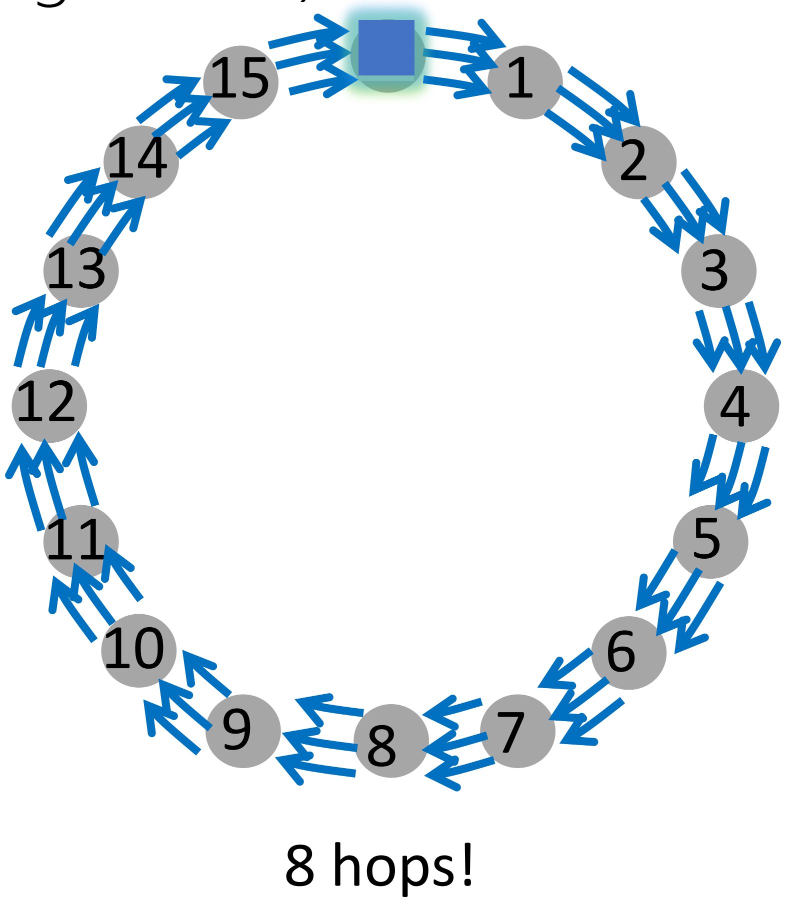


Limited degrees compared to total number of servers

Option 1: build a topology tailored for large flows

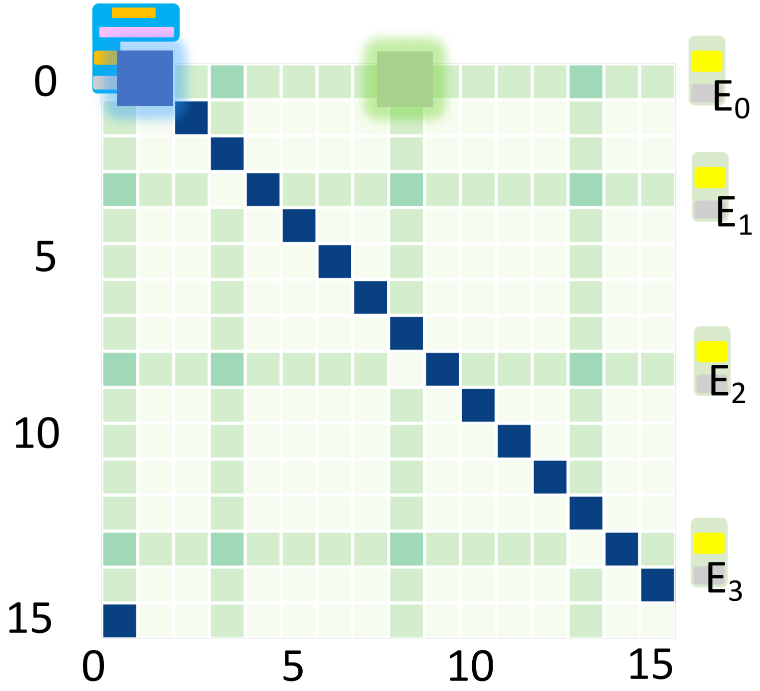
- Assume each server has three NICs (degree = 3)



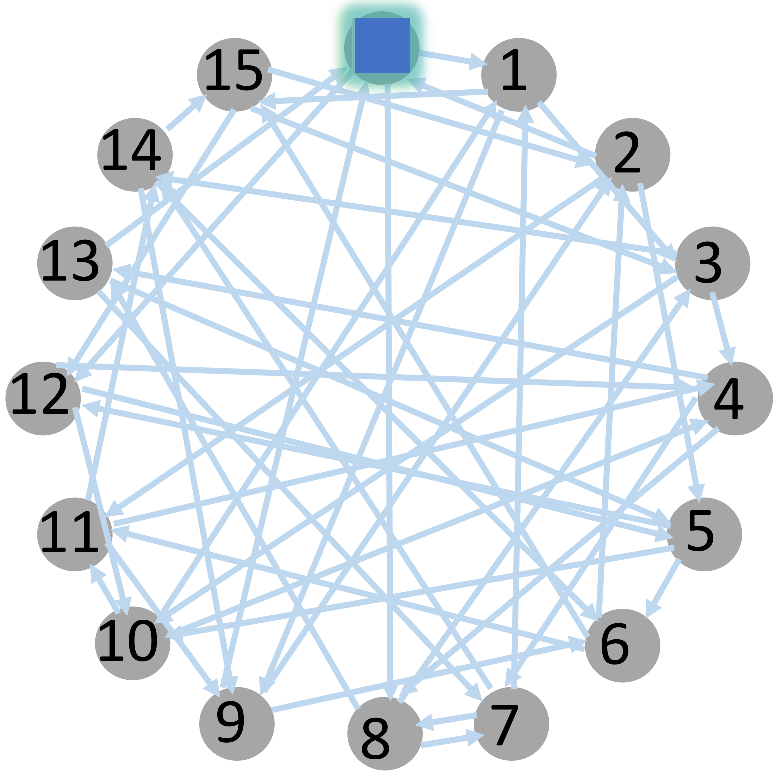


Option 2: build a topology tailored for short flows

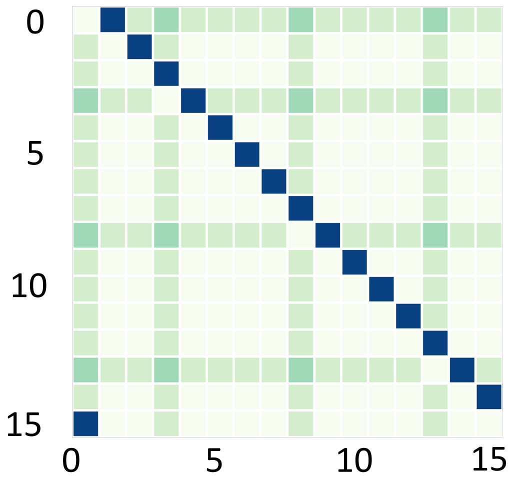
- Degree = 3

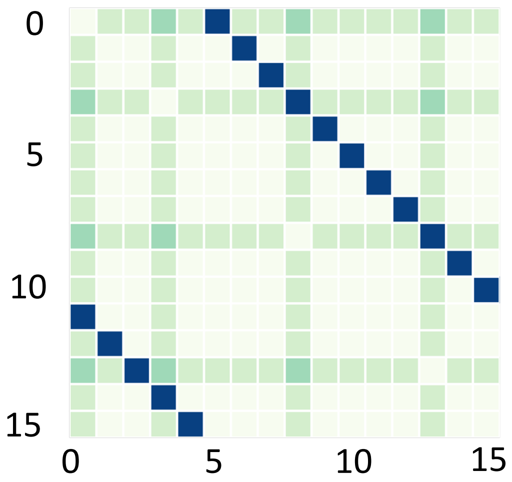


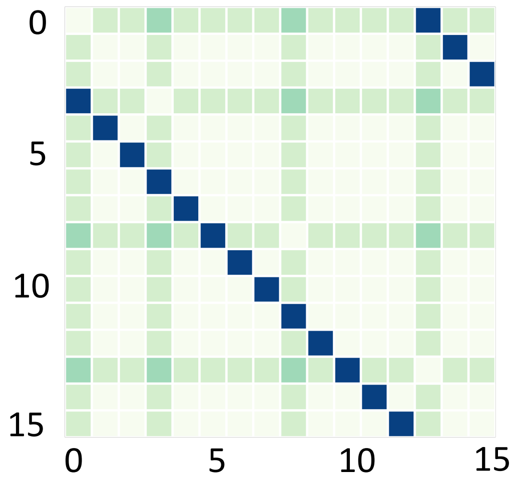
Low Bandwidth!

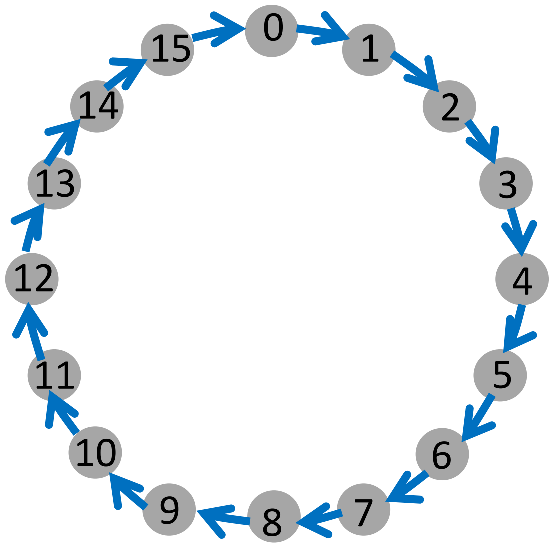


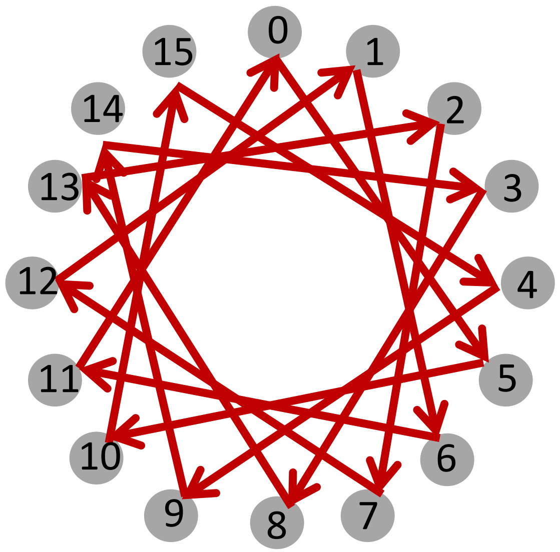
Key idea: mutate the traffic matrix

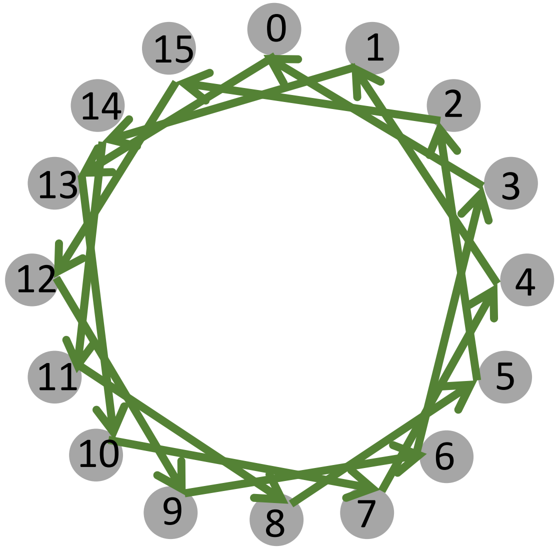






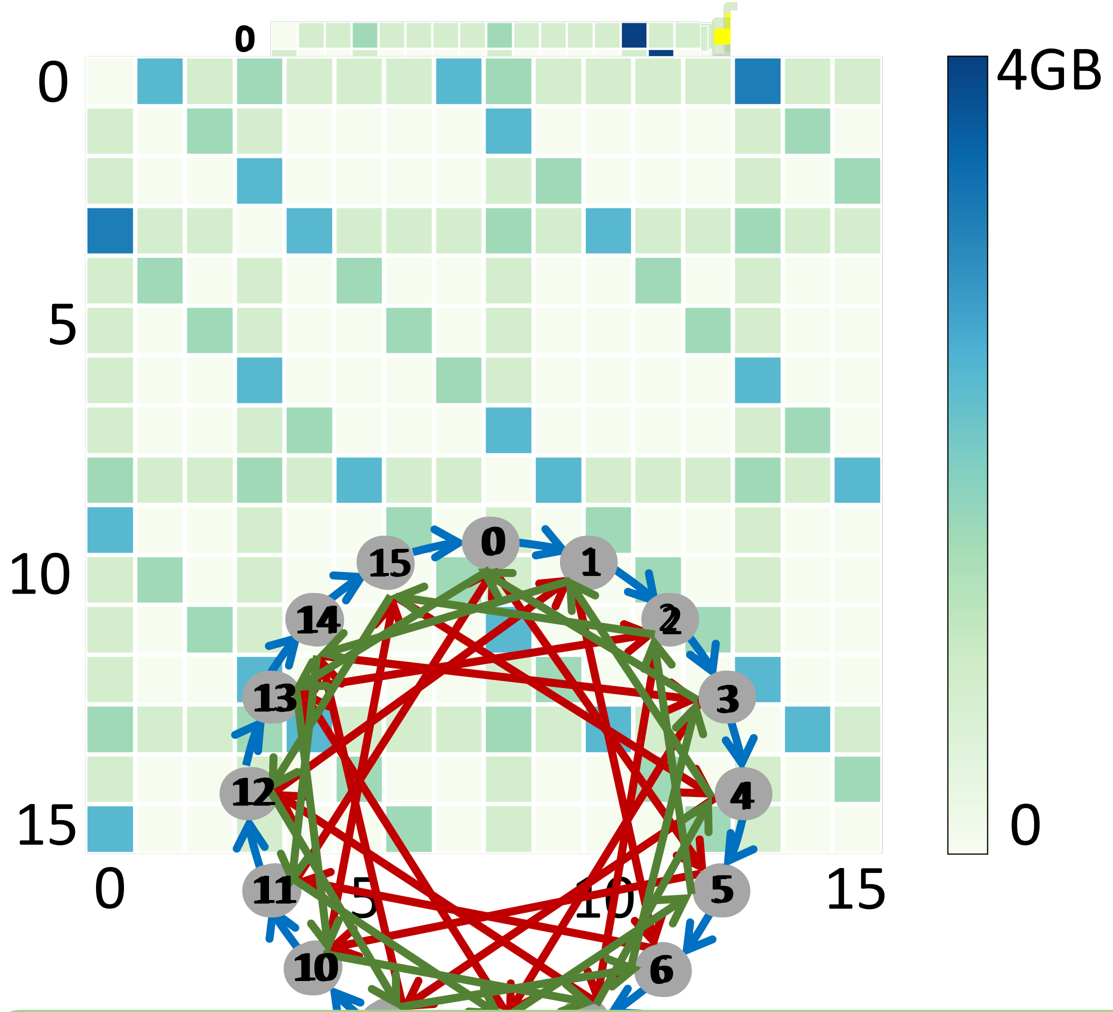


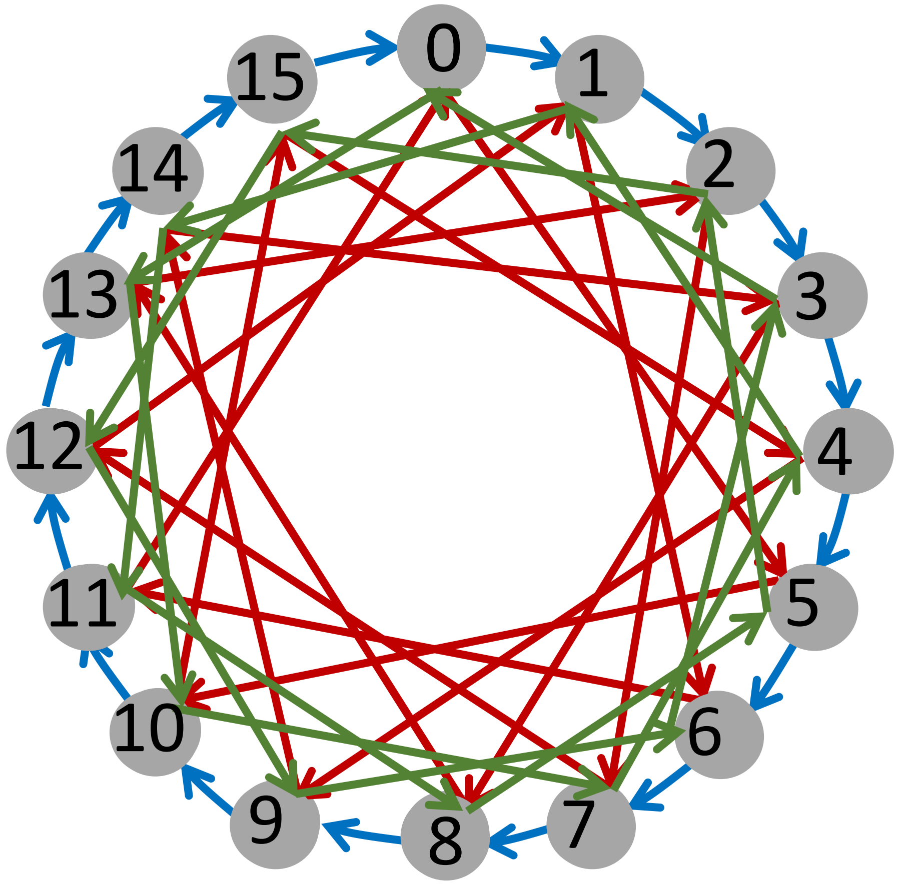




AllReduce transfers are mutable. Model-parallel transfers are not mutable.

Load-balance AllReduce traffic

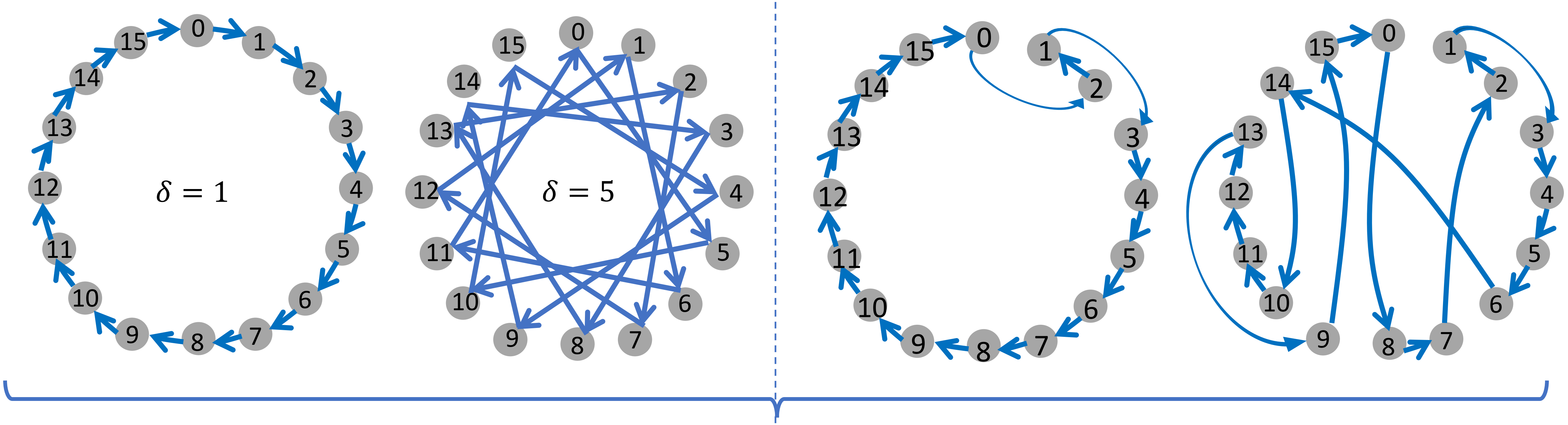




Theorem: our algorithm bounds the diameter of the topology to $O(d\,n^{1/d})$, where d is the degree of servers

Key technique: Regular permutations

- $\mathbf{n}$ total accelerator, each with degree $\mathbf{d}$



$\mathbf{O(n!)}$ permutations

Regular permutations

Irregular permutations

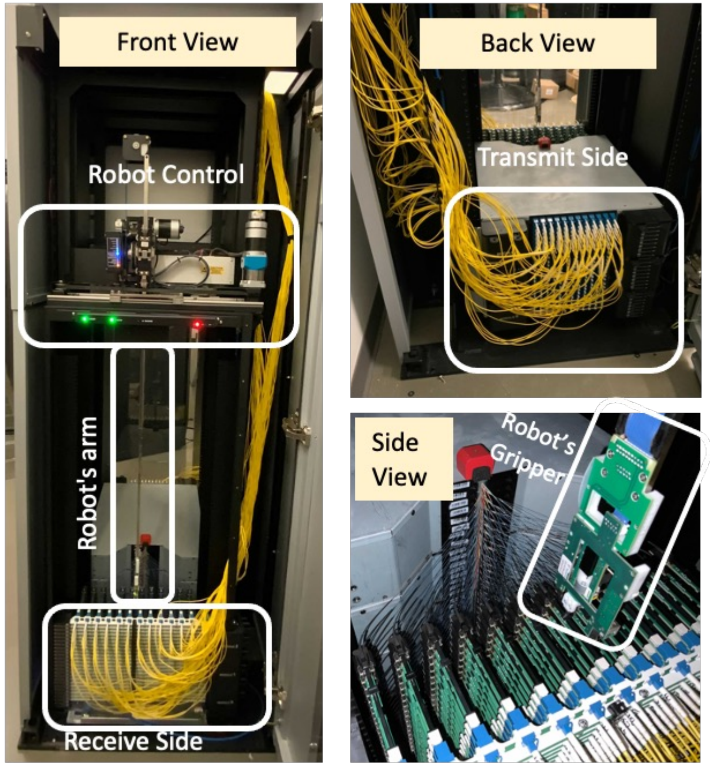
every server connects to another one with a fixed distance δ

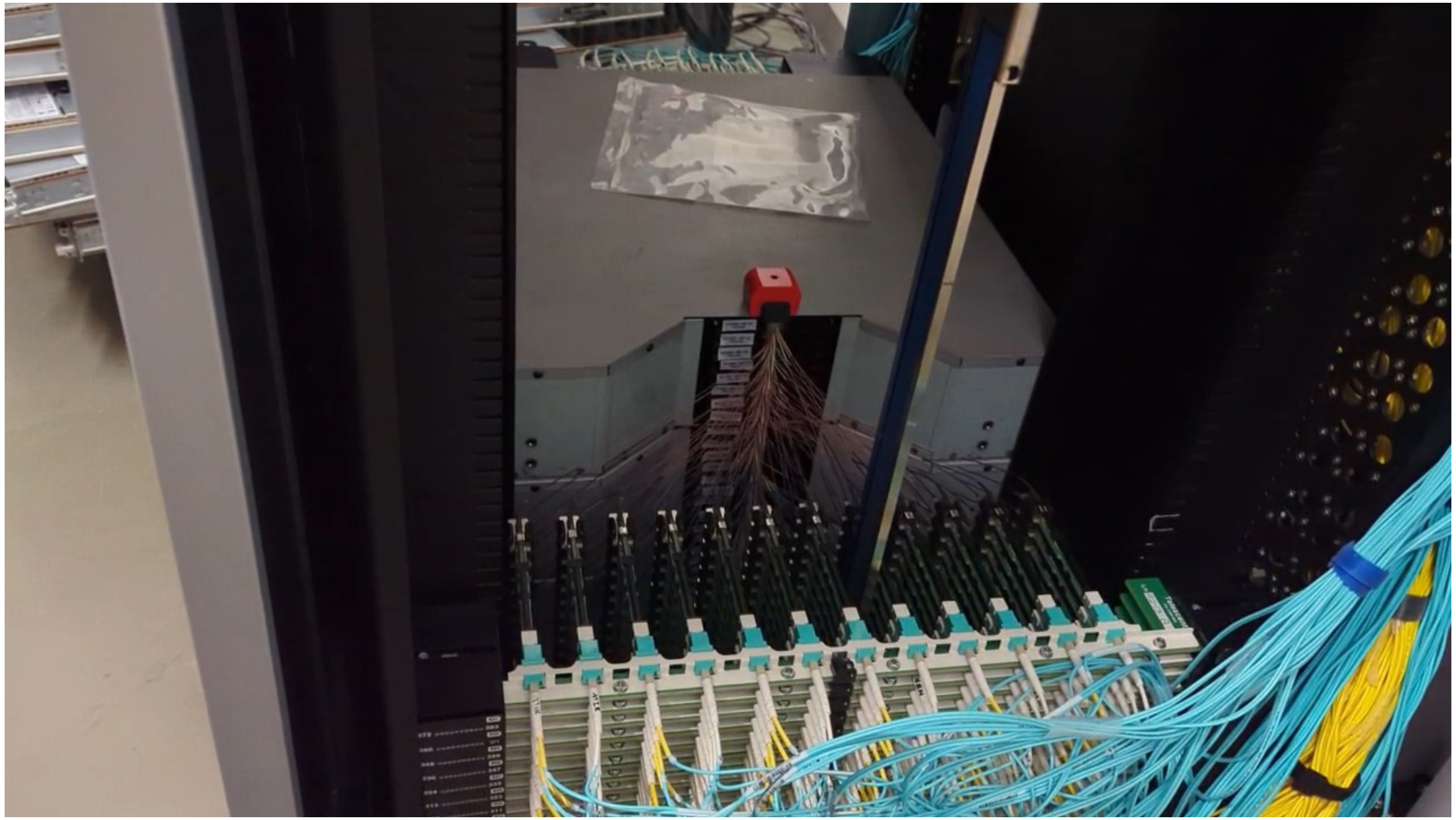
Key technique: Regular permutations

- $\mathbf{n}$ total accelerator, each with degree $\mathbf{d}$
- The possible set of δ are the positive integers less than n, such that $\gcd(\delta, n) = 1$ **-> $\boldsymbol{O}(\boldsymbol{n})$ search space!**
- Among all possible δ distances, choose a set of them within the degree to minimize the cluster diameter
- This technique works for other AllReduce algorithms as well

Testbed and simulations

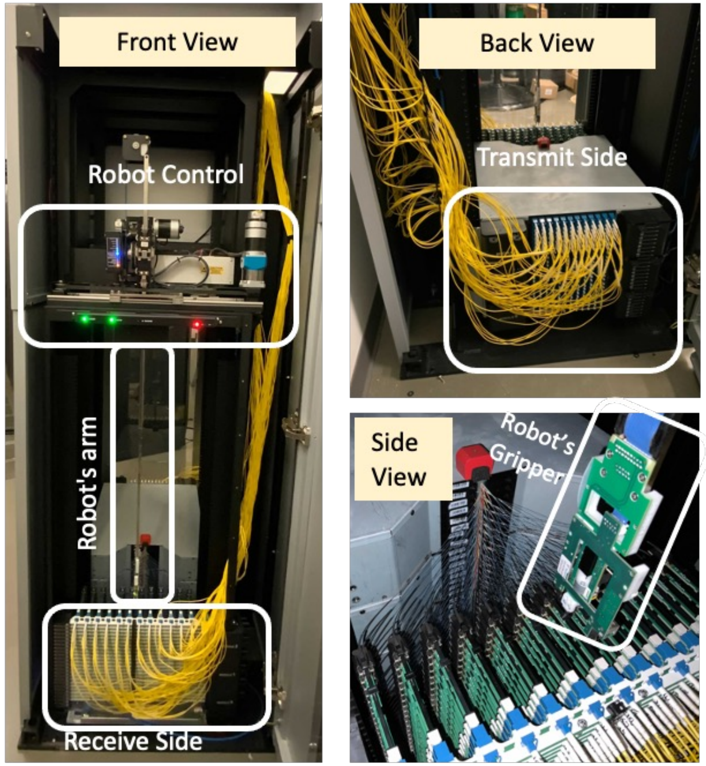
- Implemented in NCCL (code: http://topoopt.csail.mit.edu/)
- A 100 Gbps prototype with Nvidia A100 GPUs

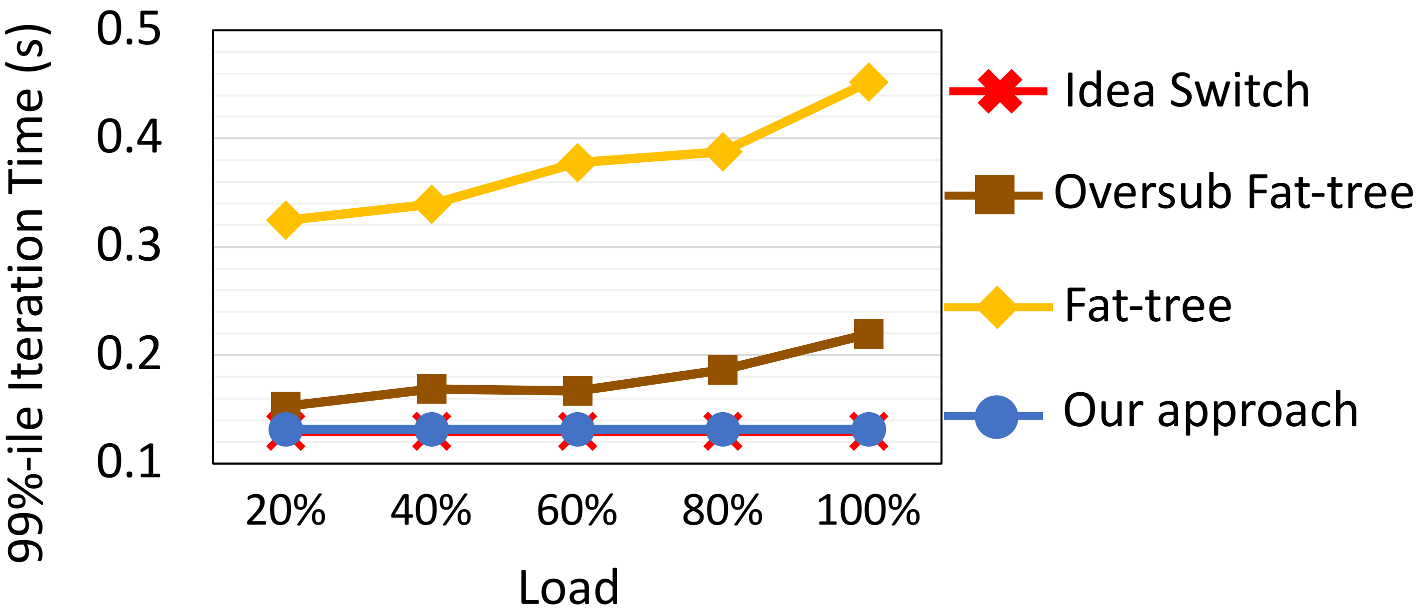




Testbed and simulations

- Implemented in NCCL (code: http://topoopt.csail.mit.edu/)
- A 100 Gbps prototype with Nvidia A100 GPUs

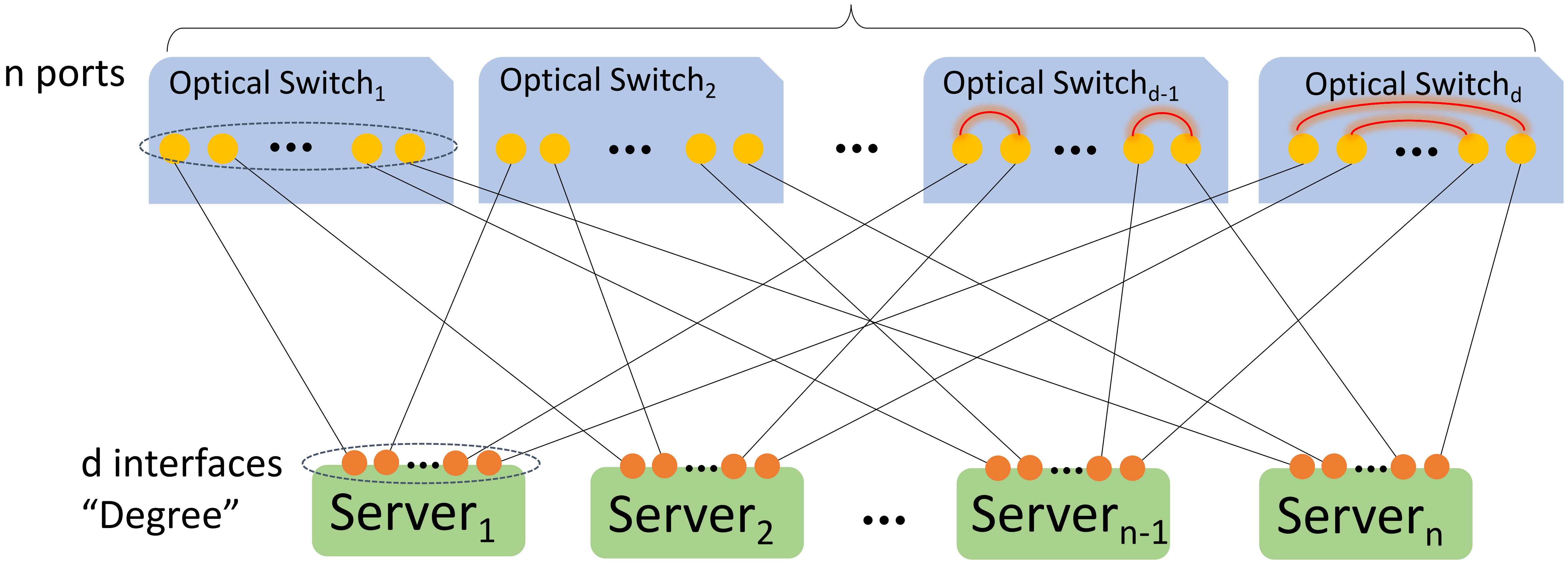




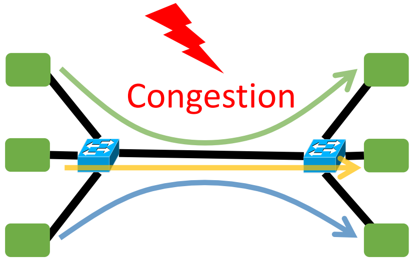
TopoOpt accelerates training time by 3.4x compared to Fat-trees.

Direct-connect topologies & Netdev community

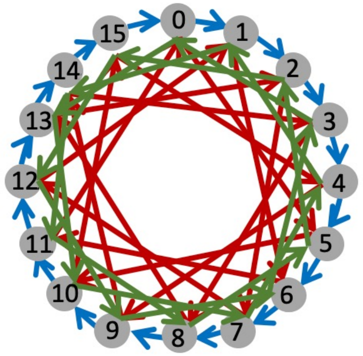
- End-host networking stack is critical for routing, load-balancing, communication collective



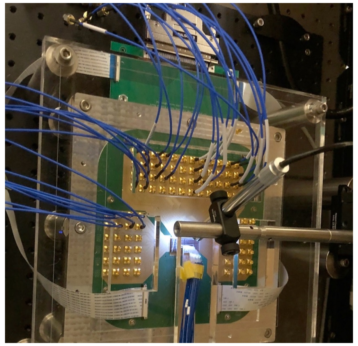
Talk outline: three key lessons



Fair congestion control is sometimes inefficient [HotNets'22, NSDI'24].



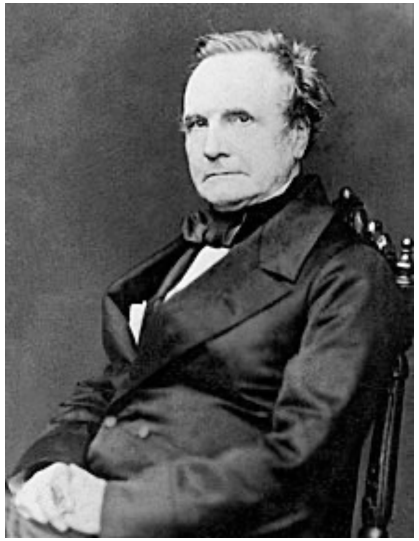
Reconfigurable networks for ML training [SIGCOMM'21, NSDI'23].



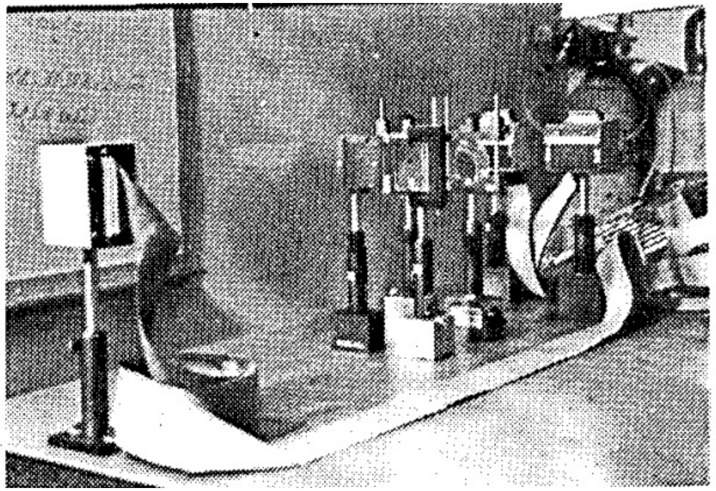
Analog compute for ML inference [SIGCOMM'23, Science'22, OFC'22].

What is photonic computation?

- Use light waves to perform computation in the analog domain
- Computers were born analog



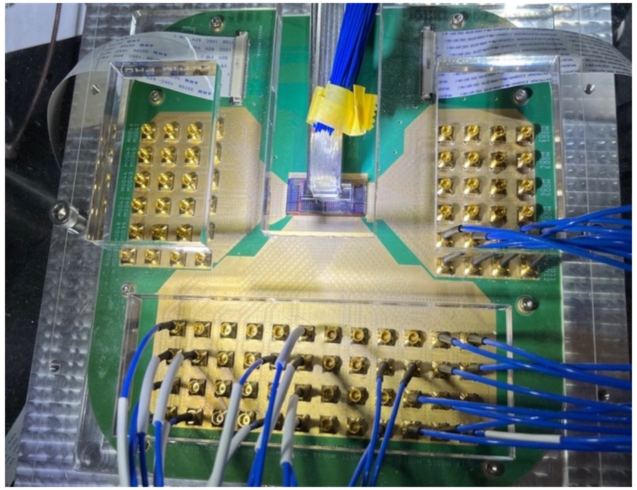
Charles Babbage conceptualized computers in 1840s as analog devices



Optical AI accelerator
Farhat et al., Opt. Lett. 1985

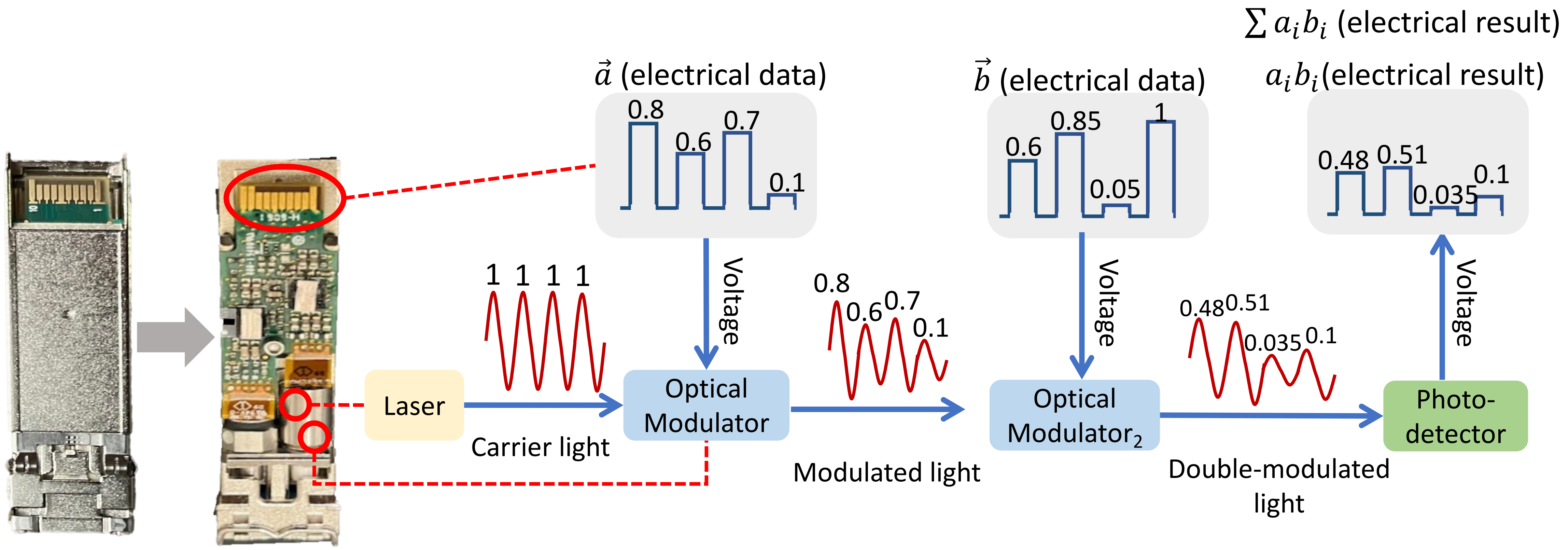
Photonics can revolutionize computation

- Compute at 100 GHz
- 40 atto Joules (10^{-18}) per operation [Science'22]



But photonic computing has never gained practical traction!

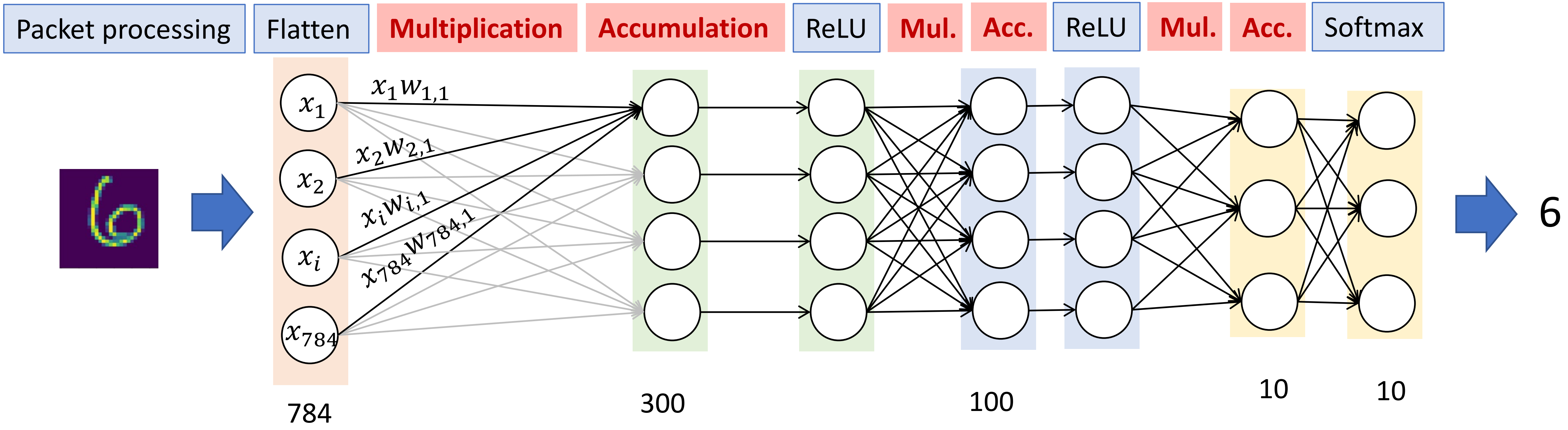
Photonic multiplication: modulating light intensities



- Commodity modulators operate at 15 GHz frequency (100 GHz in the lab)

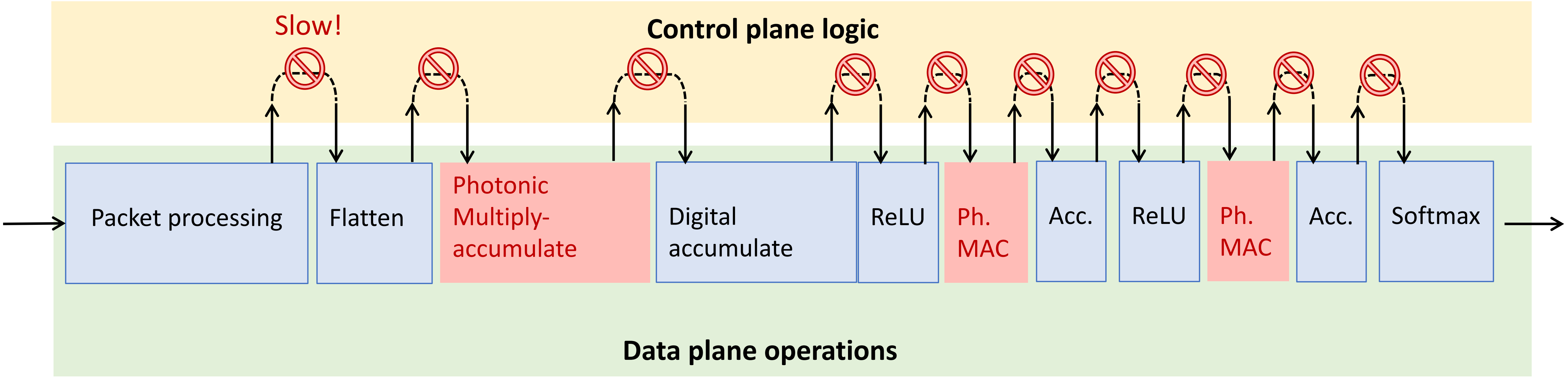
Optical modulators and photodetectors are passive devices.

Challenge: optical devices are passive



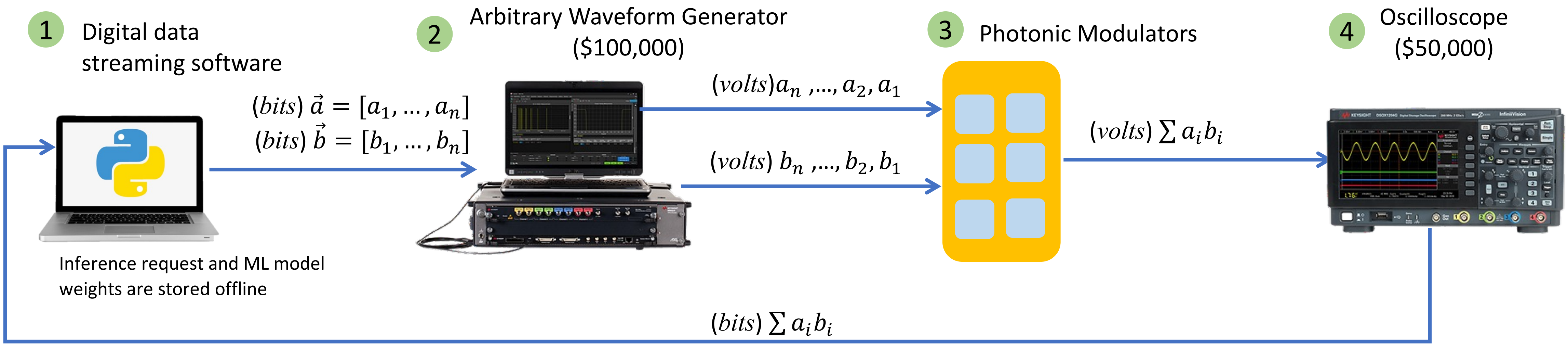
- DNN DAGs involve a sequence of complex operations
- A control logic is needed to coordinate the operations across electronics and photonics

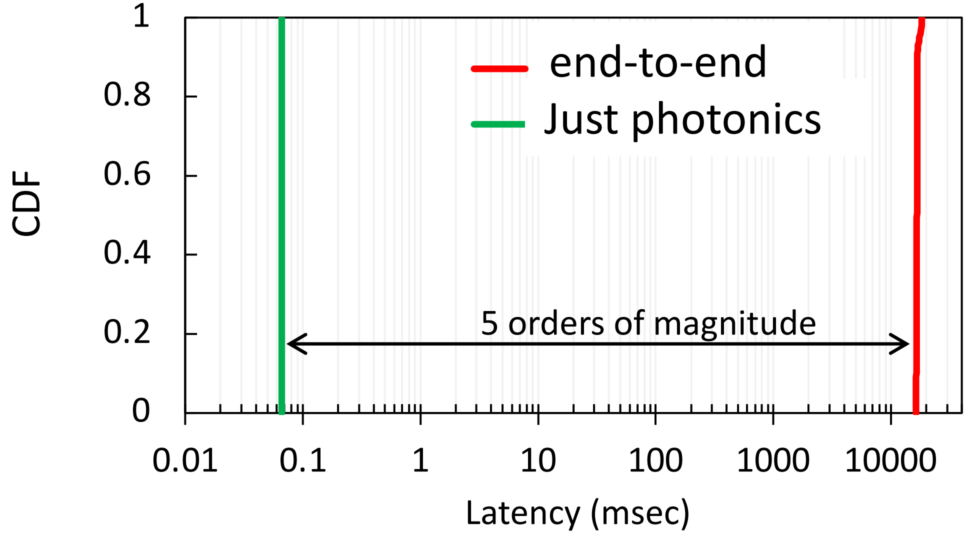
Implication: Stop-and-go data movement between digital & photonics



- The control plane logic is deeply coupled with the data plane operations
- Slows down the critical data plane latency, increases energy consumption

The Achilles' heel of photonic computing systems

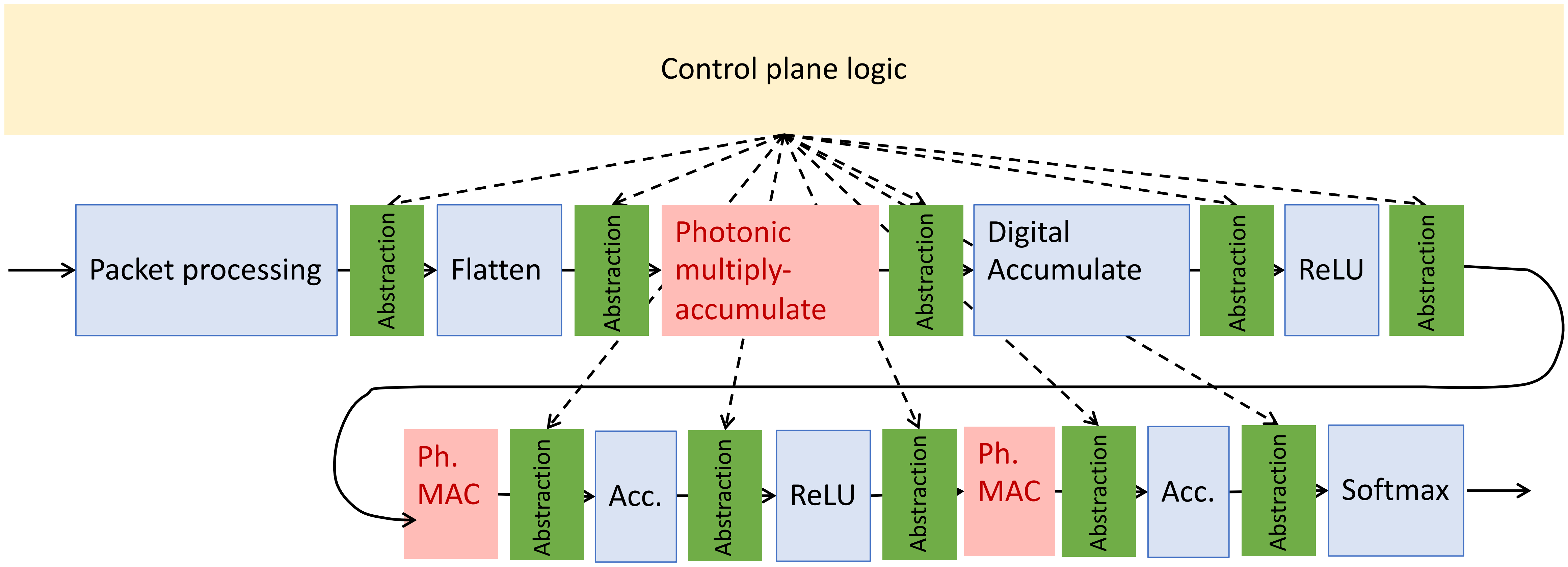




Component	Power (Watts)
Digital computer	250
Arbitrary waveform generator	220
Photonic modulators	0.001
Digitizer	1350

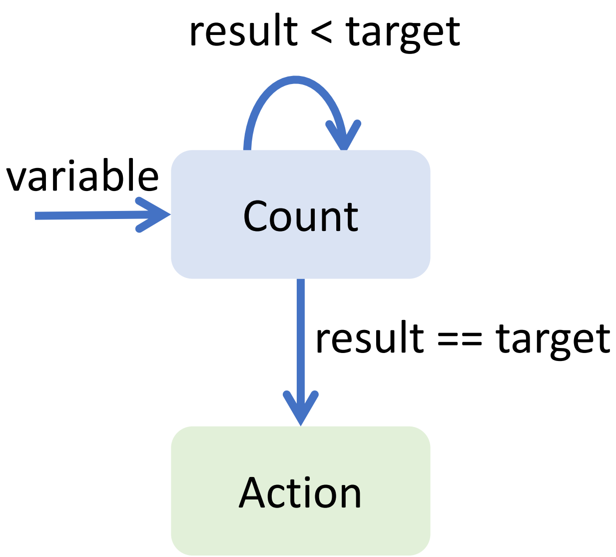
Our solution: co-design photonics and digital systems together

Key innovation: a programming abstraction for photonic computing systems



Lightning: A Reconfigurable Photonic-Electronic SmartNIC for Fast and Energy-Efficient Inference
Z. Zhong, M. Yang, J. Lang, C. Williams, L. Kronman, A. Sludds, H. Esfahanizadeh, D. Englund, M. Ghobadi, SIGCOMM 2023

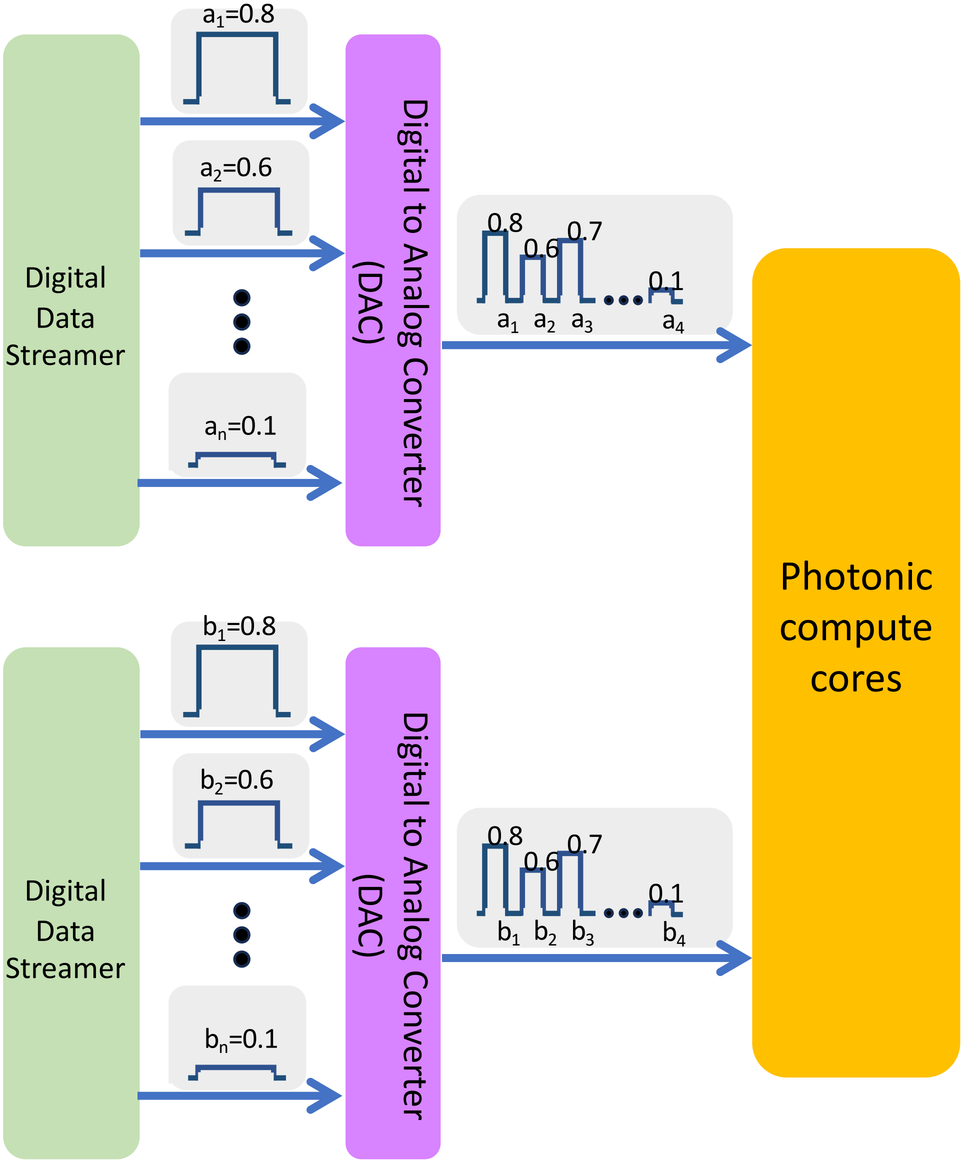
The control abstraction: reconfigurable count-action




```
module datapath_module {
        counts: {
                // variables to be counted
        }
        targets: {
                // a set of target results
        }
        actions: {
                // actions to be triggered
                // when the result is
                // equivalent to the target
        }
};
```

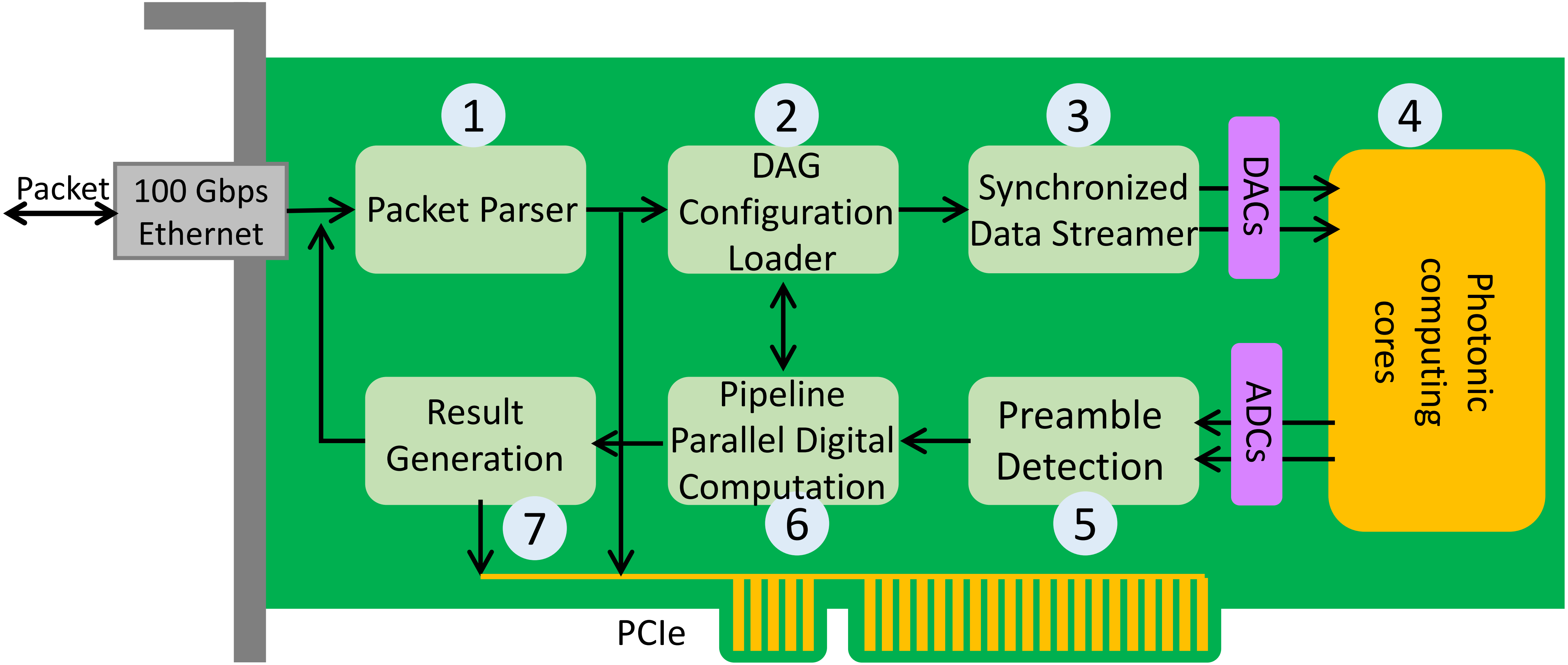
High-level idea: Trigger an action whenever the count result reaches the target.

Example: synchronized data streamer




```
module synchronized_data_streamer {
        counts: {
                // count the sum of valid DAC flags
                sum DAC[i].valid (i = 1 to num_DACs)
        }
        targets: {
                // trigger when the sum equals the number of DACs
                num_DACs
        }
        actions: {
                // stream DACs' data into photonic cores
                stream DAC[i].data (i = 1 to num_DACs)
        }
};
```

Putting it all together: Lightning SmartNIC

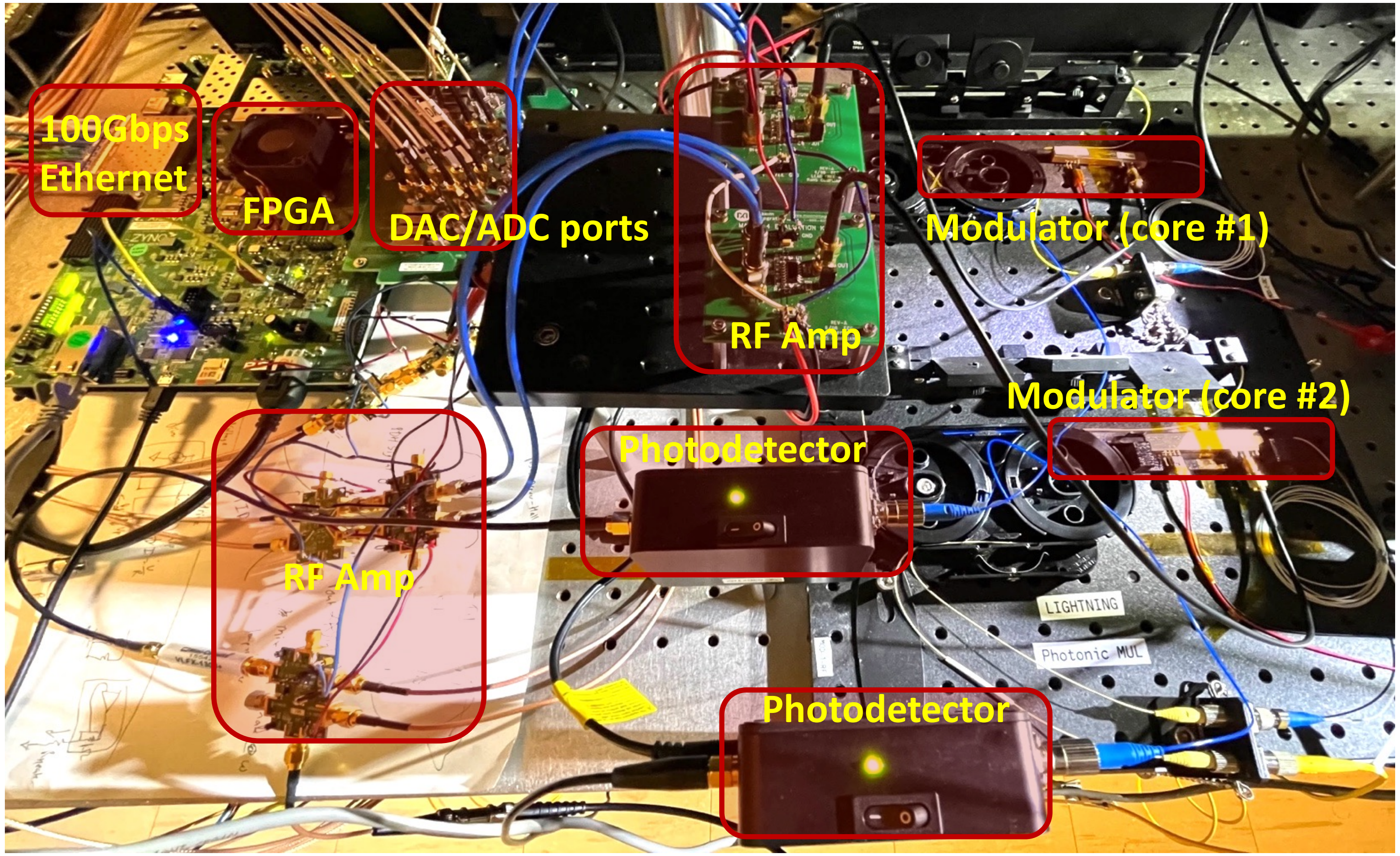


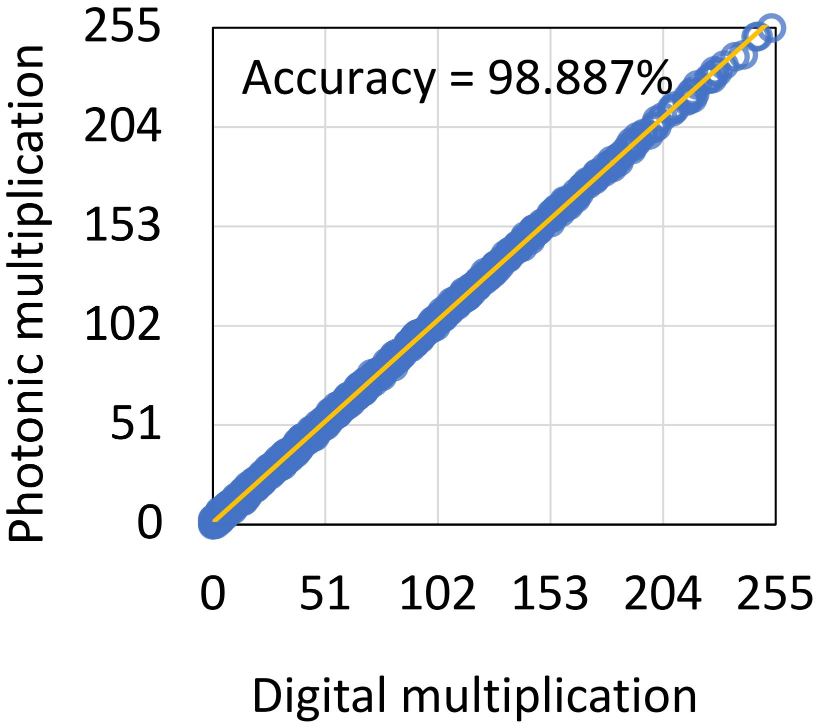
Plug-and-play kit for developers

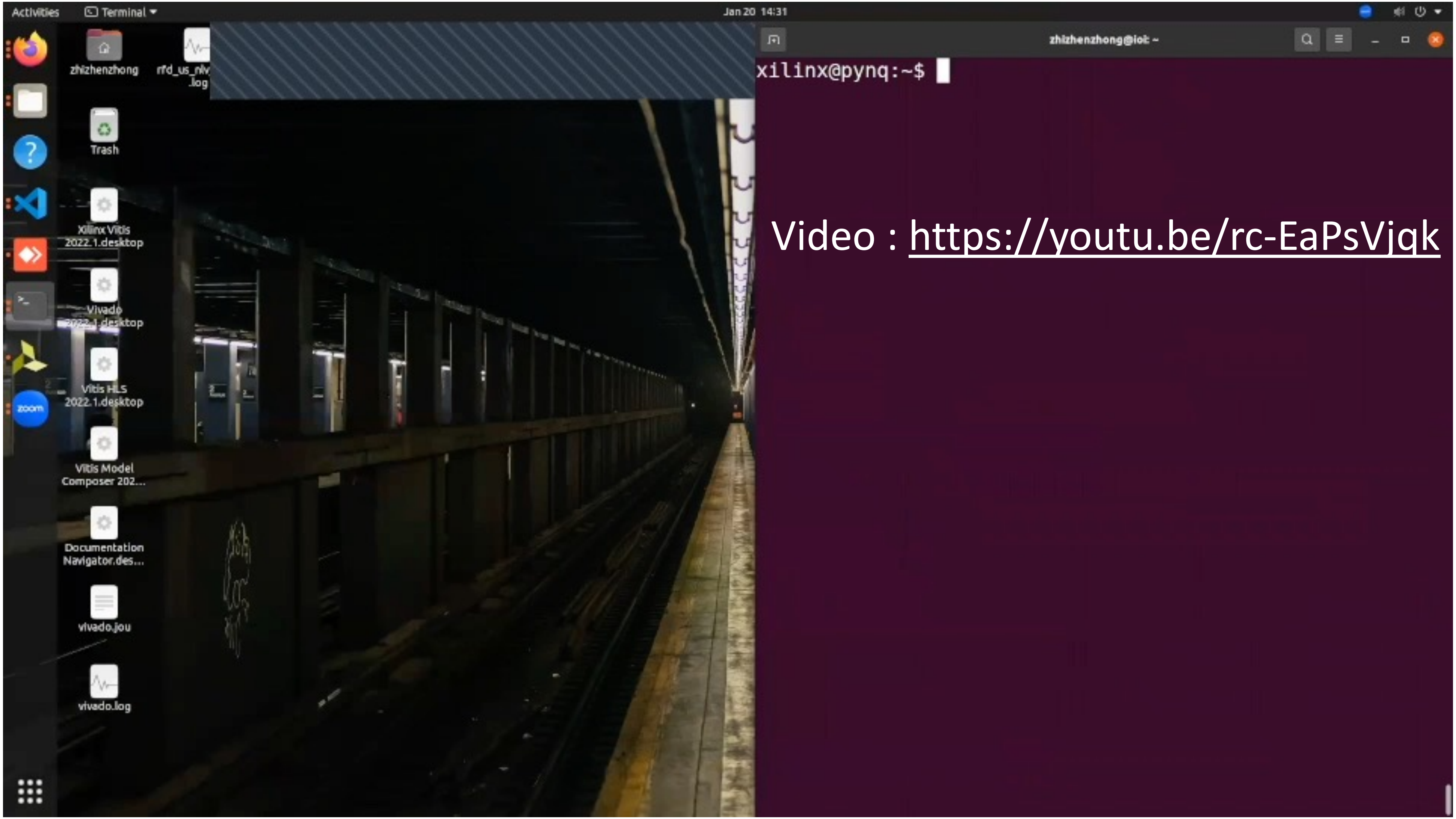
Open-source (https://lightning.mit.edu/)

```
from lightning import LightningControl, LightningConfig, LightningSignalProcessing
from qick import QickSoc

import numpy as np

# instantiate QICK library and use it to program the FPGA
soc = QickSoc()
soccfg = soc

# the lightning config that reconfigures the input values
LightningConfig["dac_0"] = 100  # value from 0 to 255
LightningConfig["dac_1"] = 50  # value from 0 to 255

# run lightning photonic computing
lightning_runtime = LightningControl(soccfg, LightningConfig)
result_waveform = lightning_runtime.acquire_decimated(soc)

# check the raw waveform detected on the ADC
lightning_sp = LightningSignalProcessing()
lightning_sp.plot_waveform(result_waveform)

# show 8-bit fix point multiplication result from 0 to 255
multiplication_result = lightning_sp.decode_adc_result
print(multiplication_result)
```

World's highest-frequency (4GHz) photonic ML inference



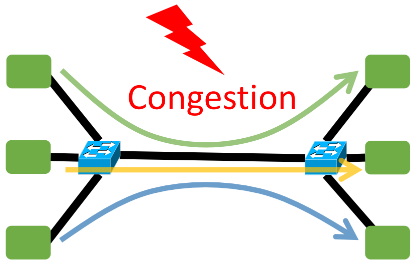




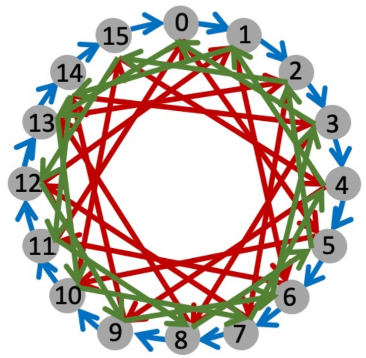
Video : https://youtu.be/rc-EaPsVjqk

Final remarks

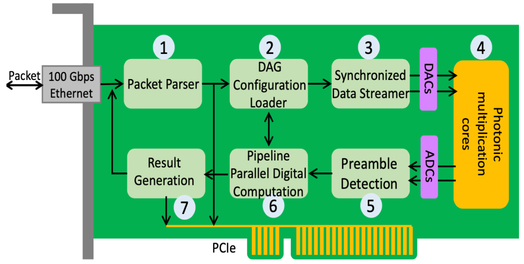
- Innovations in networking come from applications
- The network stack is vital to application performance in distributed settings
- Many opportunities for the Netdev community to impact ML networking!



Congestion control



Topology optimization



Datapath engineering

ghobadi@mit.edu

Thanks to my students, collaborators, and mentors

Sudarsanan Rajasekaran
Weiyang Wang
Mingran Yang
Benoit Pit-Claudel
Moein Khazraee
Homa Esfahanizadeh
Zhizhen Zhong
Christian Williams
Liam Kronman
Jay Lang
Alexander Sludds
Mehrdad Khani
Zhihao Jia
Anthony Kewitsch
Ajay Brahmakshatriya

Hari Balakrishnan
Dina Katabai
Mohammad Alizadeh
Dirk Englund
Yashar Ganjali
Muriel Medard
Saman Amarasinghe
Keren Bergman
Madeleine Glick
Benjamin Klenk
Ziyi Zhu
Eiman Ebrahimi
Hadi Esmaeilzadeh
Aditya Akella
Gautam Kumar

Amin Vahdat
Arvind Krishnamurthy
Srini Devadas
Victor Bahl
Ishai Menache
Jennifer Rexford
Albert Greenberg
Mark Filer
Ratul Mahajan
Adam Belay
Adam Chlipala
Ryan Hamerly
Liane Bernstein
Ying Zhang
Dheevatsa Mudigere
And many others…