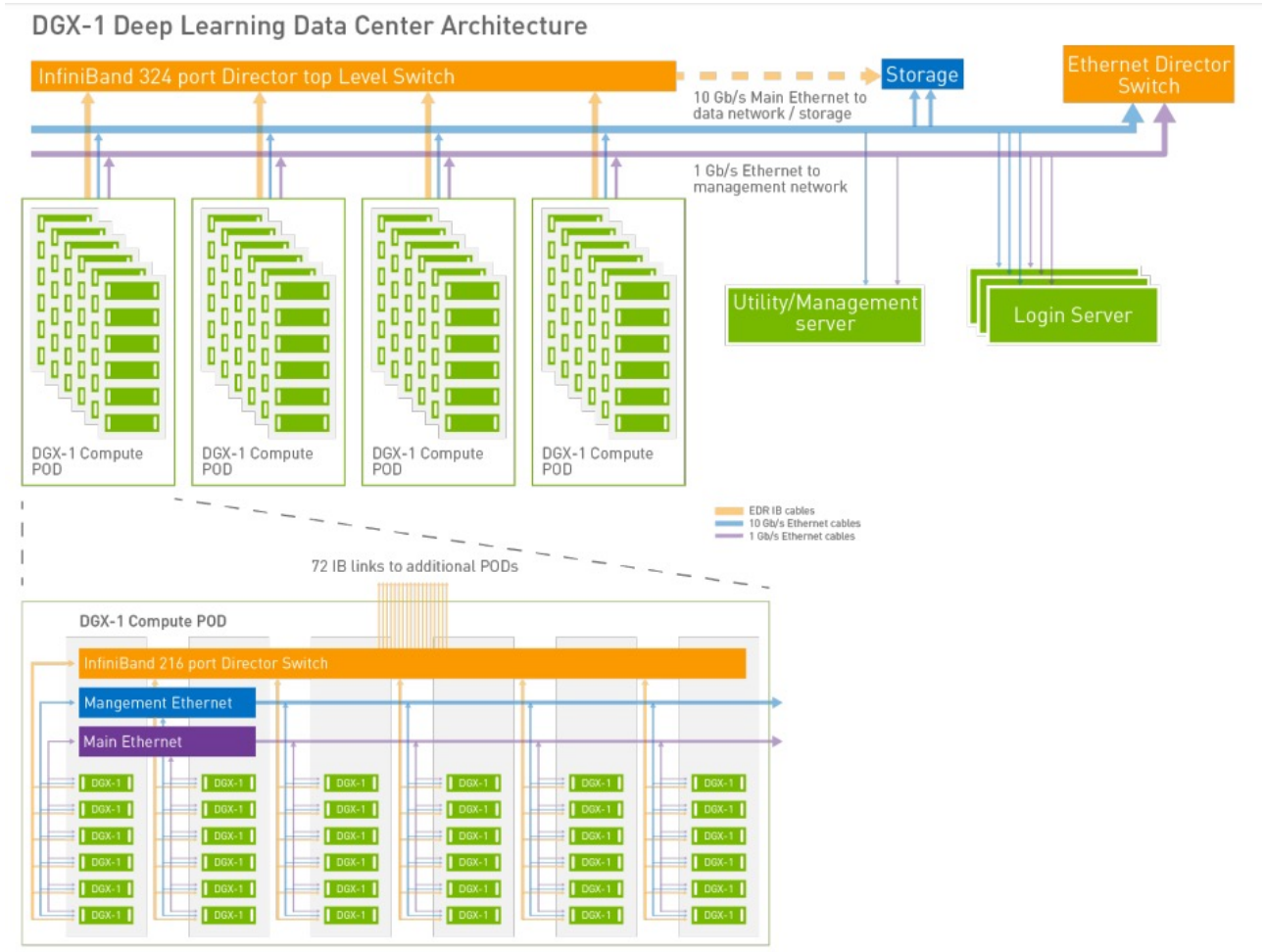
Challenges and Optimizations in Multi-accelerator Systems

Kiran Ranganath

Data centers and Supercomputers

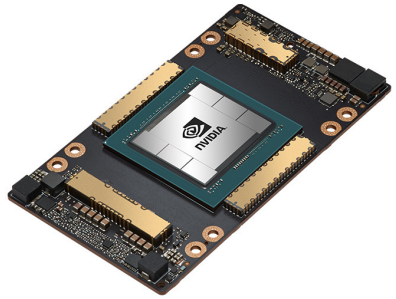


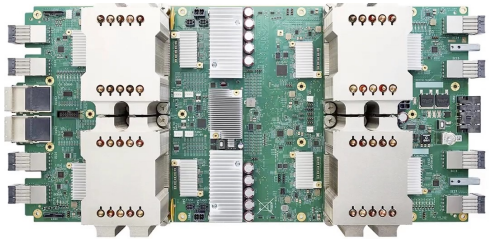
What does each server have?

- **CPUs**
  - Boot Operating System.
  - Heart of the System.

What does each server have?

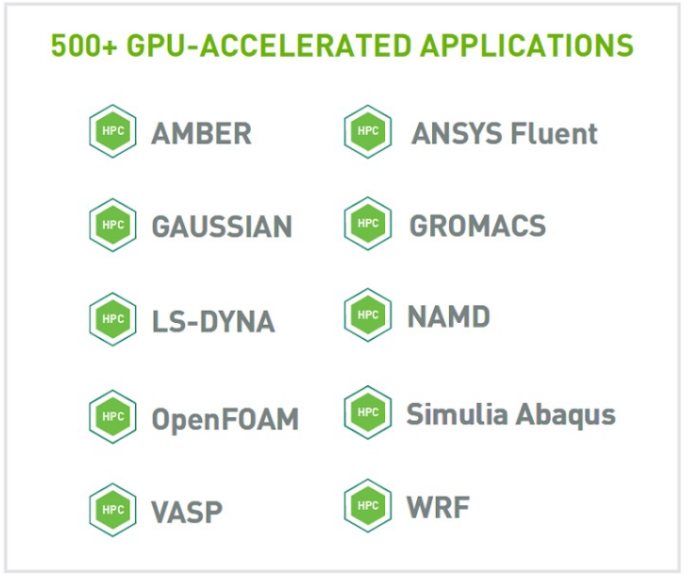
- CPUs
- **Accelerators** – GPUs, DPUs, IPUs, TPUs, ASICs, etc.





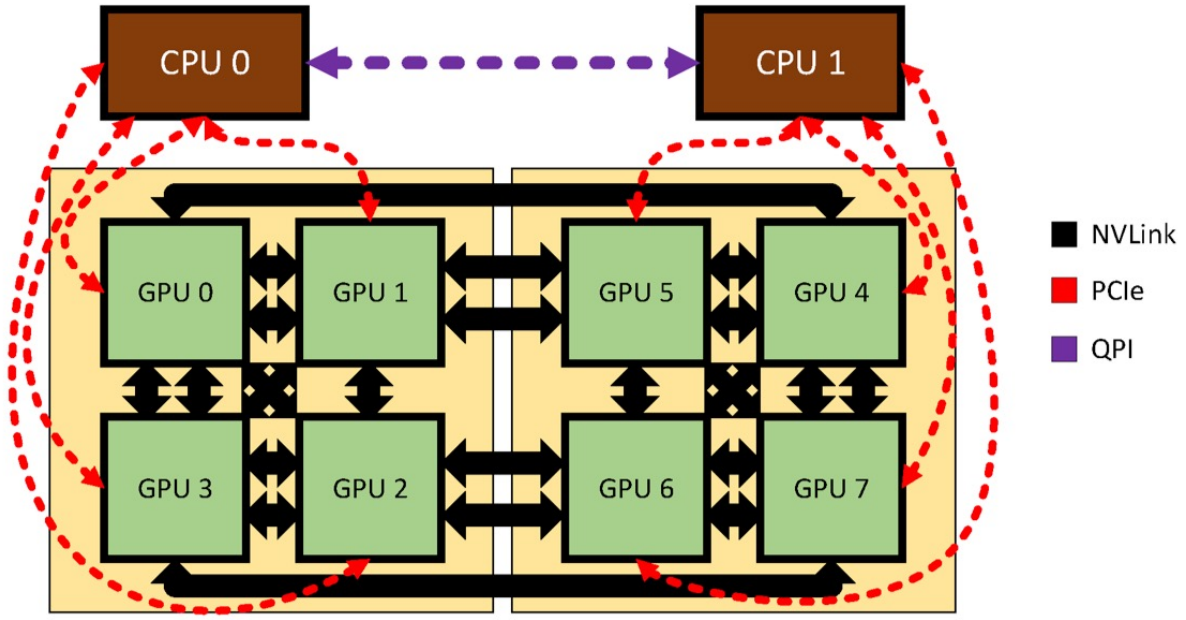
What does each server have?

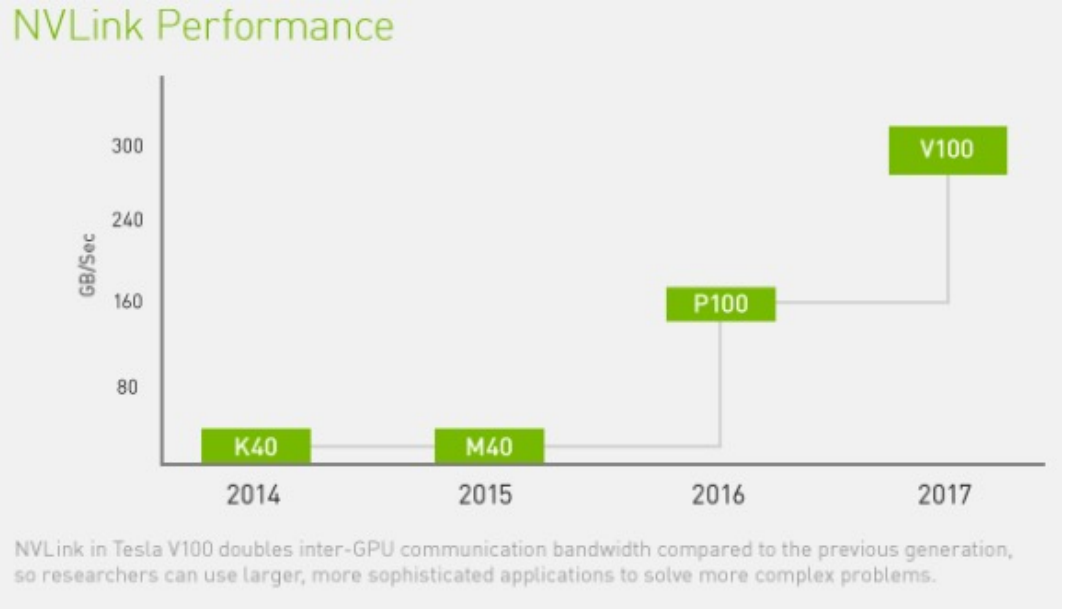
- CPUs
- Accelerators – GPUs, DPUs, IPUs, TPUs, GraphCore, etc.
- **Interconnects** – PCIe, NVLink, Nvswitch, AMD Inifinity, etc.
  - More about this later.
- **Why accelerators?**
  - Thermodynamics, Computational Fluid Dynamics, Machine Learning.



Nvidia DGX System

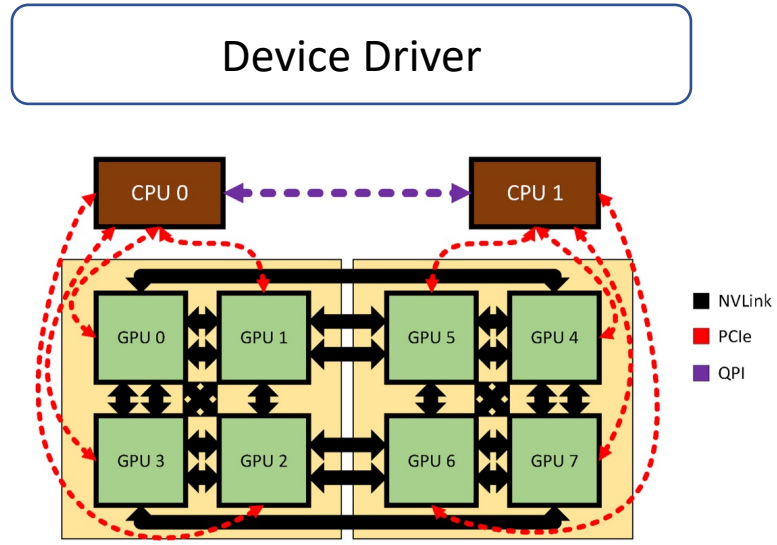
- 8 V100 GPUs
- NVLinks
  - Single – 25 GBps
  - Double – 50 GBps
  - PCIe – 12 GBps
- Each GPU has 6 NVLinks





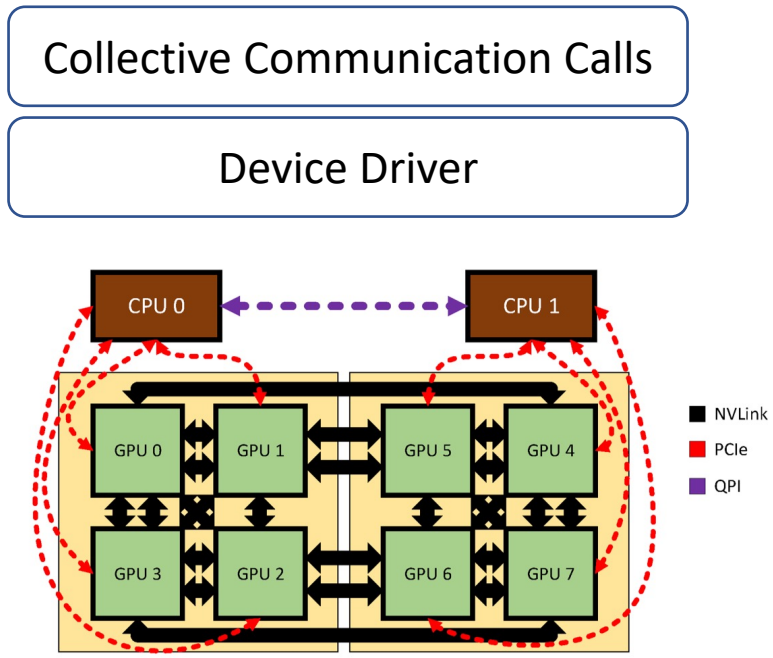
Optimization Spaces

- **Device driver** - software used to control and drive the hardware.



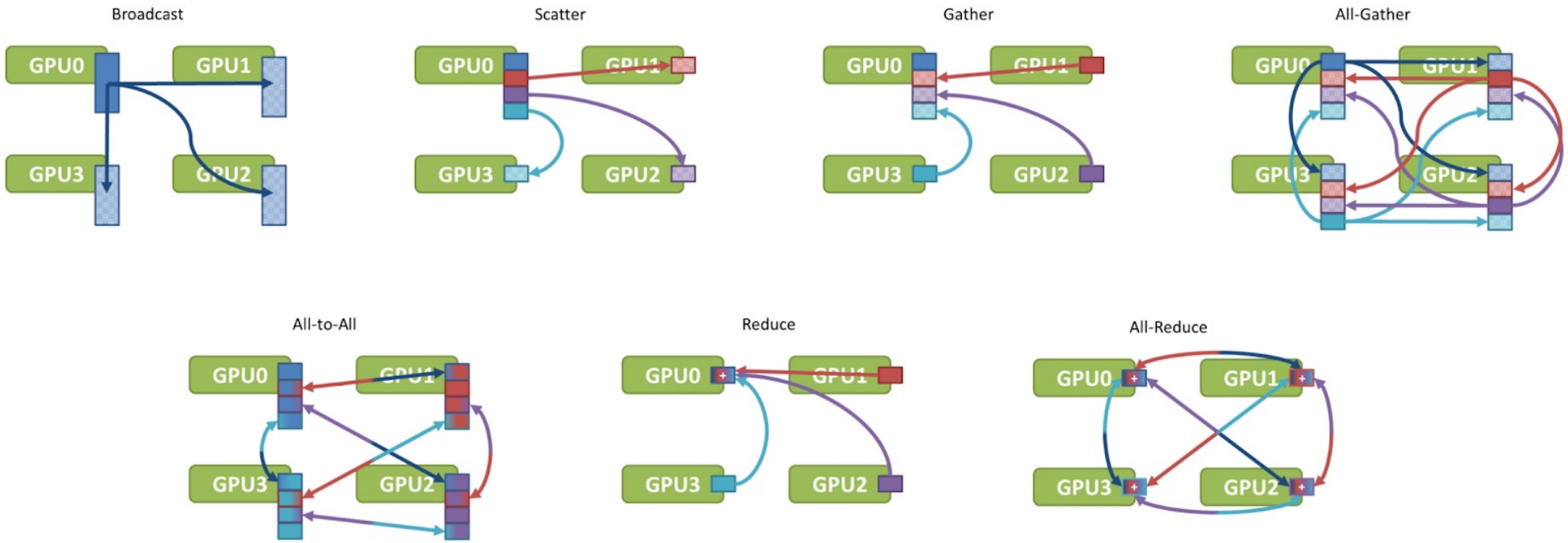
Optimization Spaces

- Device driver - software to control and drive the hardware.
- **Message Passing Interface/Collective Communication Libraries** – used to implement synchronization primitives



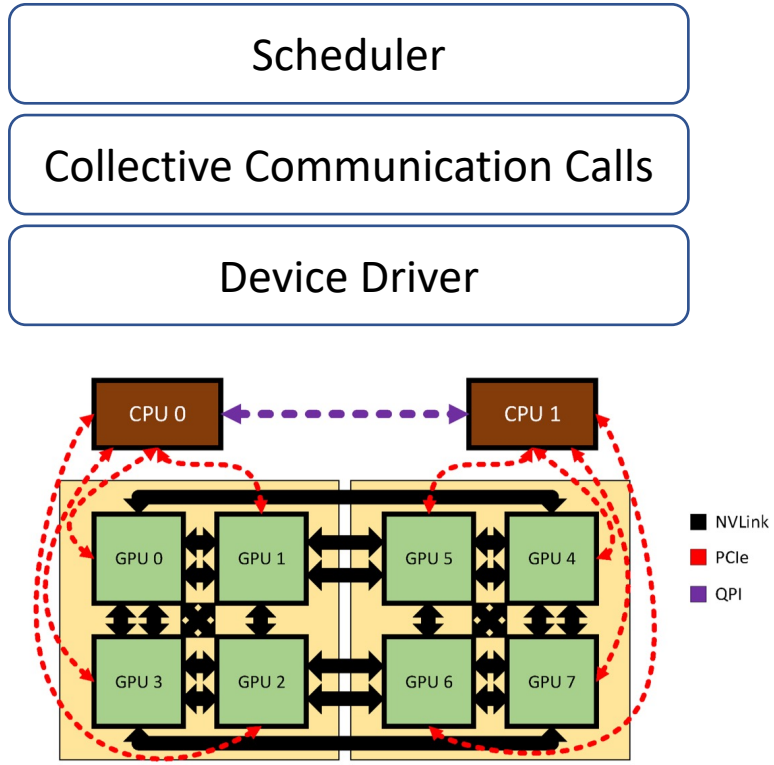
NCCL

- Nvidia Collective Communication Library



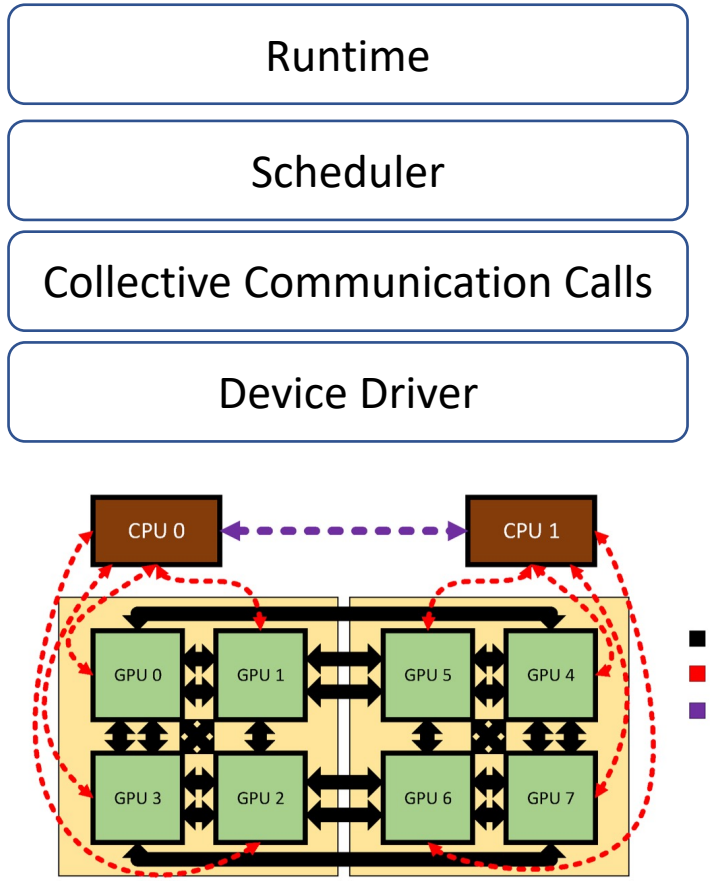
Optimization Spaces

- Device driver - software to control and drive the hardware.
- Message Passing Interface/Collective Communication Libraries – used to implement synchronization primitives
- **Scheduler** – Policy to map jobs to hardware based on constraints and requirements



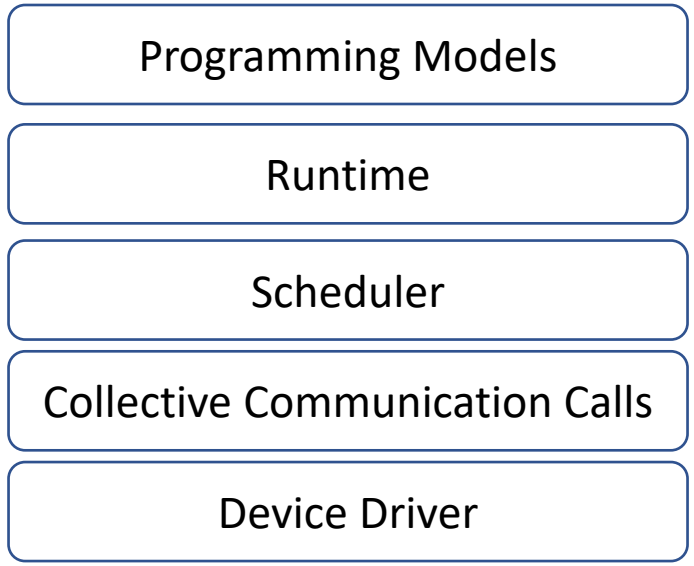
Optimization Spaces

- Device driver - software to control and drive the hardware.
- Message Passing Interface/Collective Communication Libraries – used to implement synchronization primitives
- Scheduler – Policy to map jobs to hardware based on constraints and requirements
- **Runtime** – Software to handle execution of jobs. Ex: CUDA Runtime, KOKKOS, DAGEE, ATMI, etc.



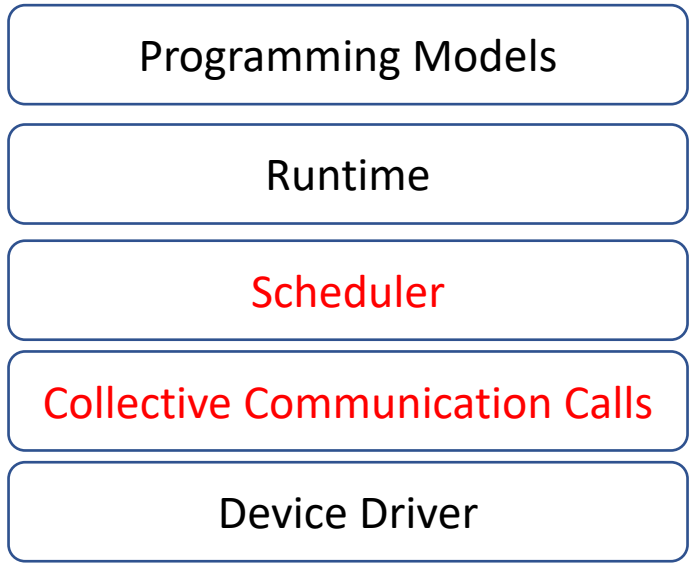
Optimization Spaces

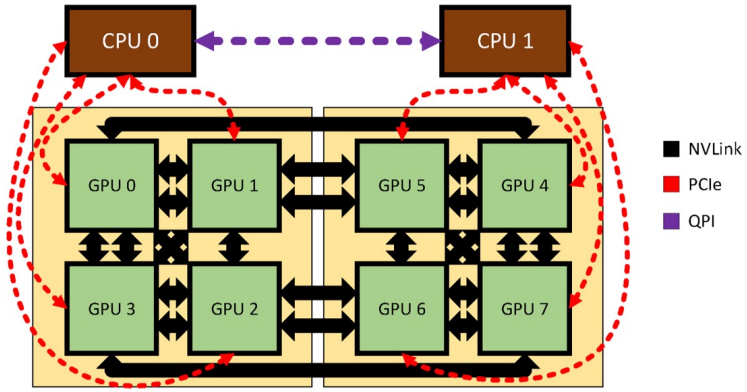
- Device driver - software to control and drive the hardware.
- Message Passing Interface/Collective Communication Libraries – used to implement synchronization primitives
- Scheduler – Policy to map jobs to hardware based on constraints and requirements
- Runtime – Software to handle execution of jobs. Ex: CUDA Runtime, KOKKOS, DAGEE, ATMI, etc.
- **Programming Models** – Implementation model to represent instructions. Ex: CUDA, HIP etc.



Optimization Spaces

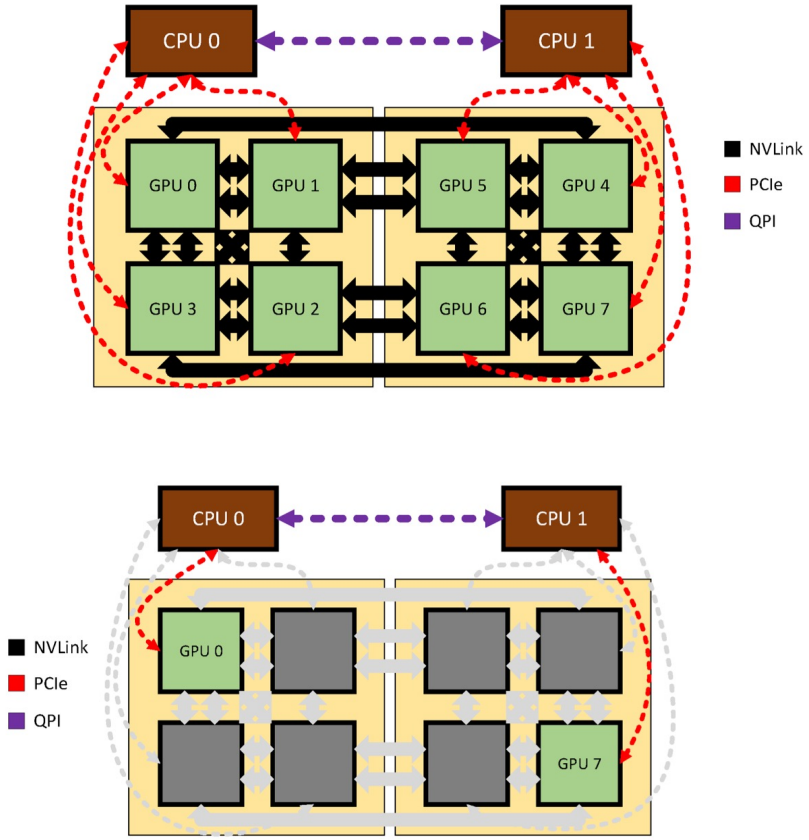
- In this presentation, we shall motivate problems and discuss solutions in,
  - Scheduler
  - Collective Communication Calls





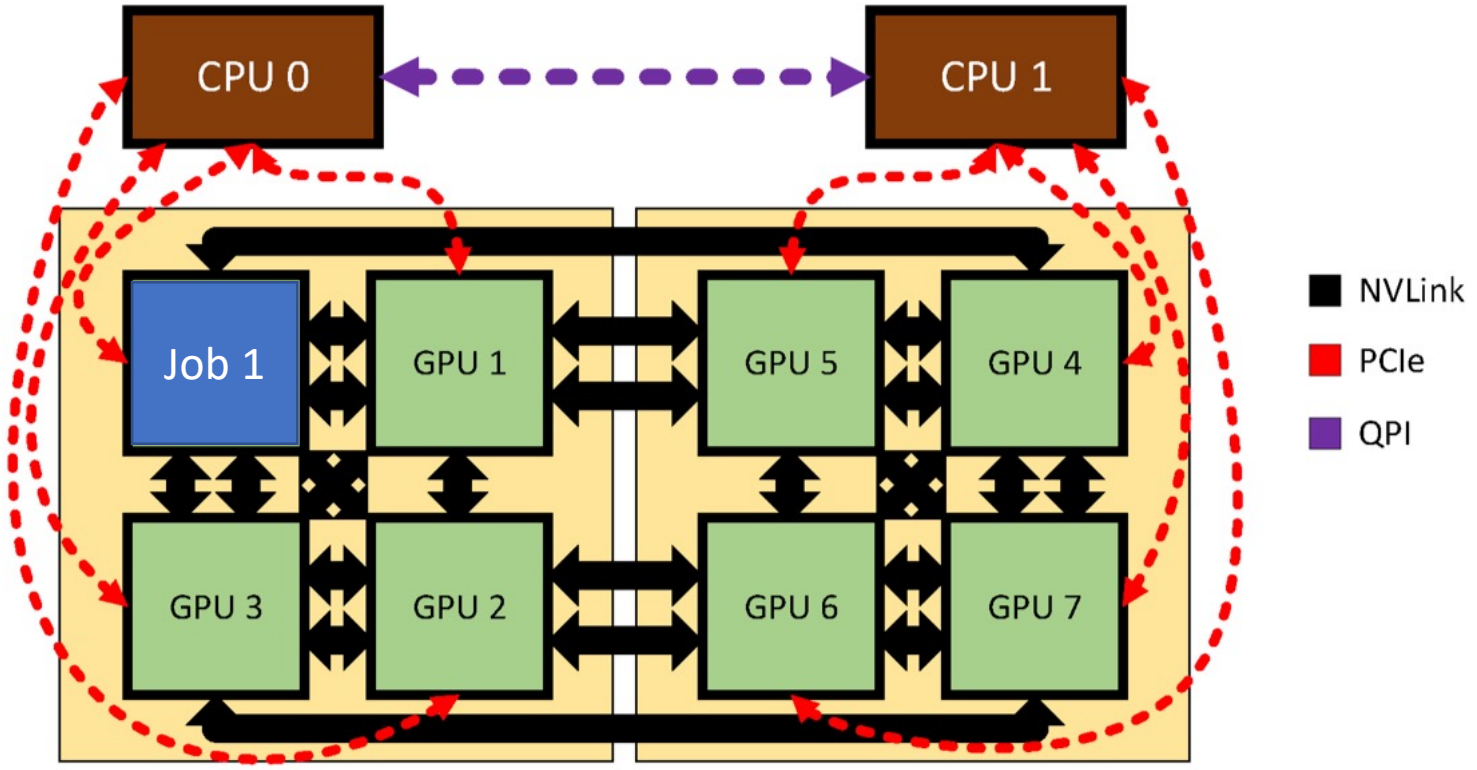
Allocation Problem

- Multi-tenancy
  - Not all applications require the whole system
  - **Fragmentation!**
    - More about this in the next slide.
- Different jobs have different needs
  - Number of GPUs
  - Compute resource utilization
  - Inter-GPU communication

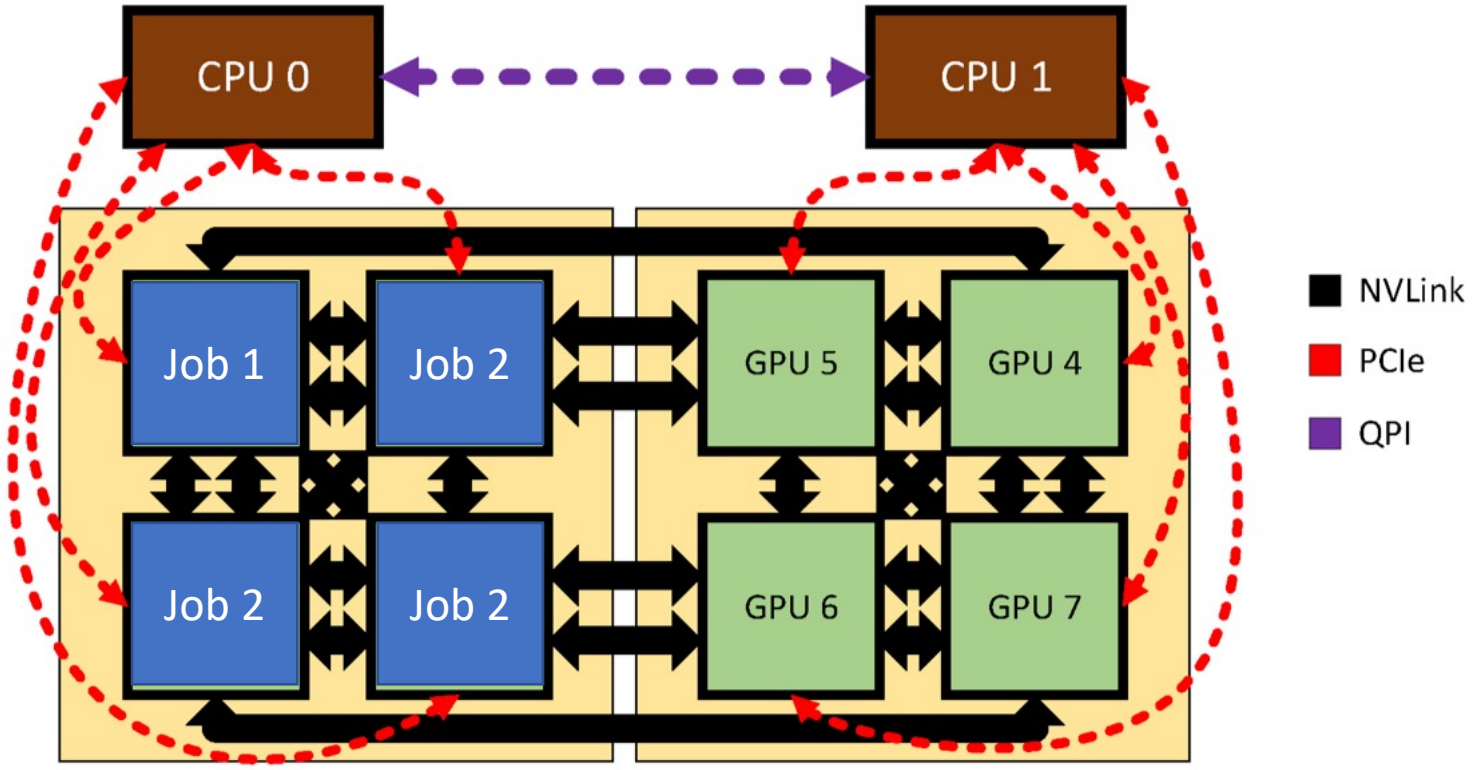


Fragmentation

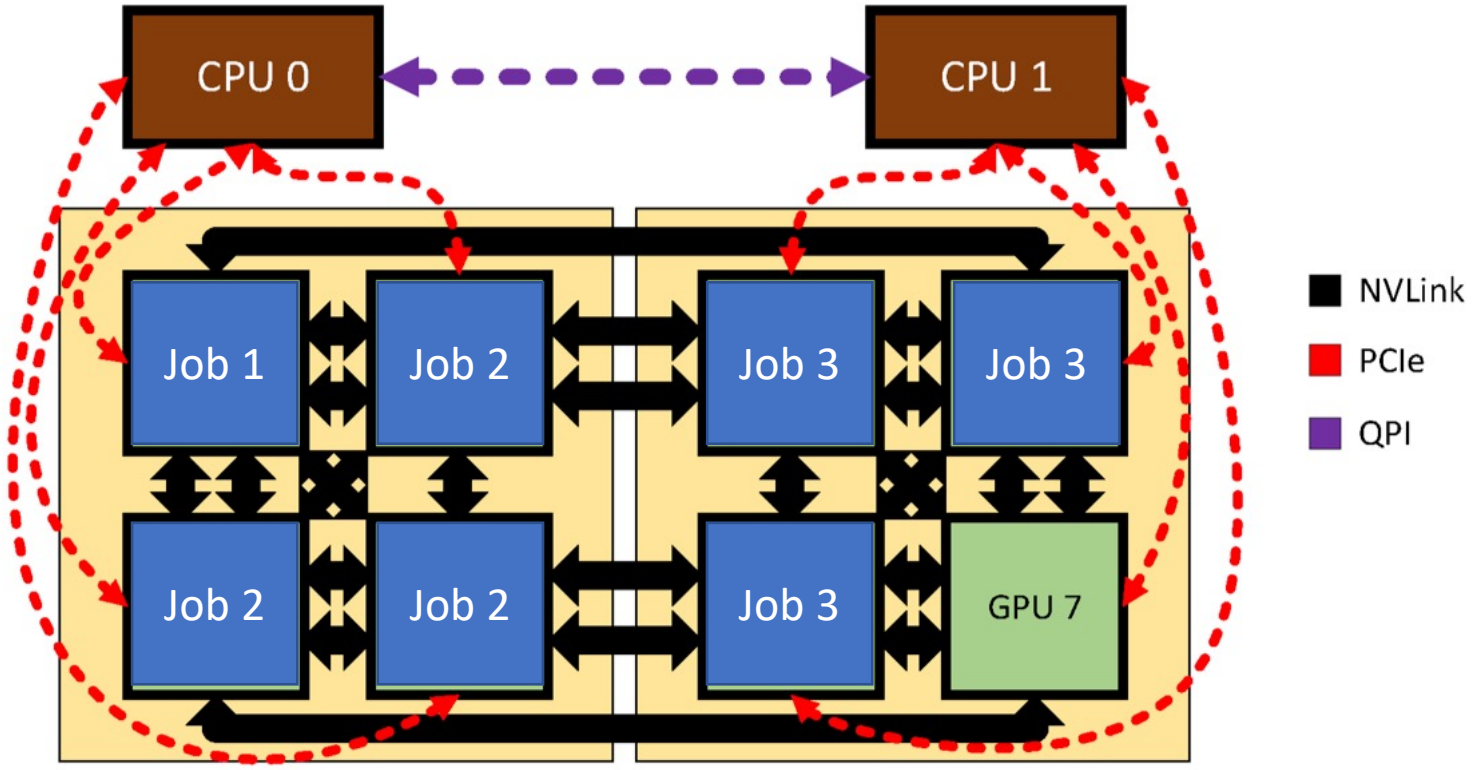
Fragmentation



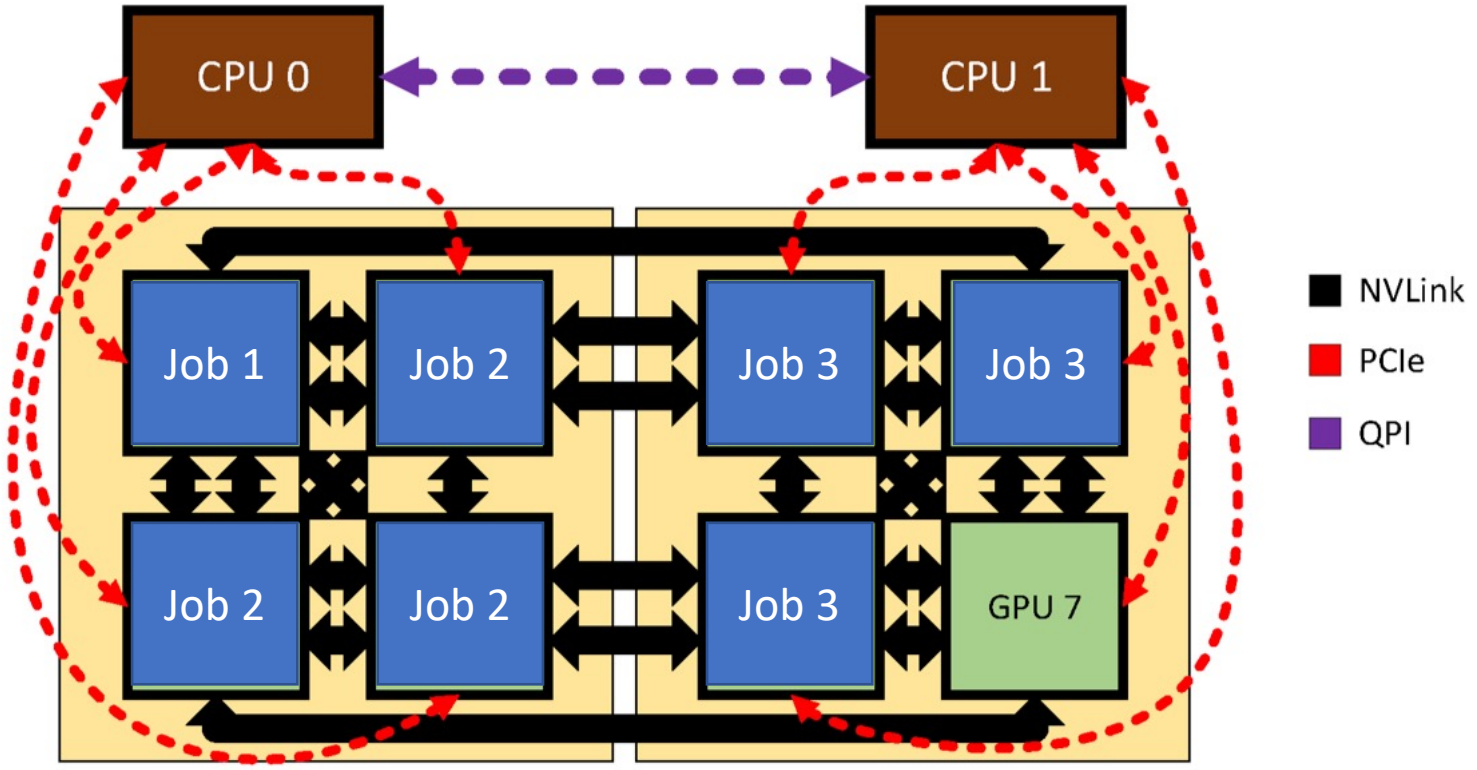
Fragmentation



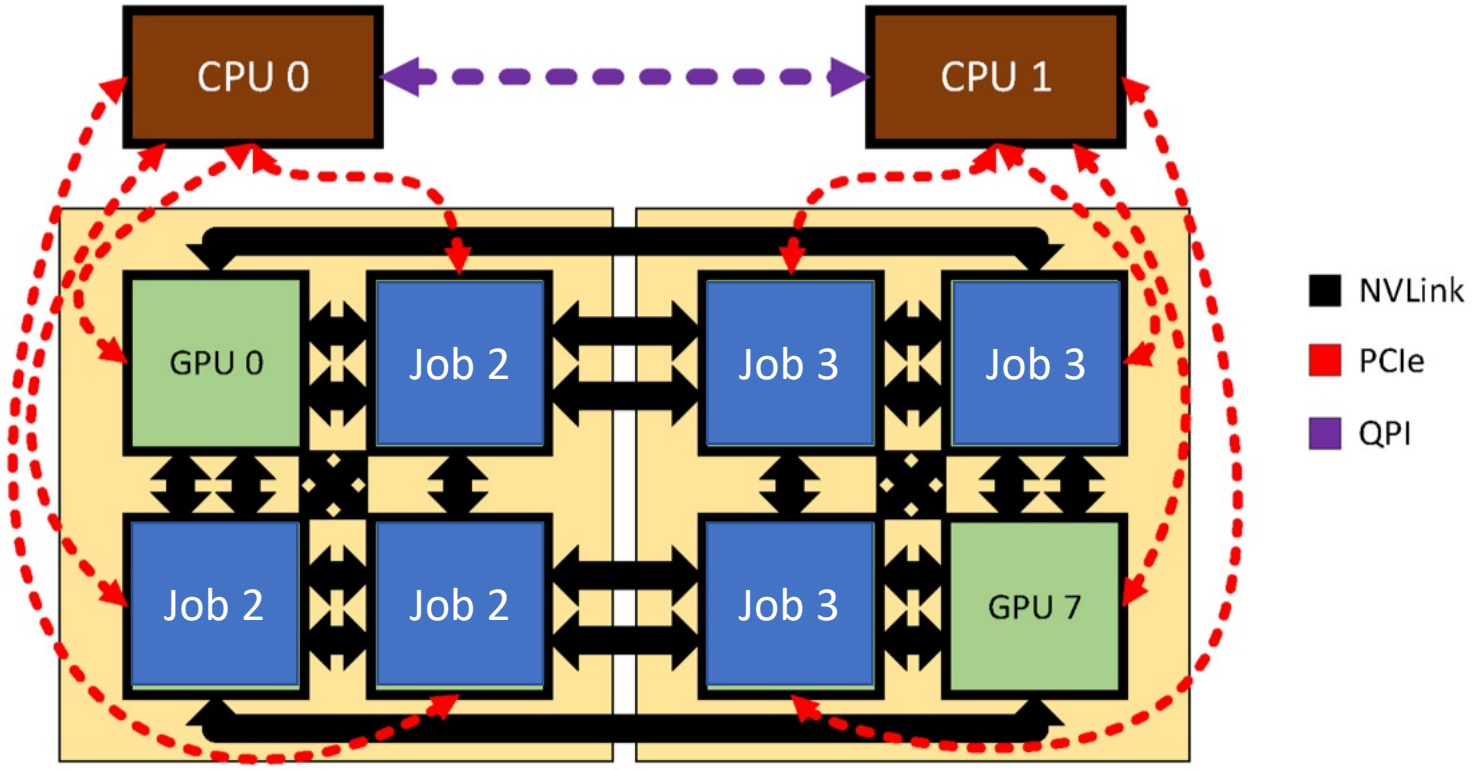
Fragmentation



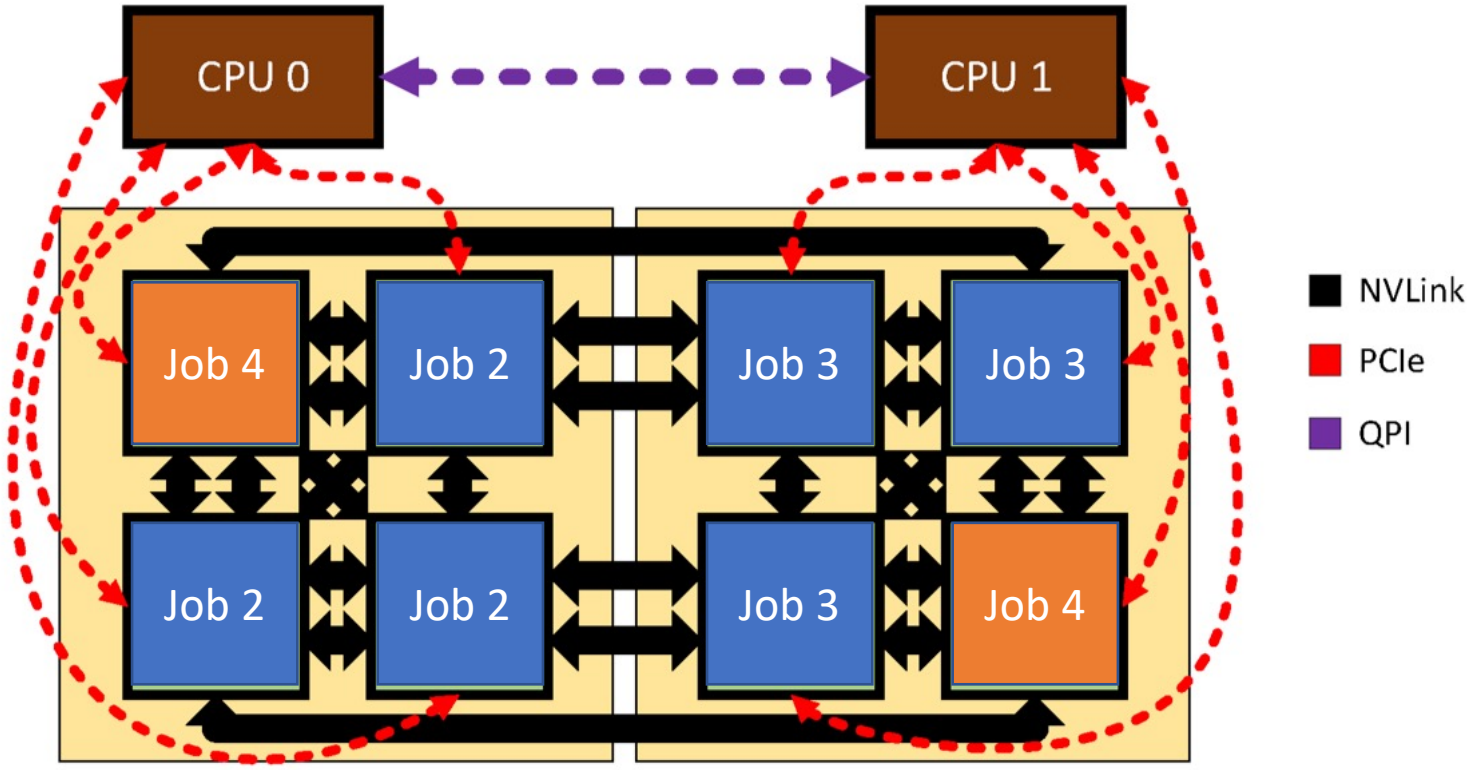
Fragmentation



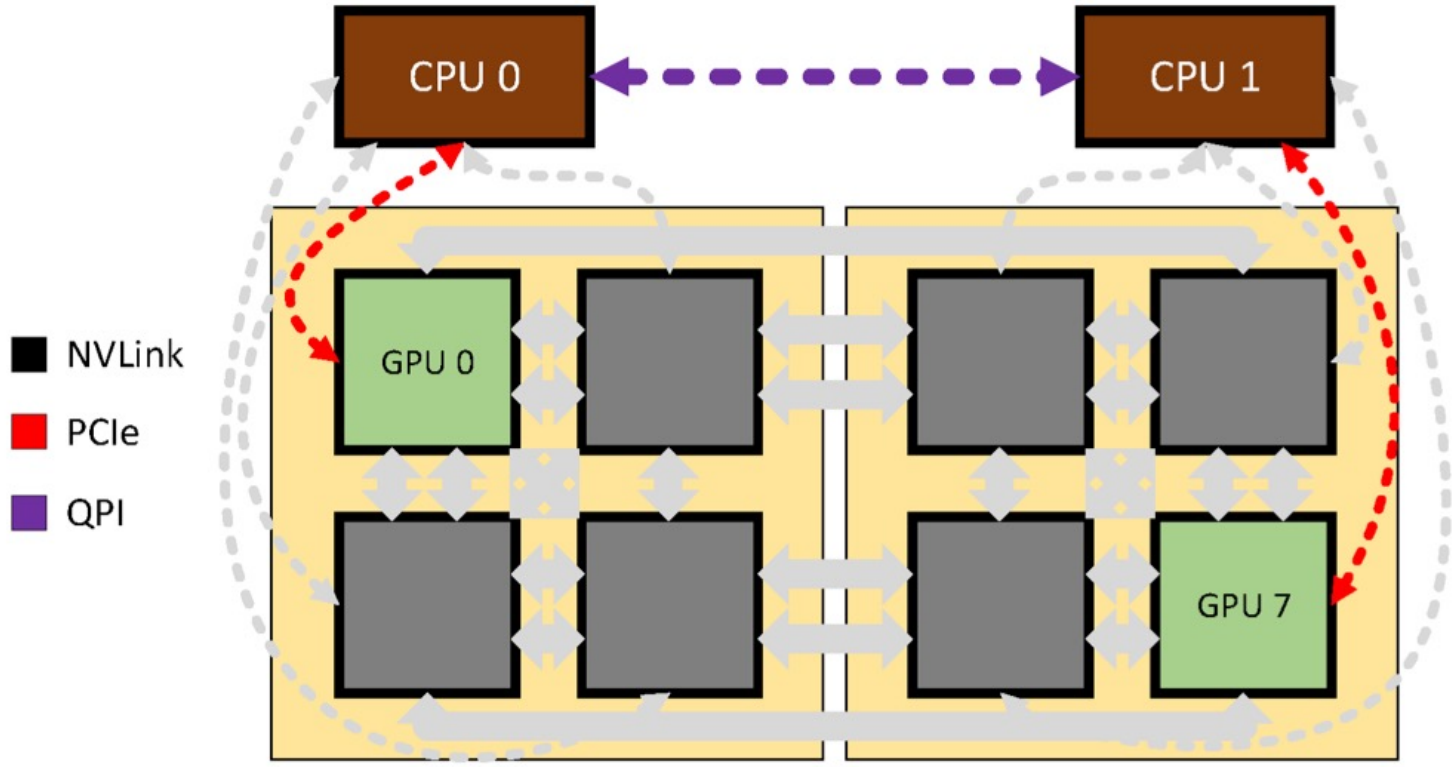
Fragmentation



Fragmentation

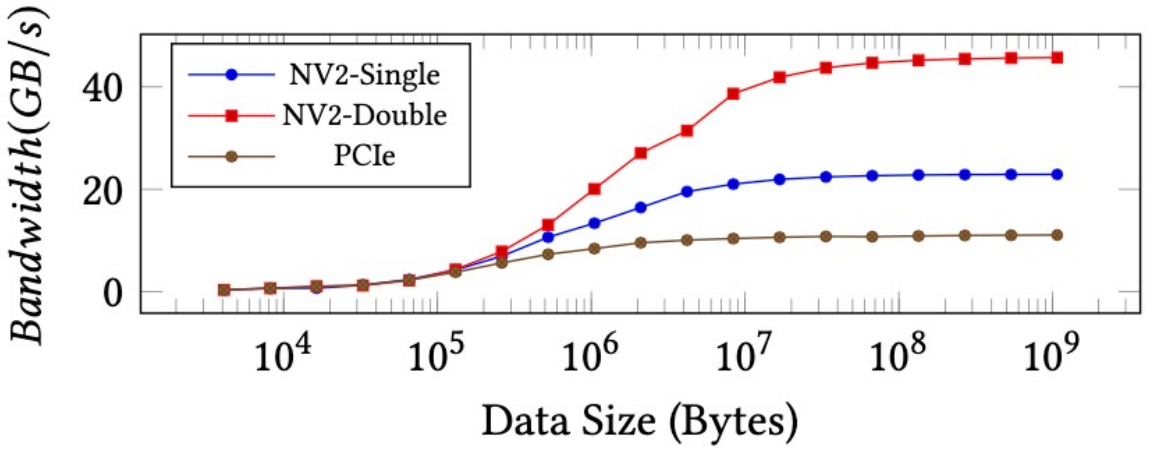


Fragmentation



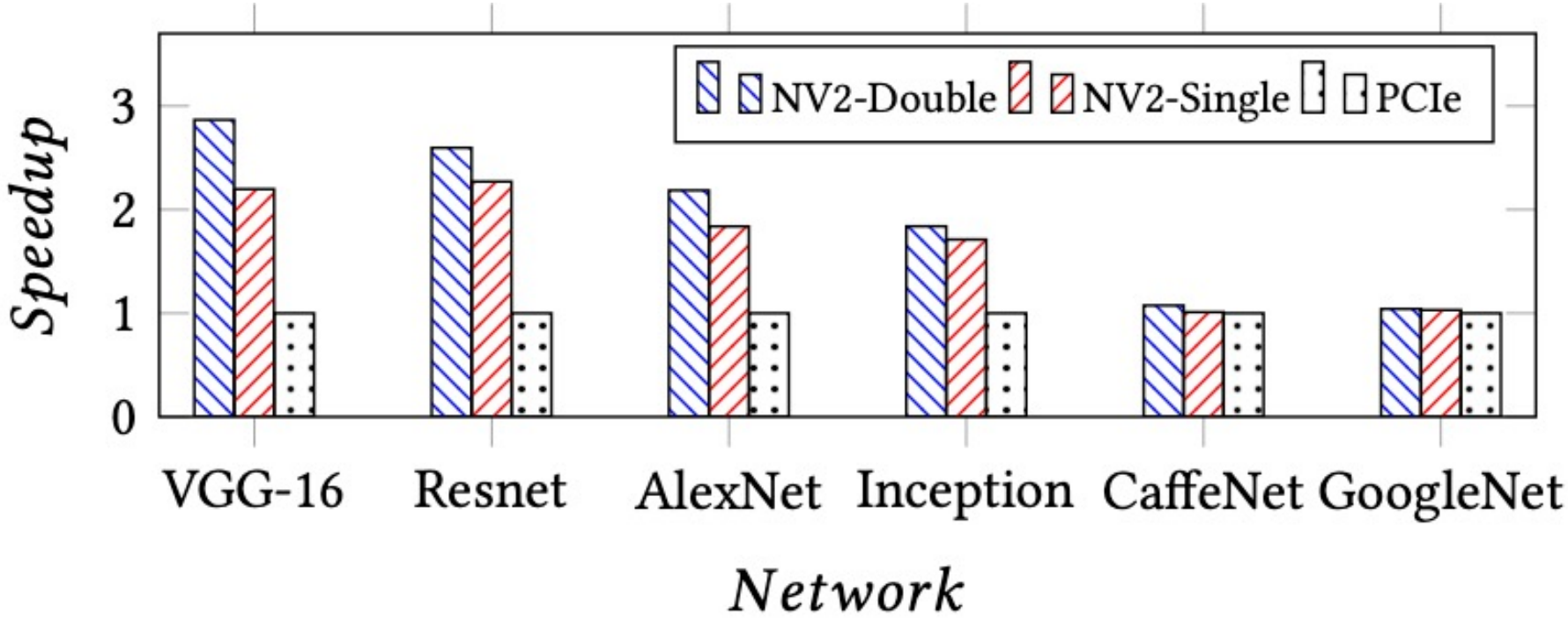
Interconnects – PCIe and NVLink

- PCIe - high-speed serial computer expansion bus standard.
- NVLink - wire-based serial multi-lane near-range communications link.



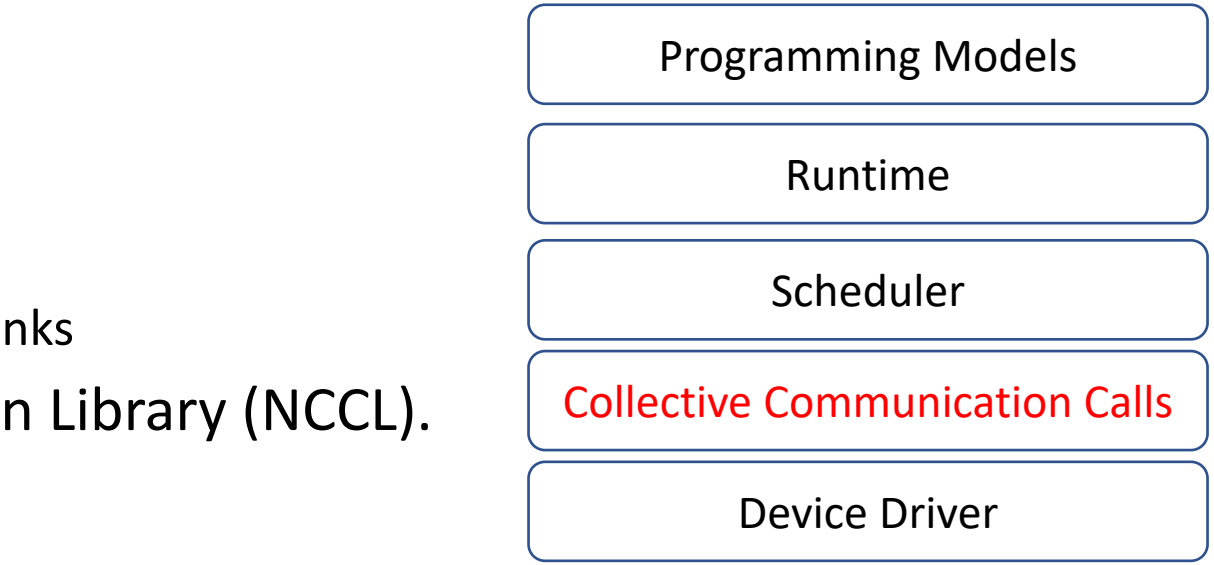
Problems with worst case allocation

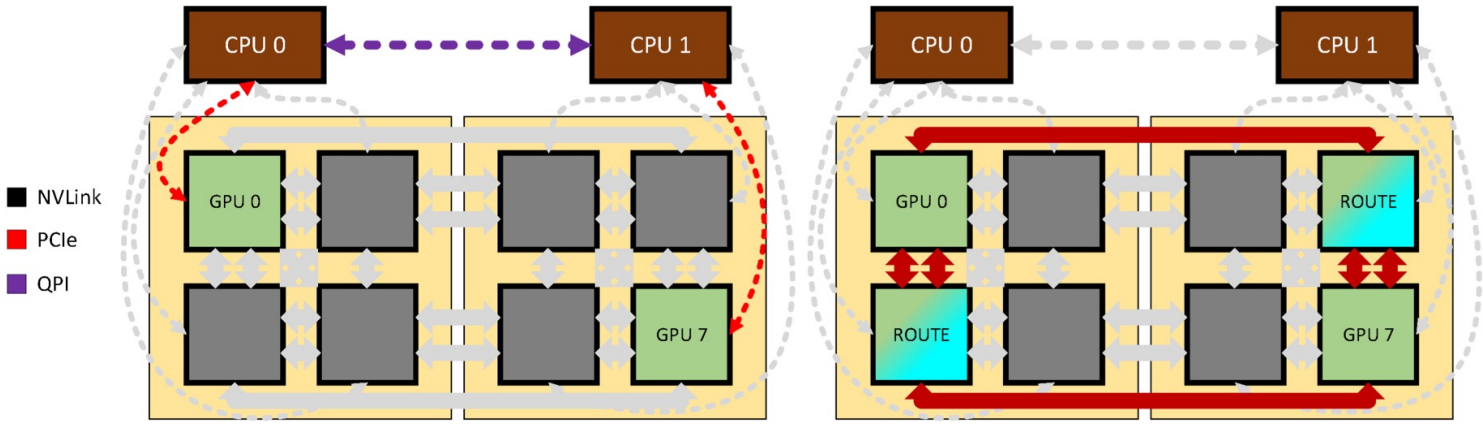
- Speed up can improves over 200% for certain ML training workloads.

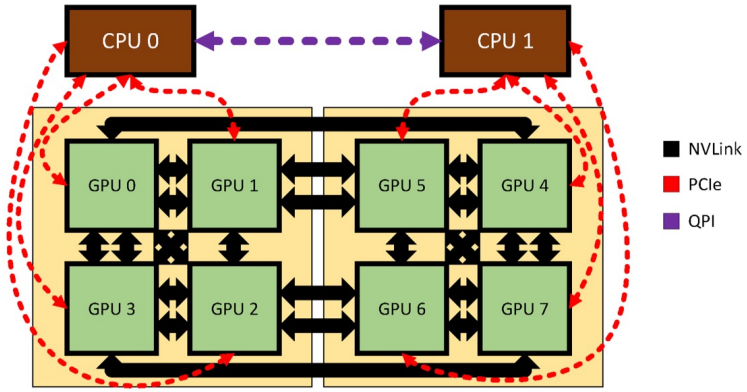


Solution-1: Optimize Worst-case allocations

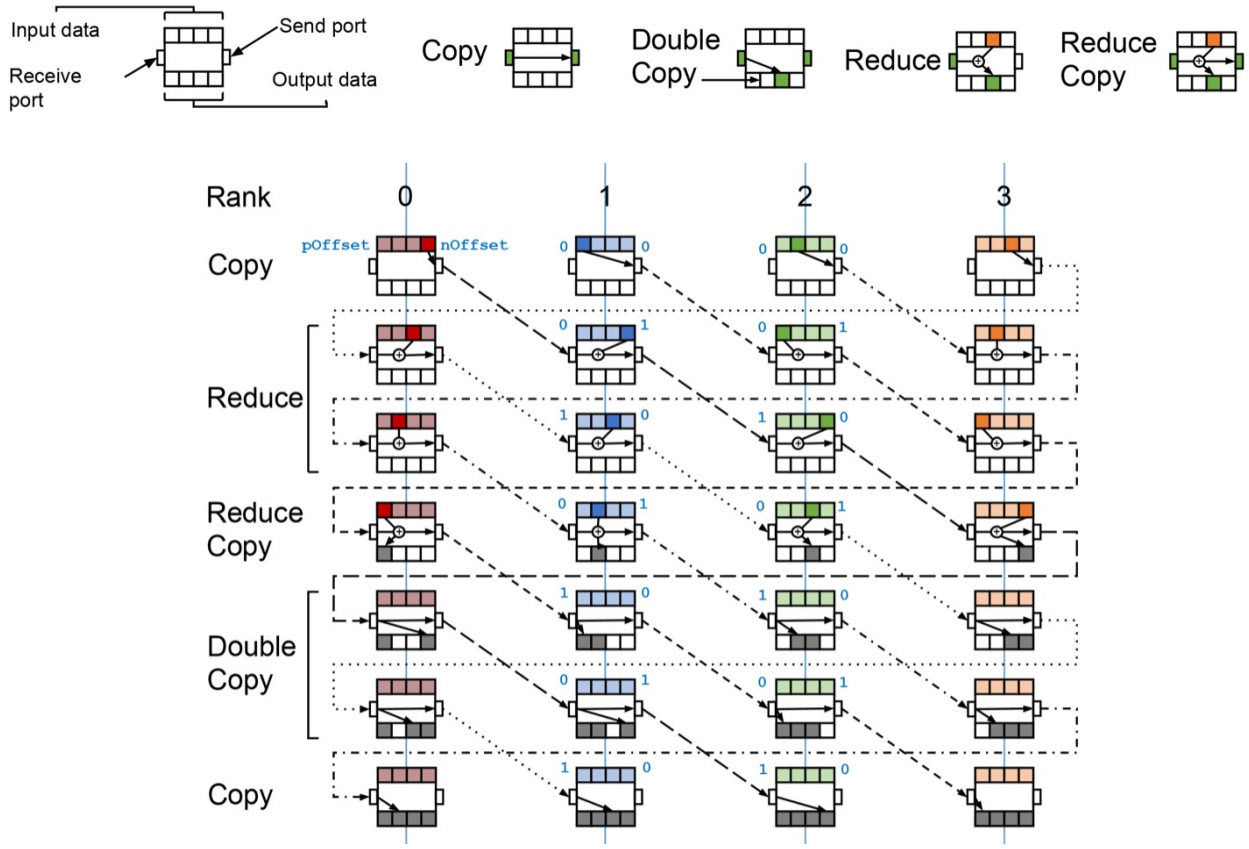
- Assuming Scheduler did its best job.
  - Fragmentation is *inevitable*!
- WOTIR – Nvlink Forwarding
  - Avoid PCIe by hopping over unutilized NVLinks
- Modify Nvidia Collective Communication Library (NCCL).



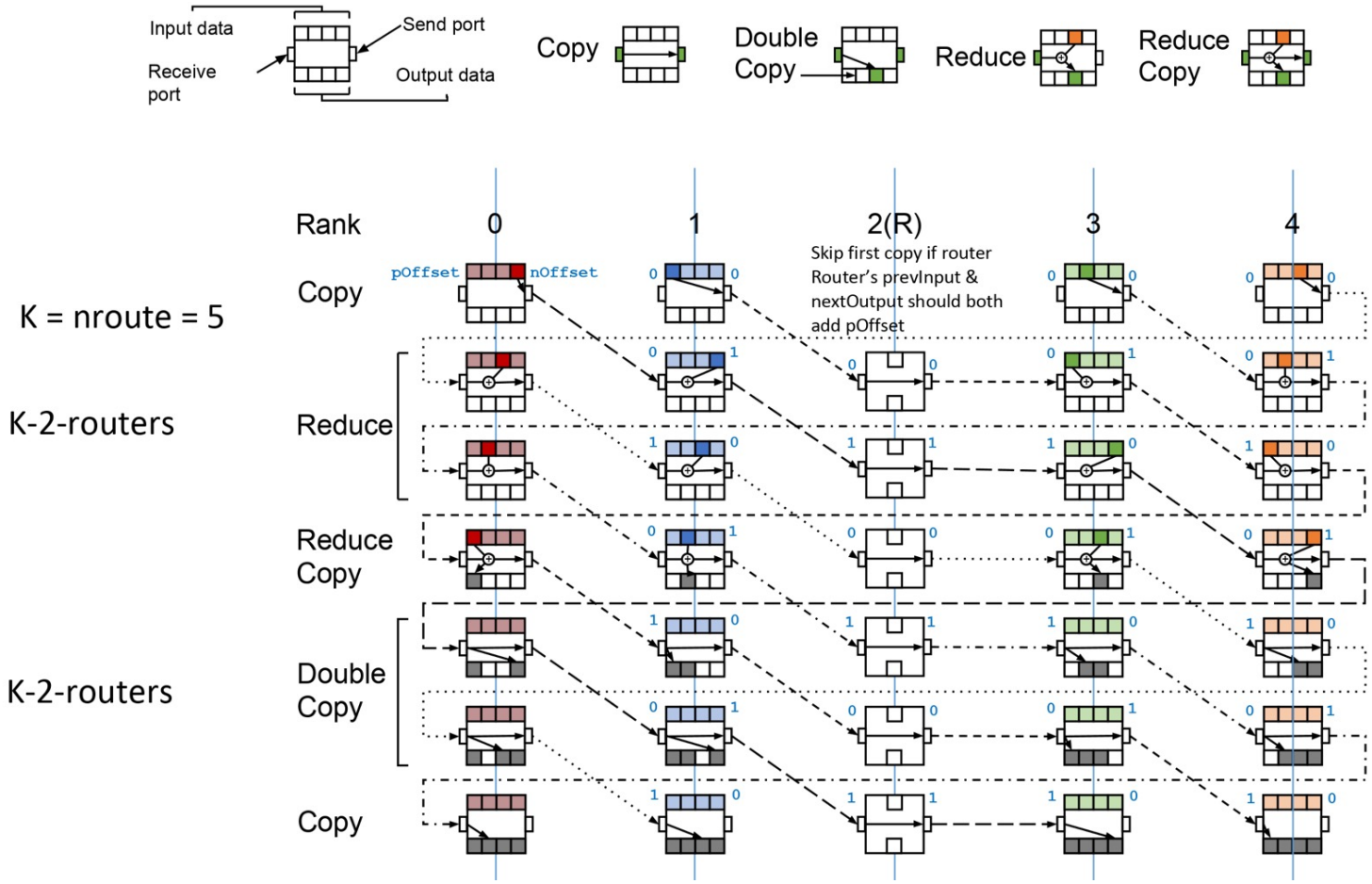




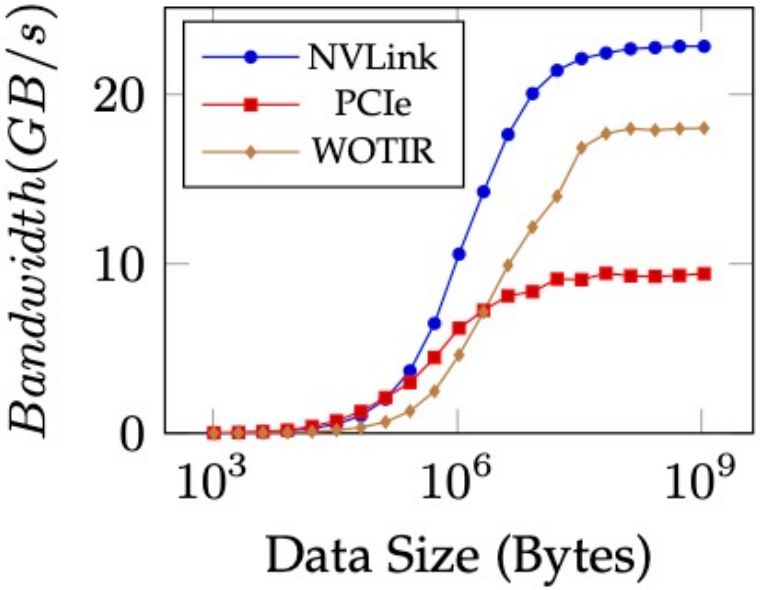
All-Reduce Implementation

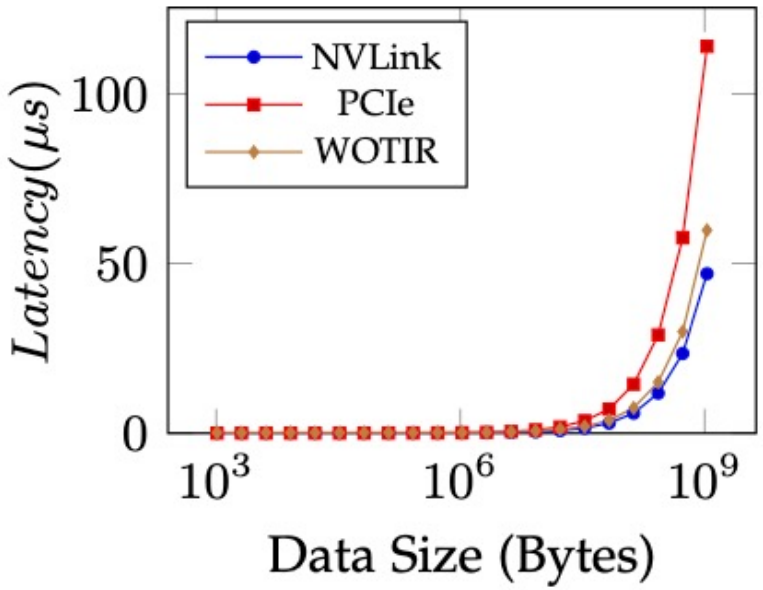


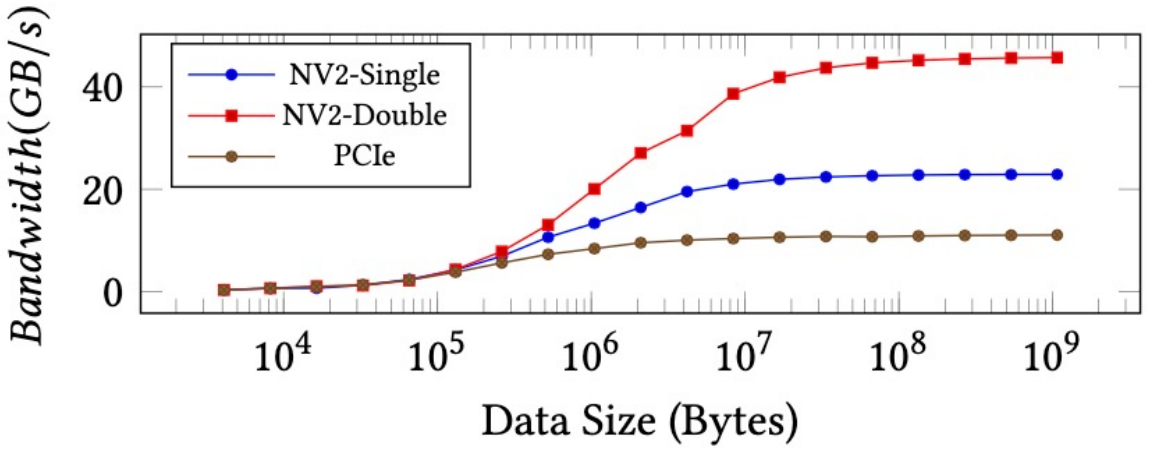
All-Reduce w/ Route Implementation



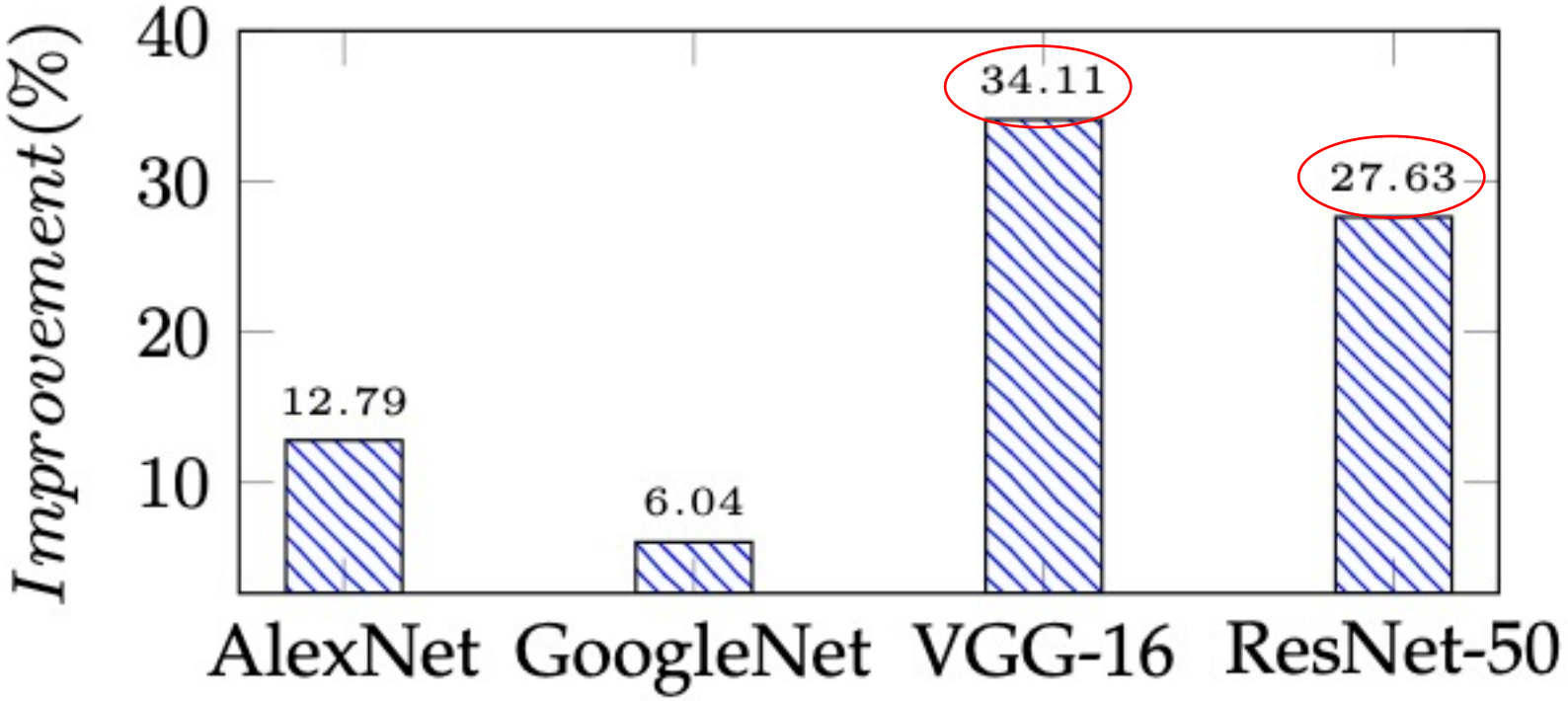
Benefits of WOTIR



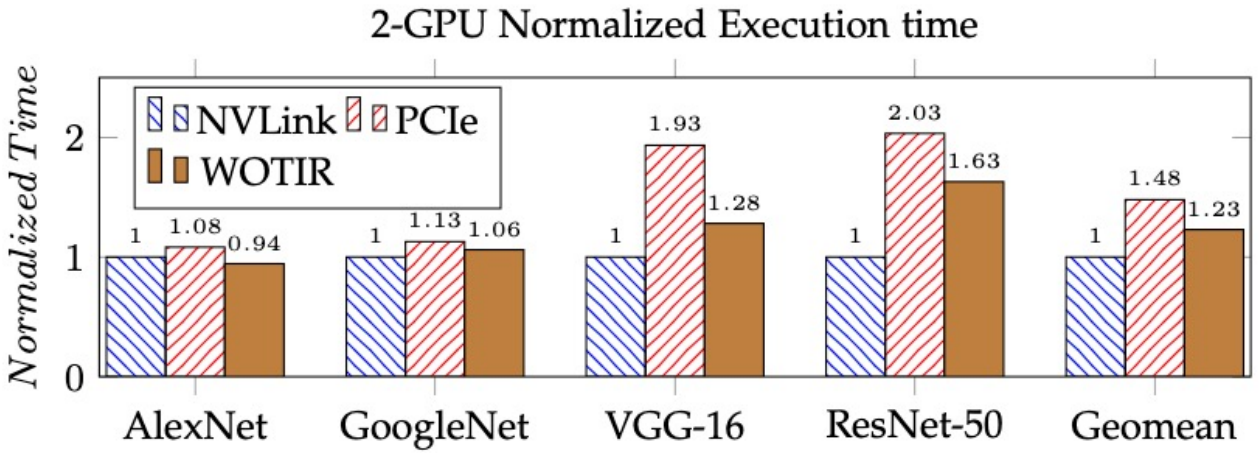


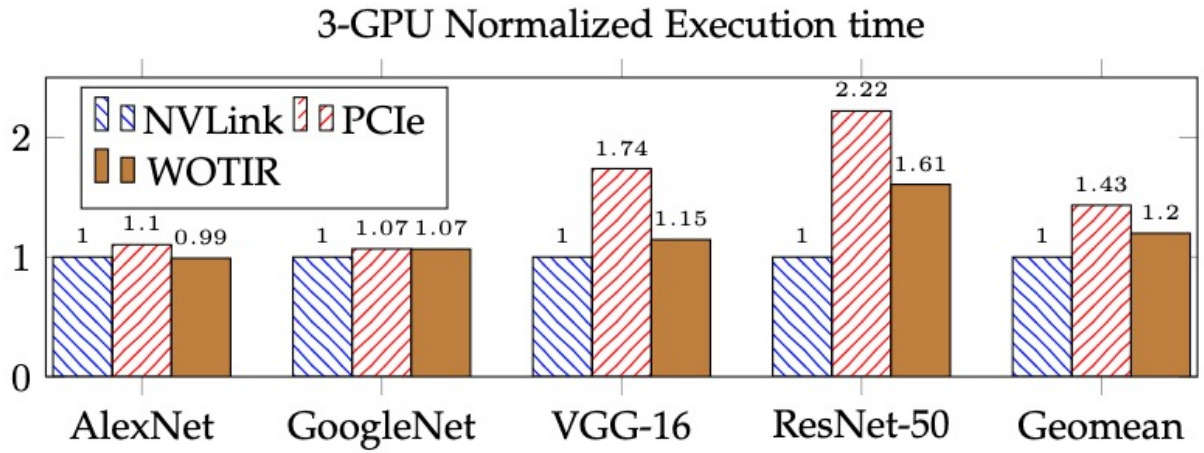


Evaluation: Benefits of WOTIR



Evaluation: Benefits of WOTIR



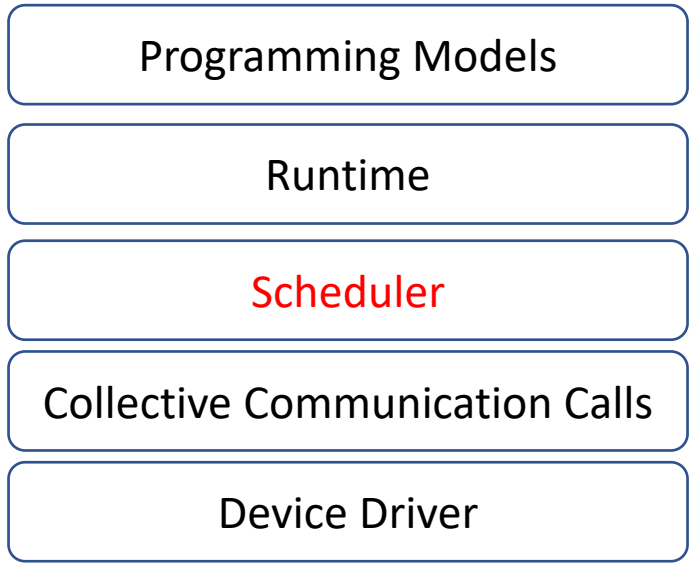


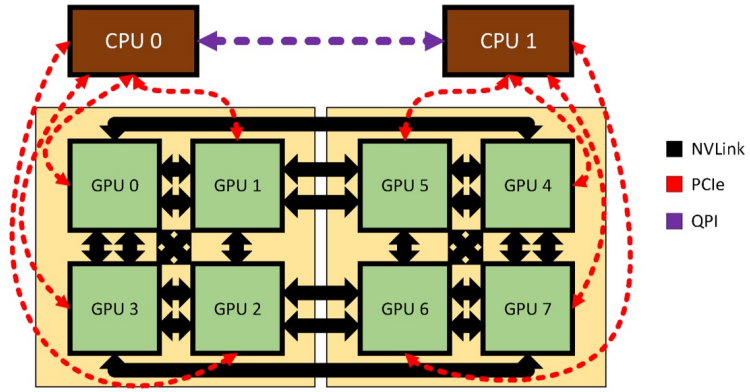
Overhead of WOTIR

State	Min (ms)	25th% (ms)	Median (ms)	75th% (ms)	95th% (ms)	Max (ms)
w/o Routing	2.033	2.034	2.035	2.036	2.040	3.140
w/ Routing	2.034	2.035	2.036	2.053	5.674	11.459

Can we avoid worst-case allocations?

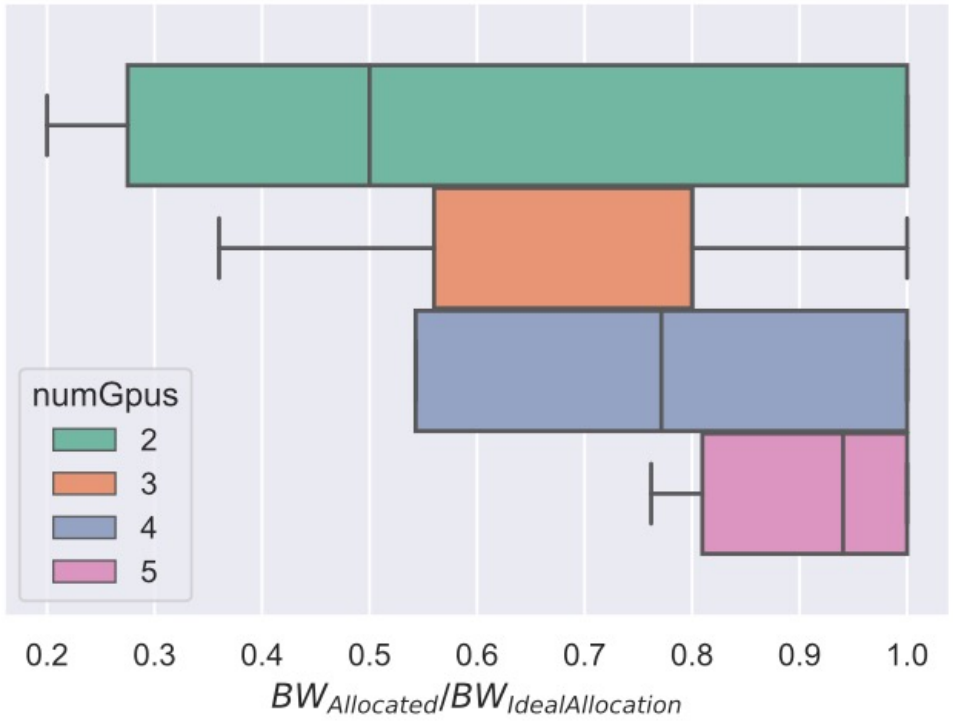
- Intelligent scheduling.
- Scheduling policies have been explored since 1960s.
- Challenges in using existing Scheduling policies.
- Heterogeneity in Compute resources
  - Earlier we mostly one type of compute resource (CPUs) and they were mostly connected via one type of interconnect.





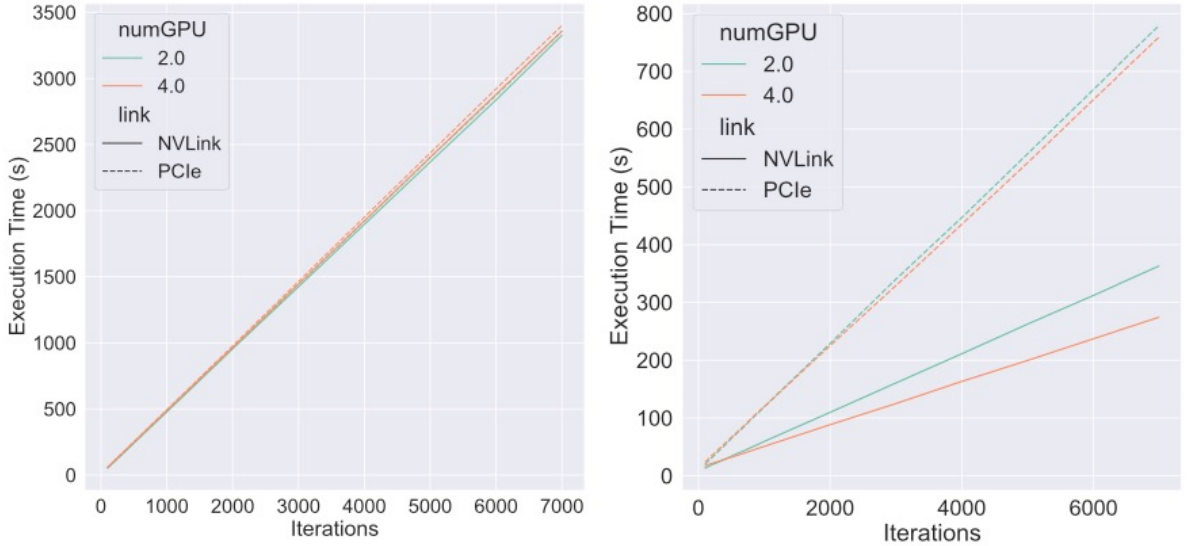
Problems with Naïve Scheduling

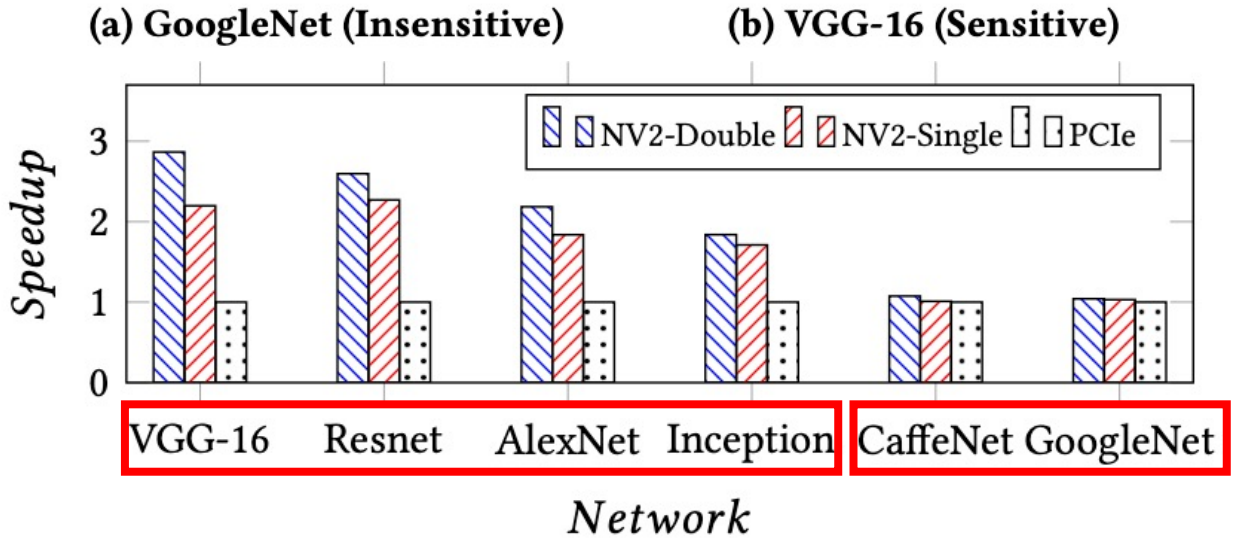
- Problems with naïve scheduling



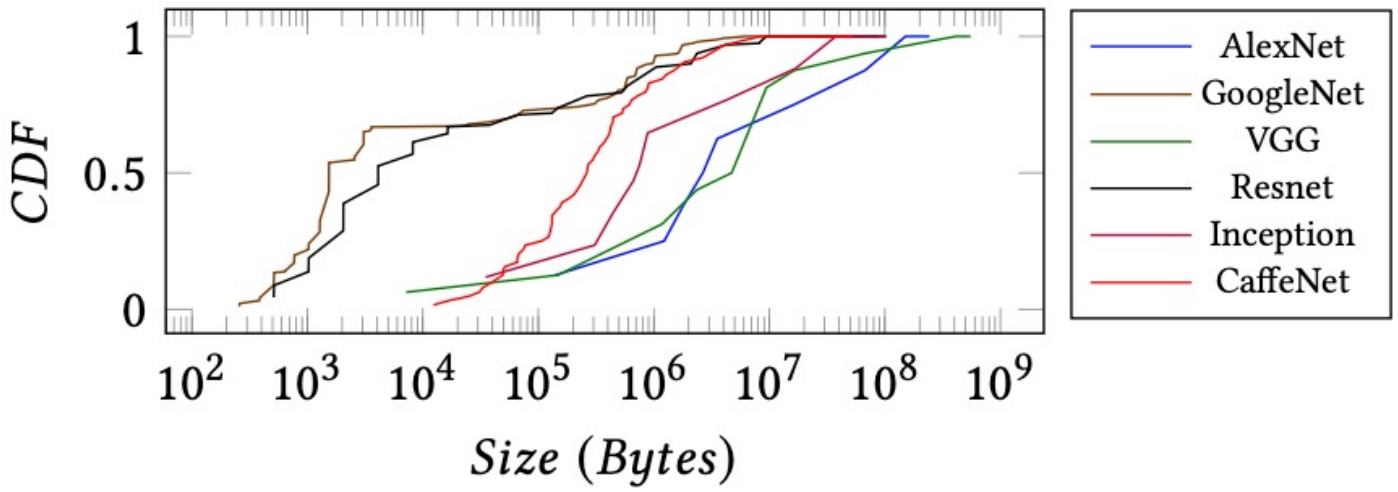
Worst case allocations

- Some applications benefit from better allocations while some do not.
- Can we use this to our advantage to improve system utilization and throughput?

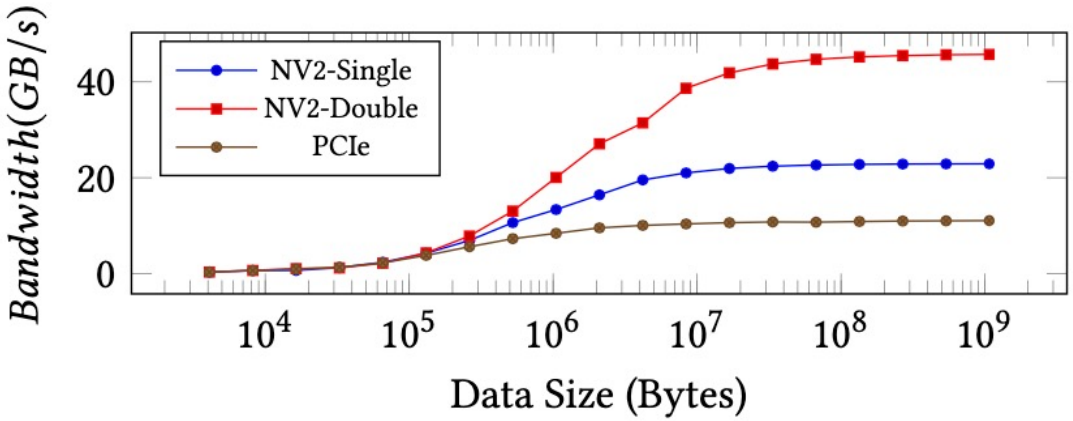


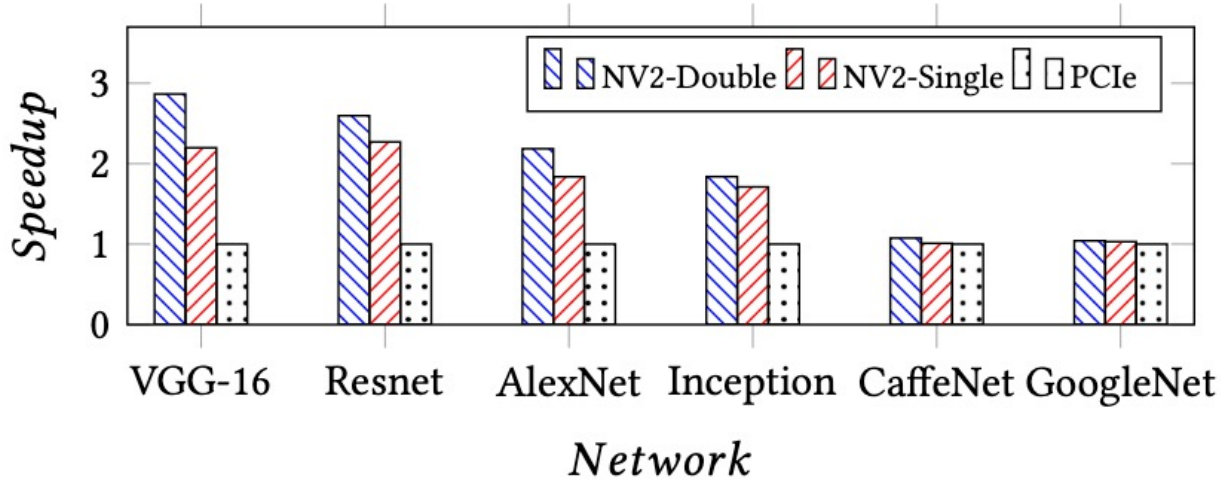


Bandwidth sensitivity



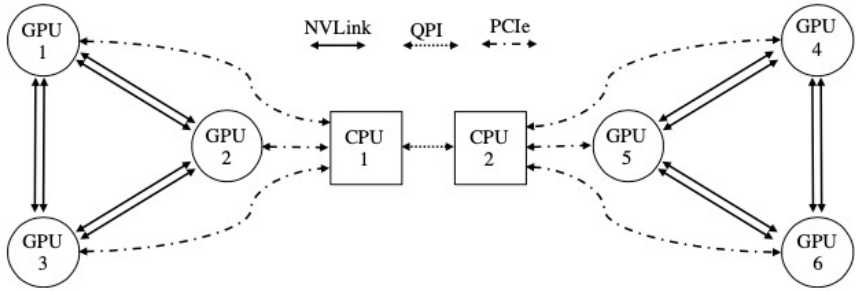
Network	Communication calls per iter.	Bandwidth Sensitive
AlexNet	80,001	Yes
Inception-v3	2,830,001	Yes
VGG-16	160,001	Yes
Resnet-50	1,600,001	Yes
CaffeNet	84,936	No
GoogleNet	640,001	No



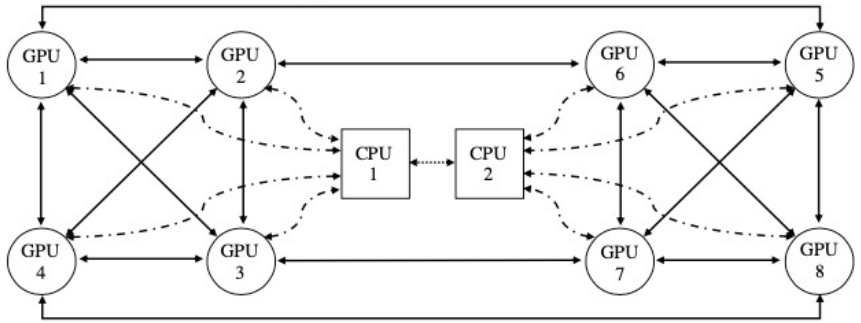


Graph-based Scheduling for Multi-Accelerator Systems

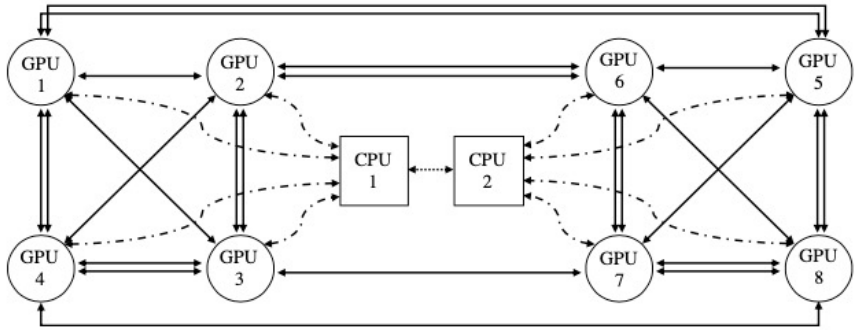
- Hardware Topologies
  - Rapidly changing architectures
  - Modern interconnects differ significantly from precursor
  - Different Programming Models
  - Different Runtime Systems



(a) Summit V100

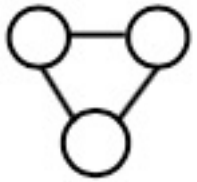


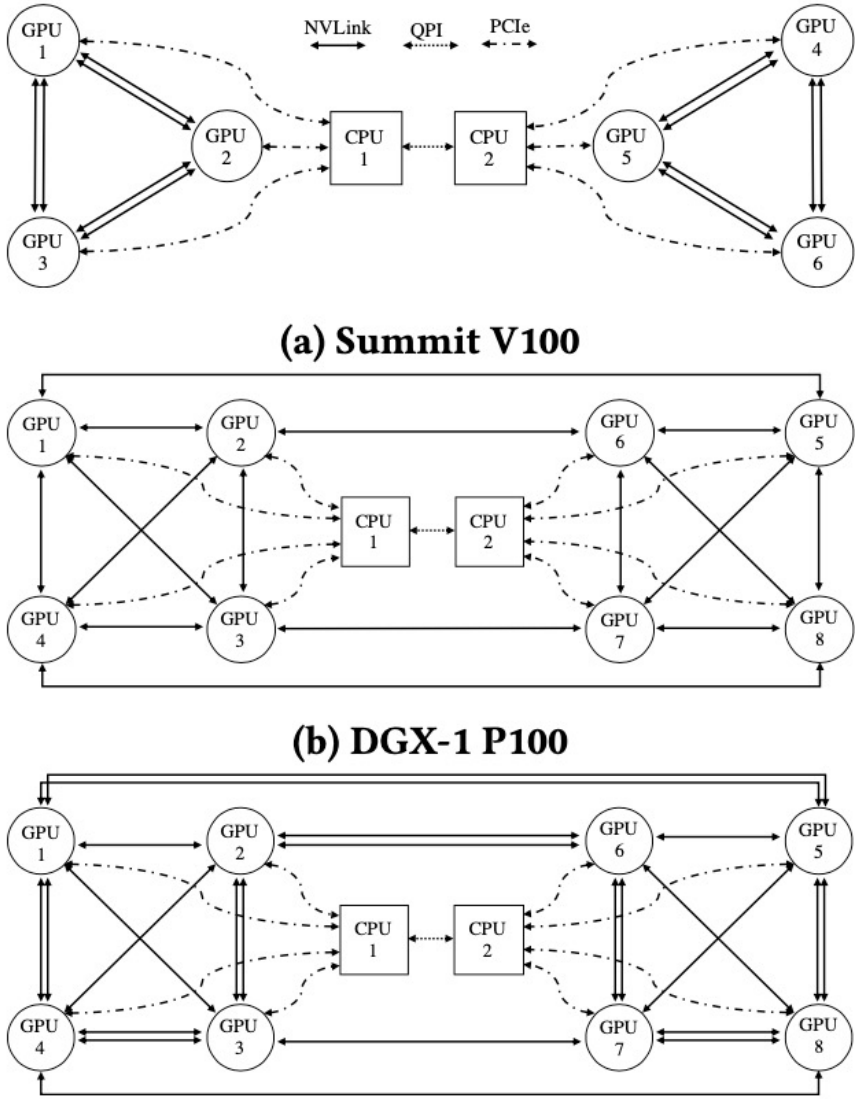
(b) DGX-1 P100



Graph-based Scheduling for Multi-Accelerator Systems

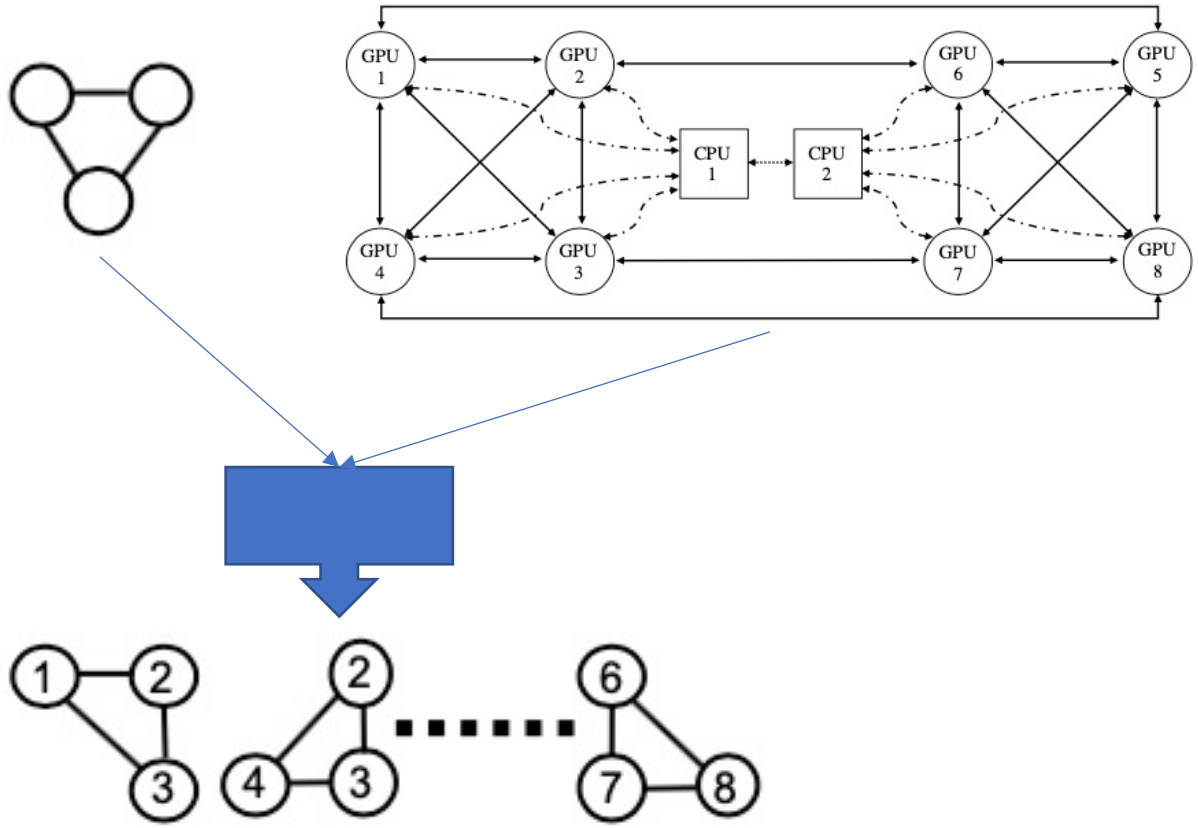
- Hardware Topology
- Application Topology
  - Different compute requirements
    - 1-CPU, 2-GPUs, etc.
  - Communication intensity among nodes.





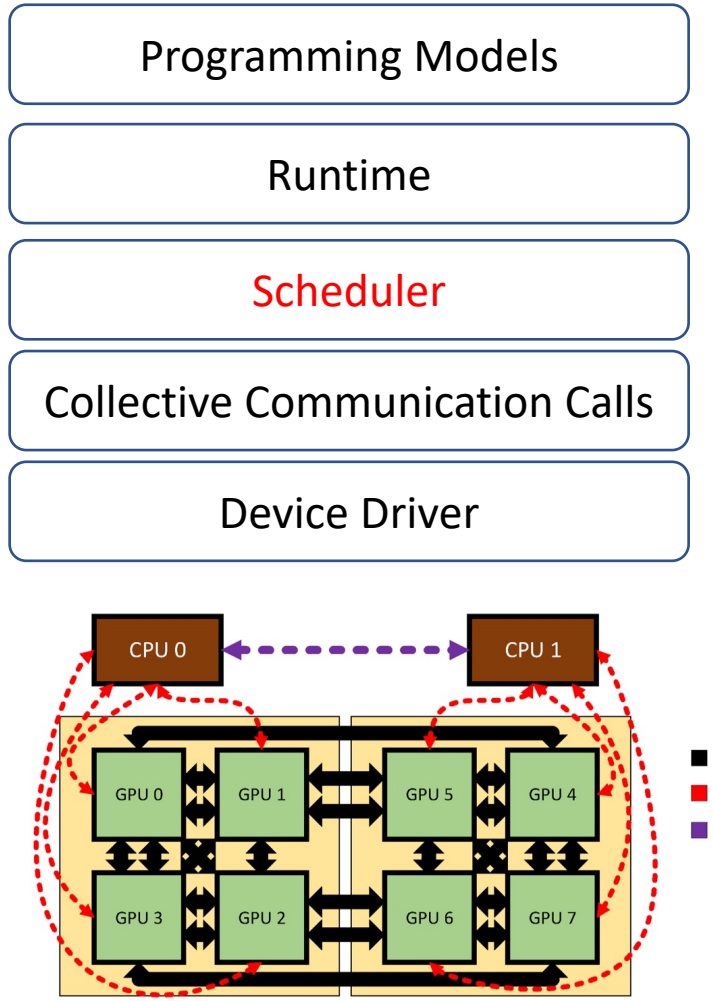
Graph-based Scheduling for Multi-Accelerator Systems

- Hardware Topology
- Application Topology
- Find ideal allocations
  - Use knowledge of Hardware and Application topologies.

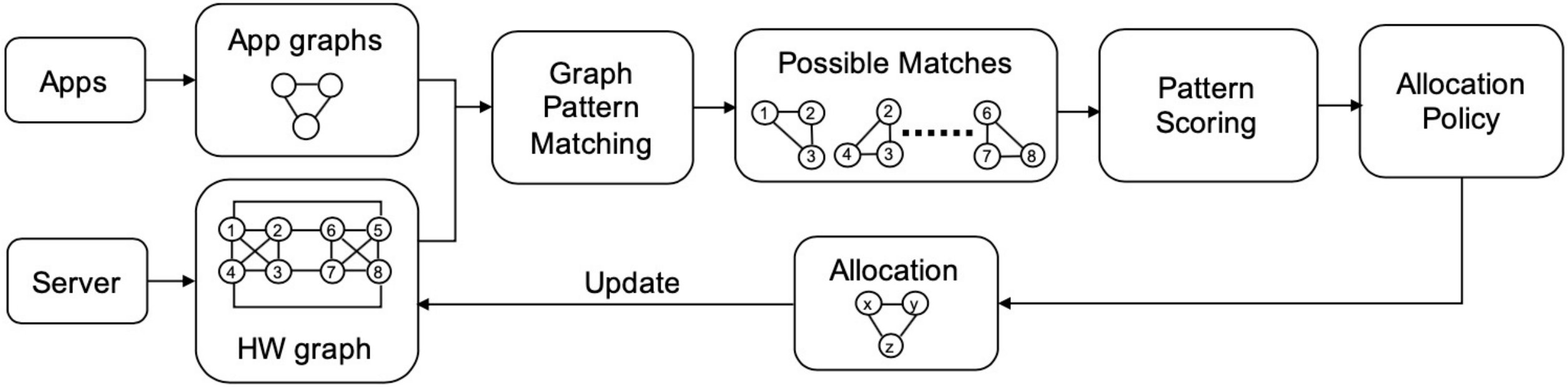


Optimization Spaces

- How can modern Multi-accelerator schedulers be designed?



Multi-Accelerator Pattern Allocation (MAPA)



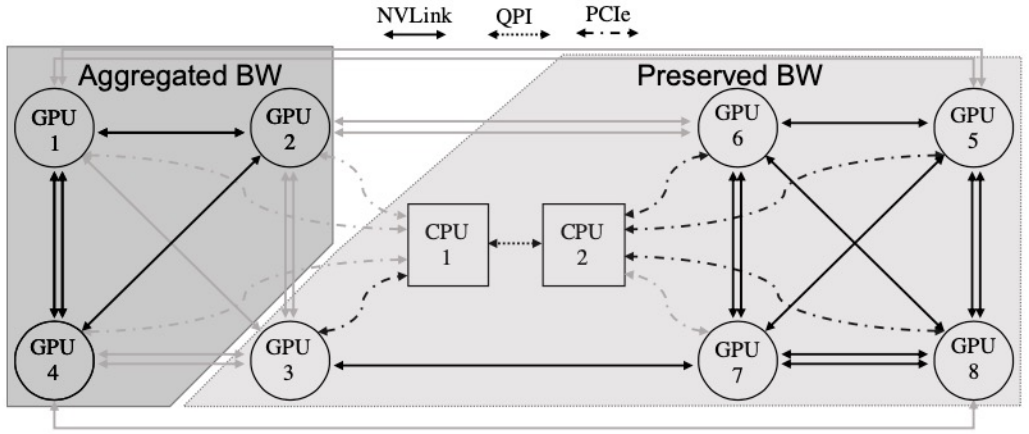
Scoring allocations

- Aggregated BW (Agg BW) – Sum of available BW for a given application topology within an allocation.

$$AggBW = \sum_{e \in (E(P) \cap E(M))} w(e)$$

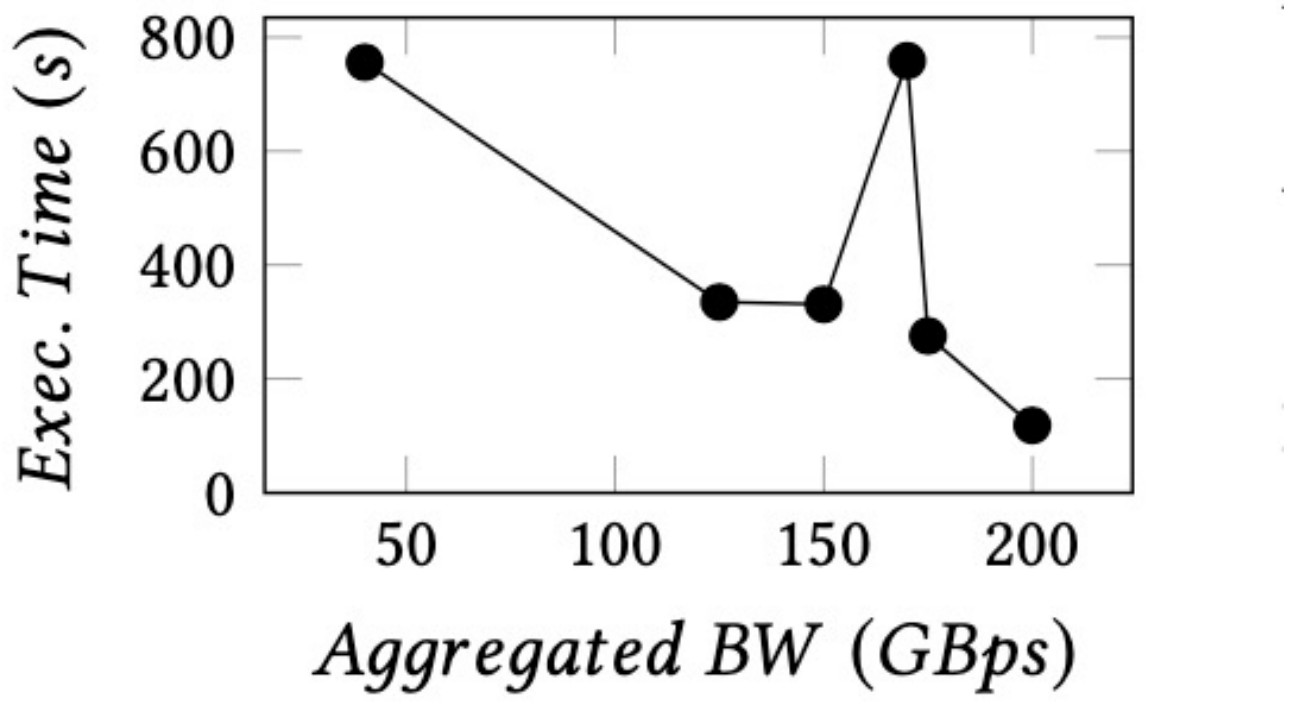
- Preserved BW – Sum of available BW given an allocation is scheduled.

$$Preserved\ Bandwidth = \sum_{e \in E(G \setminus M)} w(e)$$



Problems with AggBW

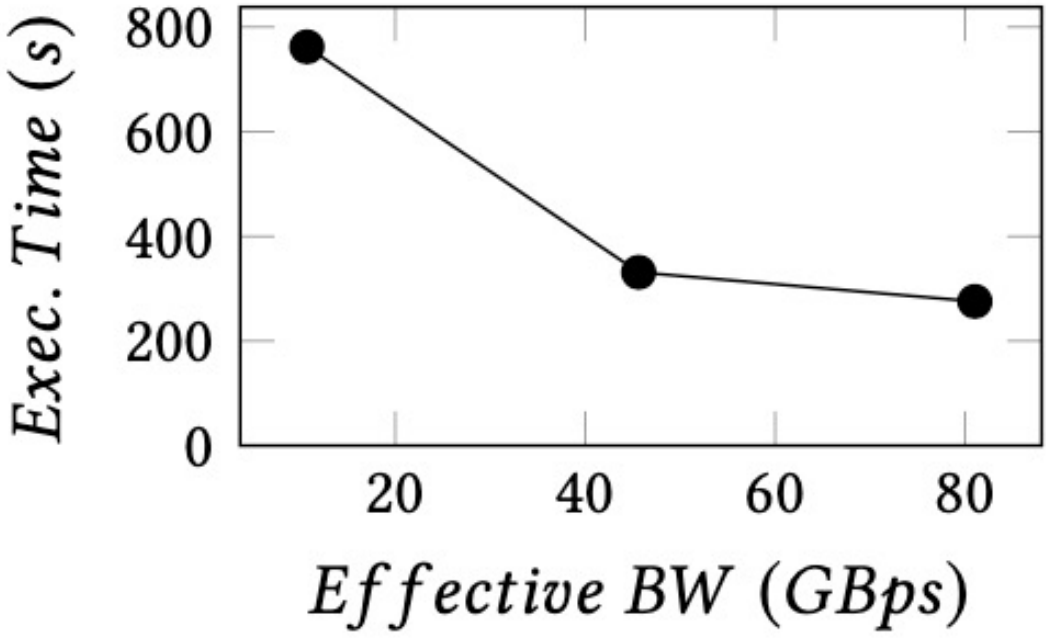
- AggBW does not translate to improved performance.



(a) VGG-16 training Execution Time

Effective BW (Eff BW)

- Bandwidth achievable in given allocation.
- Eff BW is obtained by running a NCCL-based microbenchmark.
  - All-Reduce benchmark on largest contiguous data block.



(c) VGG-16 training Execution Time

Modelling Eff BW

- It is not practical to run microbenchmarks to obtain Eff BW scores.
- Non-linear Polynomial Regression.
  - (x,y,z) – (Double NVLinks, Single NVLinks, PCIe links)

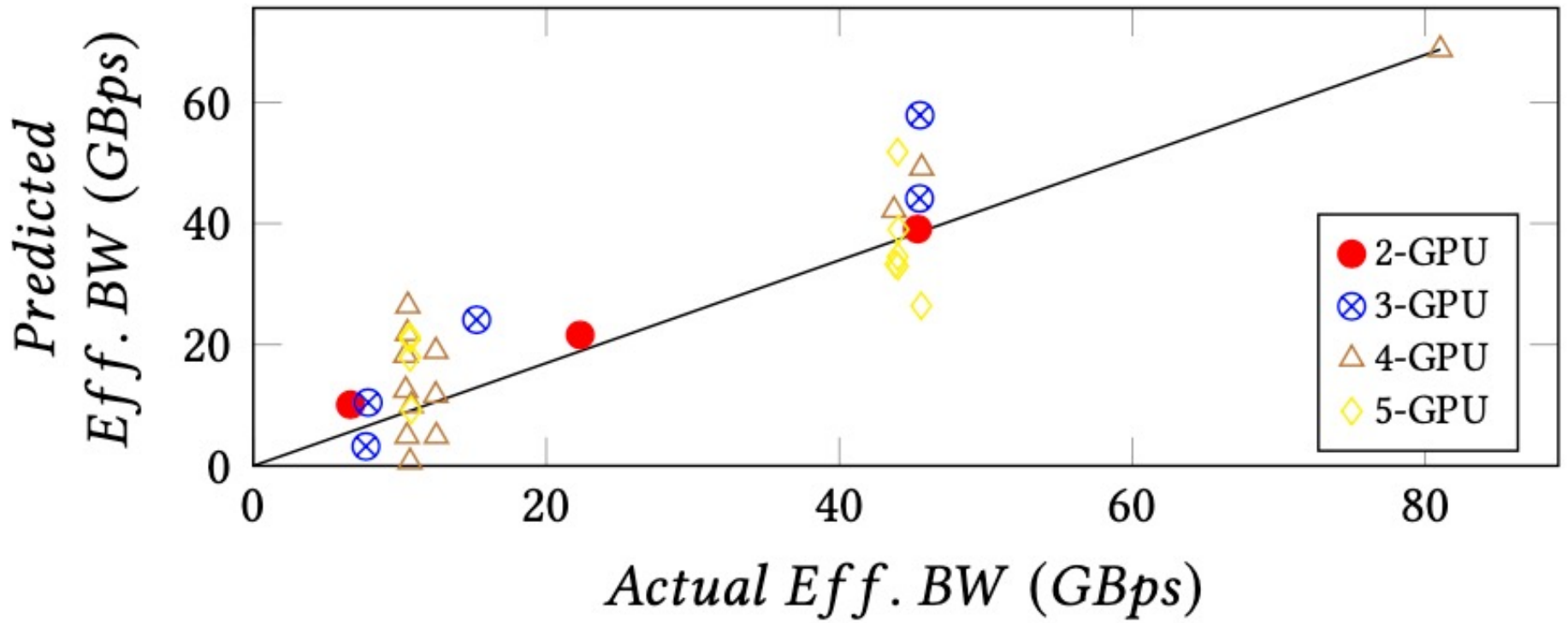
$$Predicted\ Effective\ Bandwidth =$$

$$\theta_1 x + \theta_2 y + \theta_3 z + \theta_4 \frac{1}{x+1} + \theta_5 \frac{1}{y+1} + \theta_6 \frac{1}{z+1}$$

$$+ \theta_7 xy + \theta_8 yz + \theta_9 zx + \theta_{10} \frac{1}{xy+1} + \theta_{11} \frac{1}{yz+1} + \theta_{12} \frac{1}{zx+1}$$

$$+ \theta_{13} xyz + \theta_{14} \frac{1}{xyz+1}$$

Predicted Eff BW

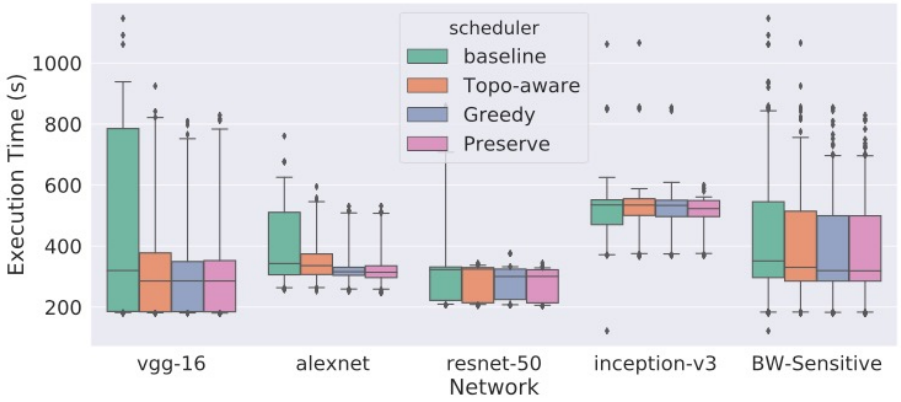


Scheduling Policy

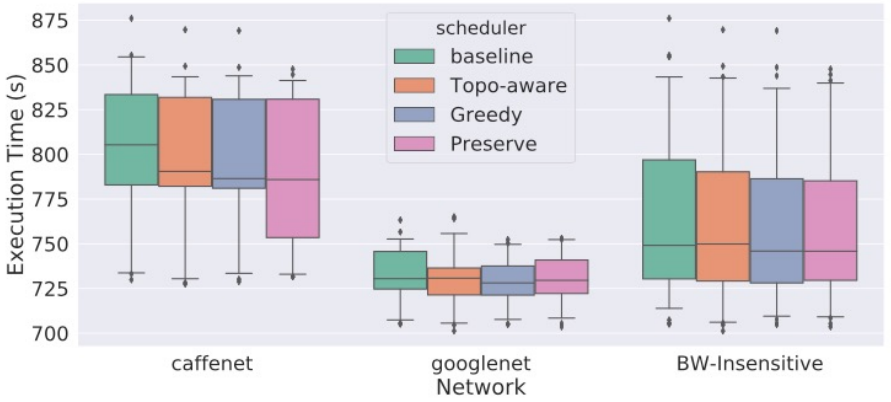
Algorithm 1: Preserve Allocation Policy

```
Result: Allocation
HWgraph hGraph;
AppGraph aGraph;
Allocation alloc;
Patterns possiblePatterns = graphPatternMatching (aGraph,
  hGraph);
if aGraph is bwSensitive then
    foreach pattern in possiblePatterns do
        if EffectiveBW (pattern) > EffectiveBW (alloc) then
            alloc = pattern;
        end
    end
end
else
    foreach pattern in possiblePatterns do
        if PreservedBW (pattern) > PreservedBW (alloc) then
            alloc = pattern;
        end
    end
end
```

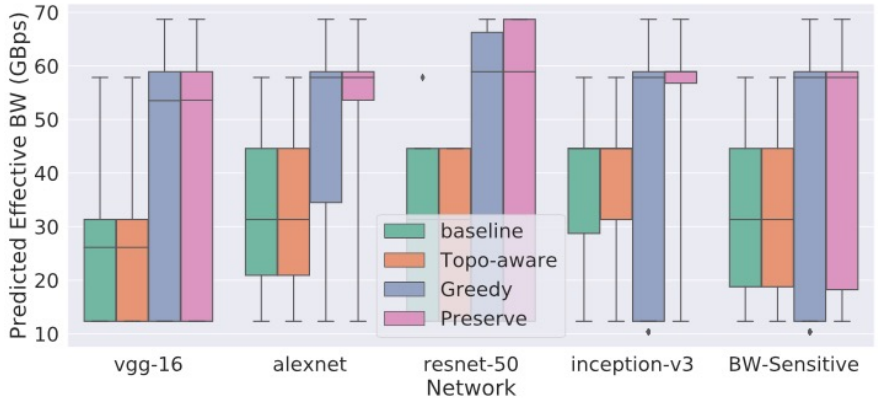
Evaluation: Benefits of MAPA



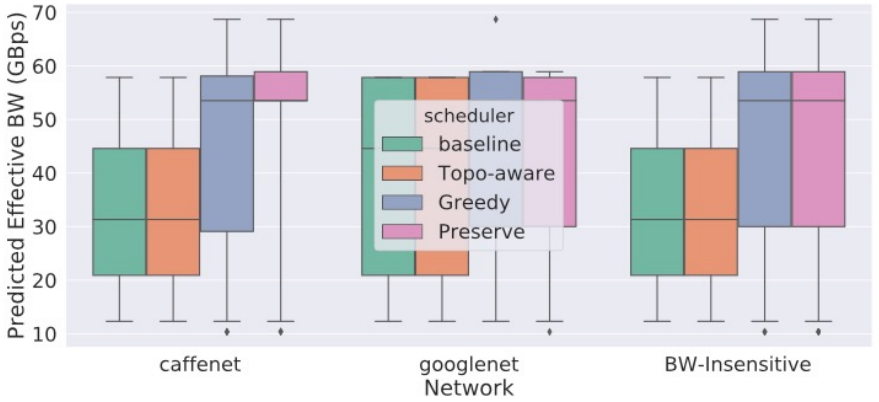
(a) Execution time of bandwidth sensitive jobs



(b) Execution time of bandwidth insensitive jobs



(c) Effective bandwidth of bandwidth sensitive jobs



(d) Effective bandwidth of bandwidth insensitive jobs

Summary of Evaluation

Policy	MIN	25th %	50th %	75th %	MAX	Tput
Baseline	1.000	1.000	1.000	1.000	1.000	1.00
Topo-aware	1.002	1.029	1.385	1.014	1.075	1.07
Greedy	0.997	**1.059**	**1.519**	1.048	1.319	1.08
Preservation	**1.006**	1.057	1.119	**1.124**	**1.352**	**1.12**

Read 1.12 and 1.35 as 12% and 35% better respectively.

Conclusion

- Fragmentation is inevitable.

Conclusion

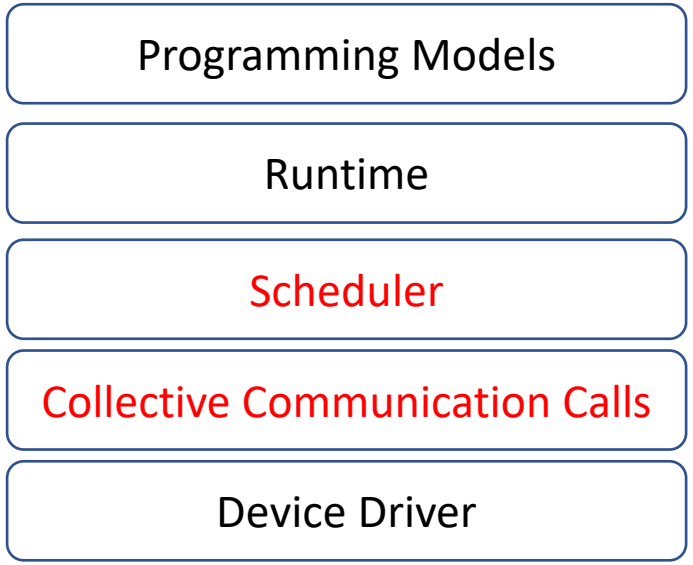
- Fragmentation is inevitable.
- Efficient scheduling can help mitigate fragmentation.

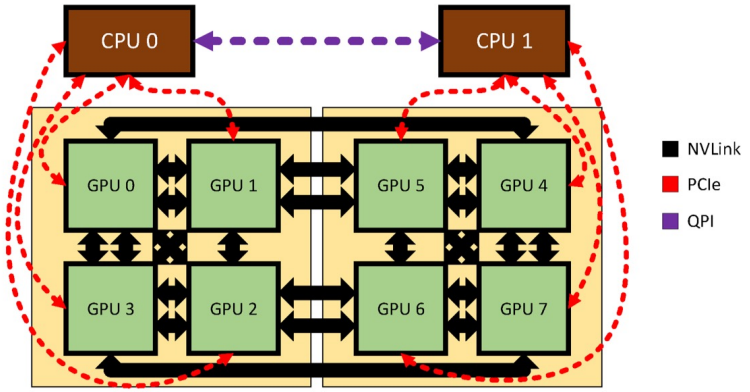
Conclusion

- Fragmentation is inevitable.
- Efficient scheduling can help mitigate fragmentation.
- Use WOTIR to alleviate effects of unavoidable bad allocations.

Conclusion

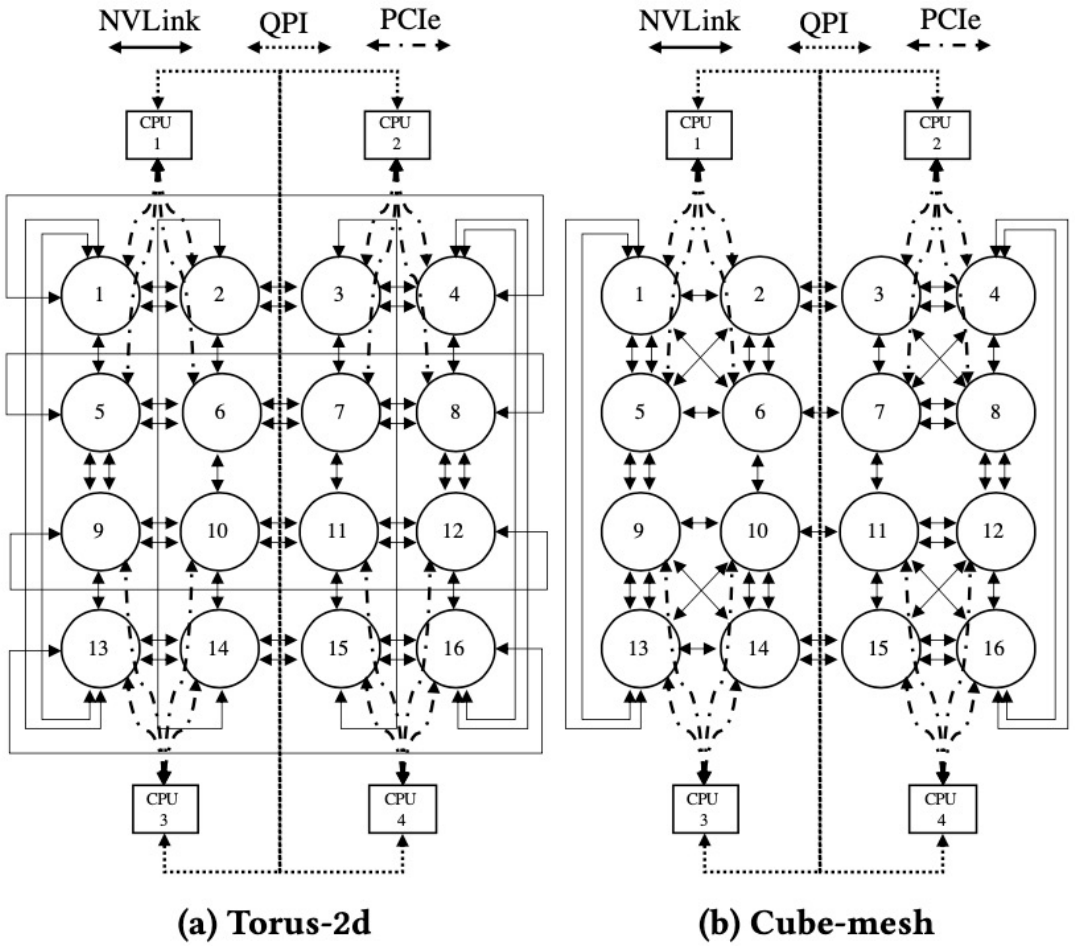
- Fragmentation is inevitable.
- Efficient scheduling can help mitigate fragmentation.
- Use WOTIR to alleviate effects of unavoidable bad allocations.
- Existing scheduling algorithms may not work with multi-accelerator constraints.





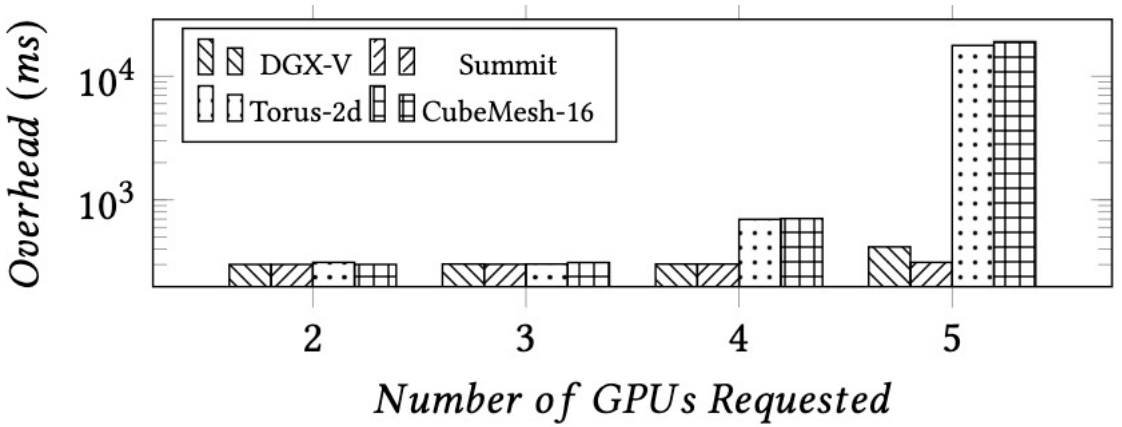
Conclusion

- Fragmentation is inevitable.
- Efficient scheduling can help mitigate fragmentation.
- Use WOTIR to alleviate effects of unavoidable bad allocations.
- Existing scheduling algorithms may not work with multi-accelerator constraints.
- Propose novel hardware topologies to assist schedulers.



Ongoing/Other work

- MAPA overhead
  - Approximate graph matching
  - Probabilistic graph mining
- Scale Scheduling policies
  - Summit
    - 9,216 POWER9 22-core CPUs
    - 27,648 NVIDIA Tesla V100 GPUs



Ongoing/Other work

- MAPA overhead
  - Approximate graph matching
  - Probabilistic graph mining
- Runtime optimization (Summer'19)
  - Effective kernel placement decisions at runtime.
- Optimizing Programming Model (Summer'20)
  - Programming Models to enable and simplify multi-accelerator programming.
    - Multiple types of accelerators (CPUs, GPUs, TPUs, etc.)