SE3331-1 (2023 Fall)

# Distributed Computing frameworks
# The case of distributed training

Xingda Wei, Yubin Xia

IPADS, Shanghai Jiao Tong University

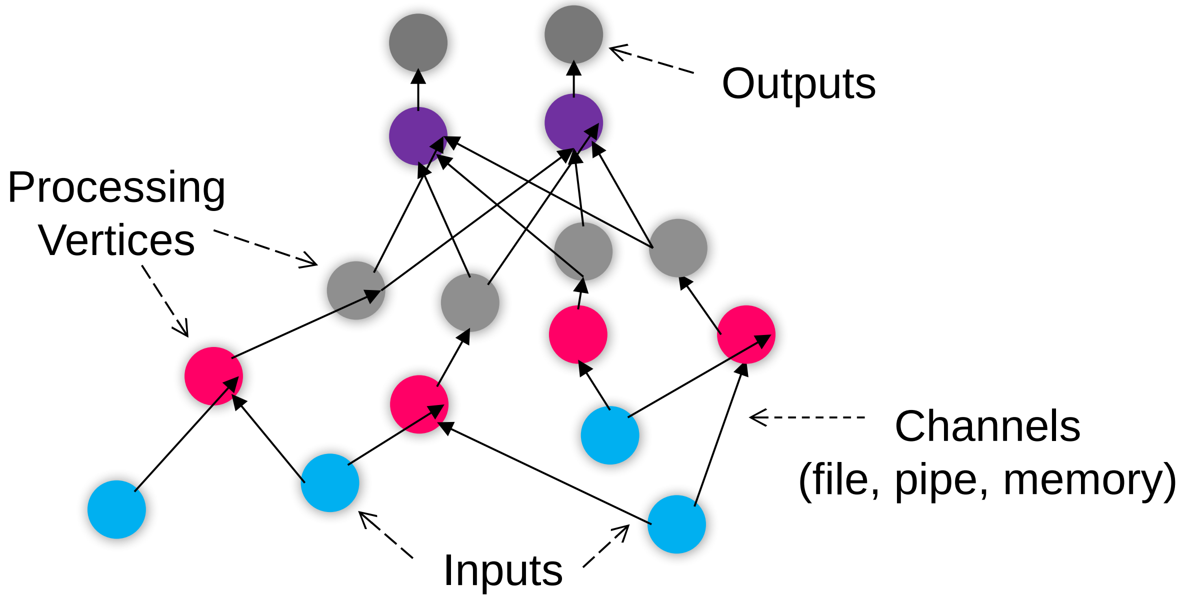
https://www.sjtu.edu.cn

Credits: AI-sys Sp22, CMU15-418

## Review: The computation graph abstraction

**Computations are expressed as a graph (Directly acrylic graph)**

- Vertices are computations (or data)
- Edges are communication channels
- Each vertex has several input and output edges



## Dryad proposes & uses DAG as the distributed computation abstraction

**Created by Microsoft (EuroSys'07)**

– Authors: Michael Isard, Andrew Birrell, et al.

**Similar goals as MapReduce**

– A **general-purpose** distributed execution engine for coarse-grain data-parallel applications

– Automatic management of scheduling, distribution, and fault tolerance

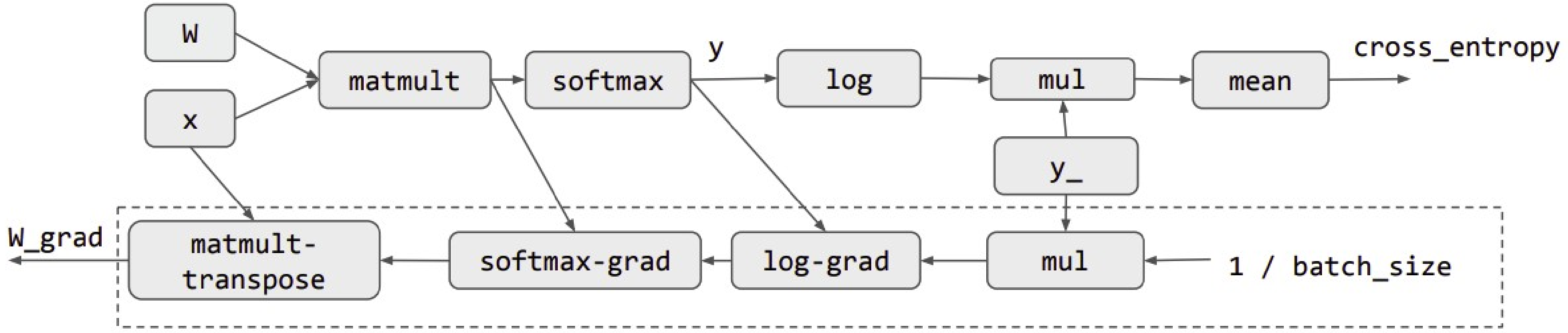
**But needs application-specific semantic to split the nodes**

– Otherwise, the graph fallback to a chain, so there is no parallelism

– A little harder than MapReduce, but also hides the distributed execution details

## Review: computation graph as the language of AI

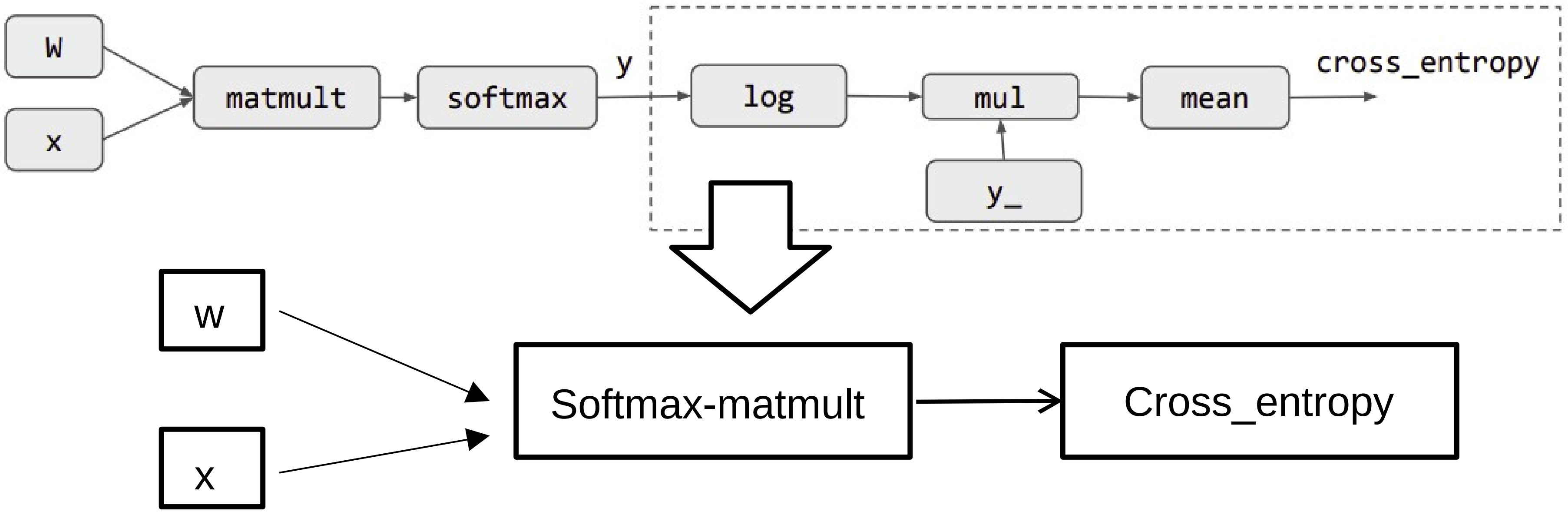
```
4   # Create the model
5   x = tf.placeholder(tf.float32, [None, 784])
6   W = tf.Variable(tf.zeros([784, 10]))
7   y = tf.nn.softmax(tf.matmul(x, W))

15  W_grad = tf.gradients(cross_entropy, [W])[0]
```



## One thing to note: graph fusion

**We can fuse multiple graph nodes to form a larger node**

- Benefit: reduce communications
  - Both internal (within a device) & cross-node (distributed)



## Distributed training

## Review: why distributed training? Device doesn’t scale

**A device has limited physical capacity to store “cores” (chip size)**

- Our cores are generalized, e.g., can either be CPU cores, GPU cores (cores w/o cache coherence + many SIMD ALUs, etc.), domain-specific cores

**Why? Recall our previous calculation**

- Basically, a A100 needs 30 seconds for an A100 GPU to finish an iteration on a single input (a.k.a, token) in the optimal case
- How many tokens are trained? 13T tokens! [1]
- To use one A100 to train GPT-4, we need about 412 years to finish the training

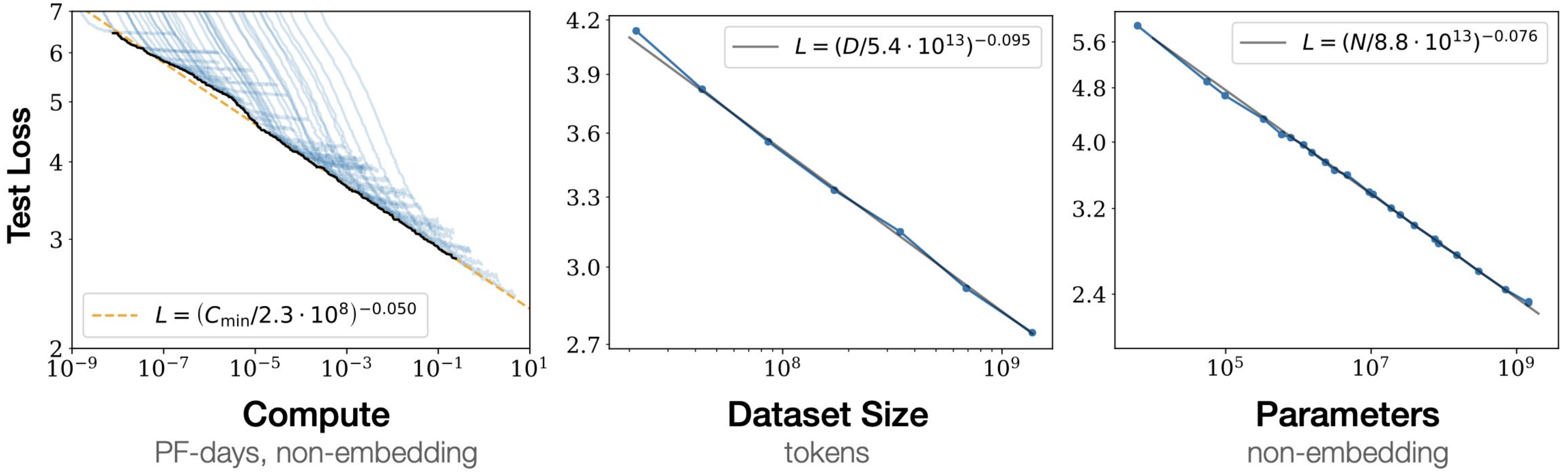
[1] https://the-decoder.com/gpt-4-has-a-trillion-parameters/

## Review: why distributed training? Scaling law

Massive data needed (the larger, the better)

Massive model needed (the larger, the better)

Massive computation power needed



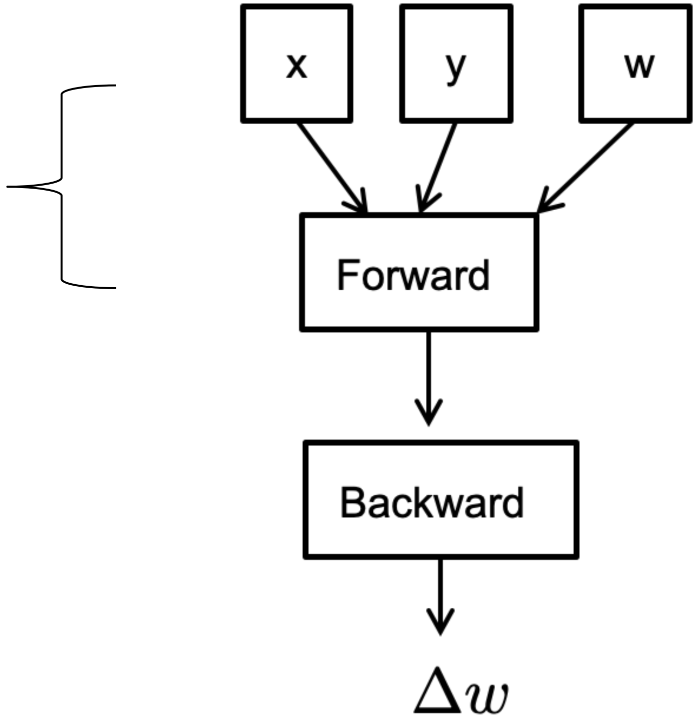
## Review: Ideal Metric of Success for Efficient Training

$$\left(\frac{\text{“Learning”}}{\text{Second}}\right) = \underbrace{\left(\frac{\text{“Learning”}}{\text{Record}}\right)}_{\substack{\textit{Convergence}\\ \textbf{Machine Learning}\\ \text{Property}}} \times \underbrace{\left(\frac{\text{Record}}{\text{Second}}\right)}_{\substack{\textit{Throughput}\\ \textbf{System}\\ \text{Property}}}$$

## Pseudocode for Stochastic Gradient Descent

```
// execute on a master
for iter in num_iters:

        iter+1 iter
```

iter



## Review: Parallelization Opportunities

***Data Parallelism:** Distribute the processing of data to multiple PEs.*

***Model Parallelism:** Break the model and distribute processing of every layer to multiple PEs*

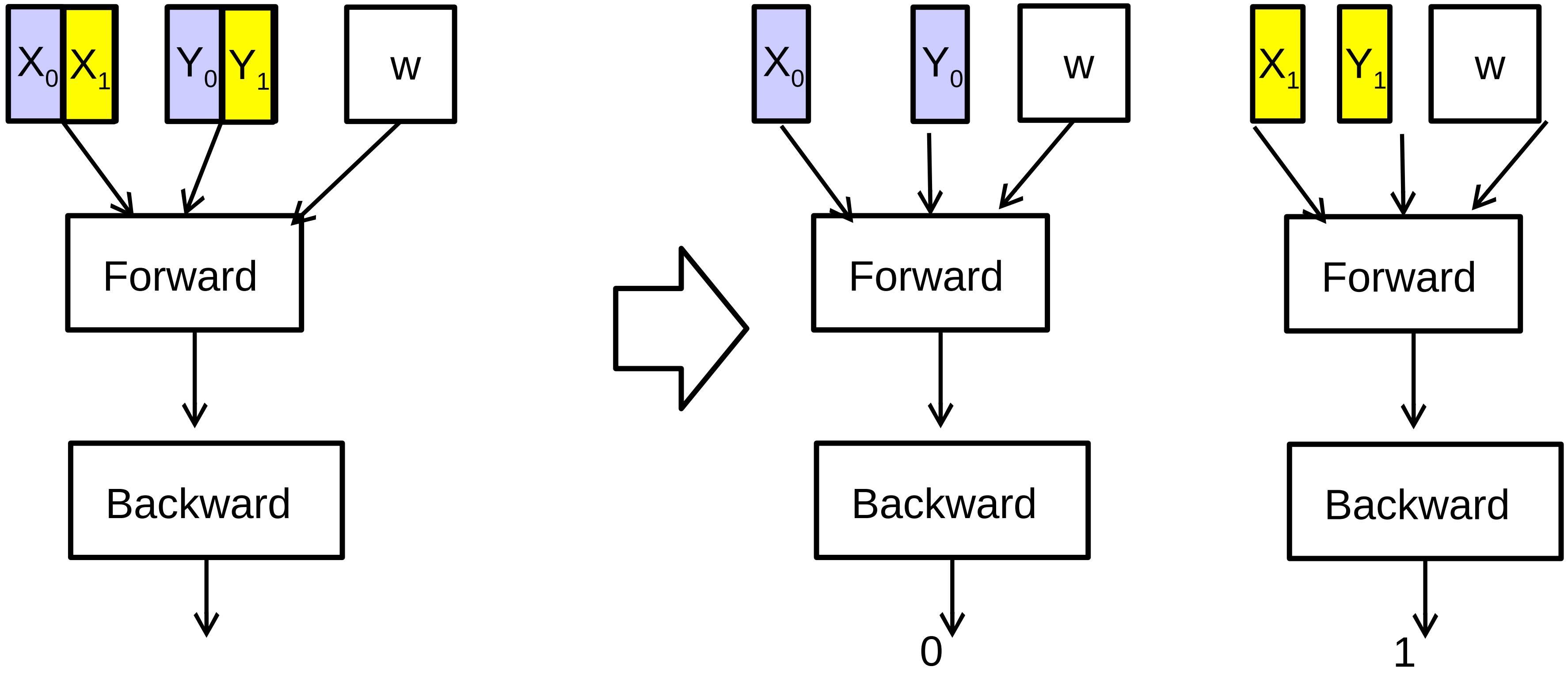
*For either approach it is also possible to use **synchronous** or **asynchronous** updates*

## (Sync) data parallel

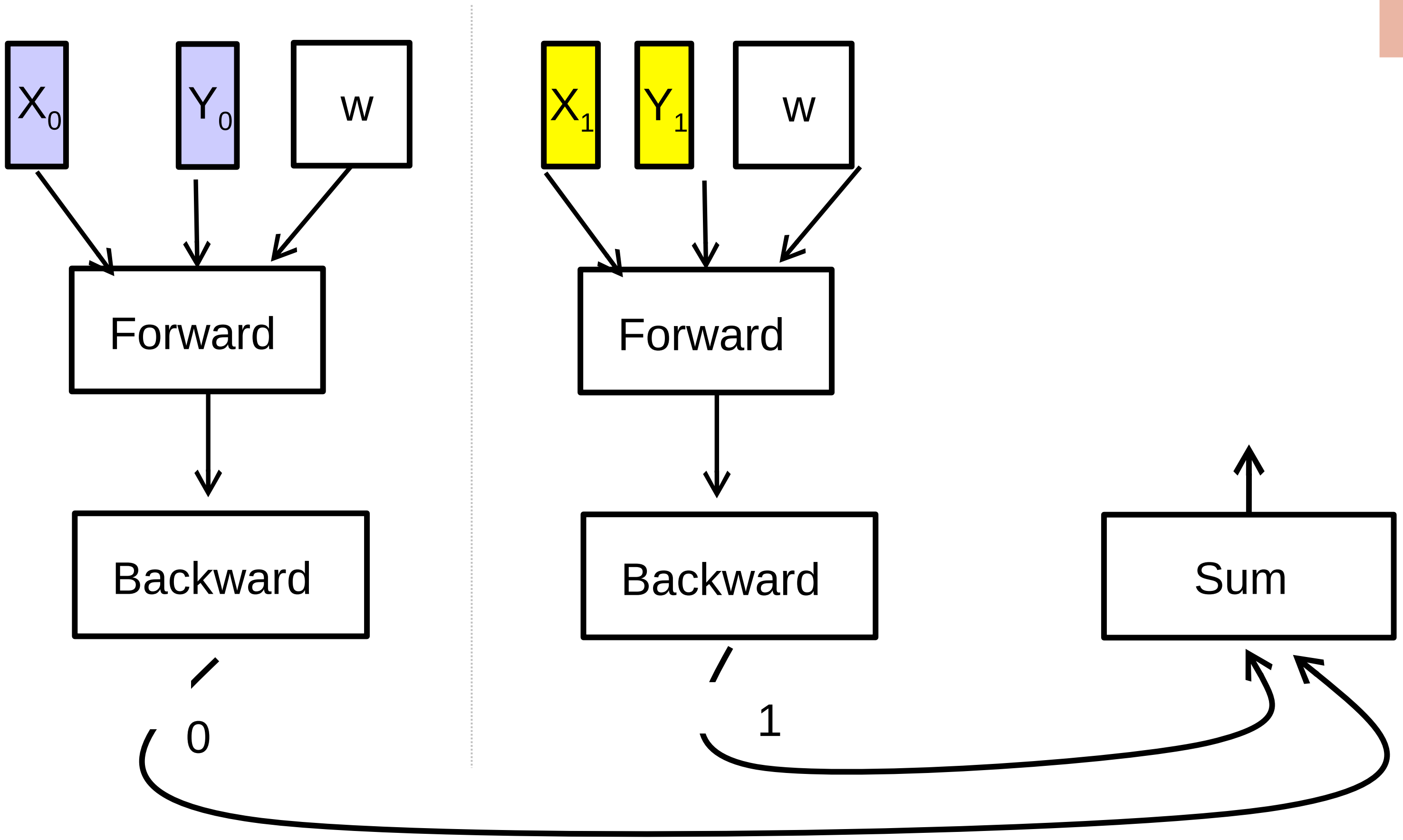
## Synchronous Data Parallelism

**Parallelize on the x (and y)**

– Observation: the sum operator can be easily parallelized



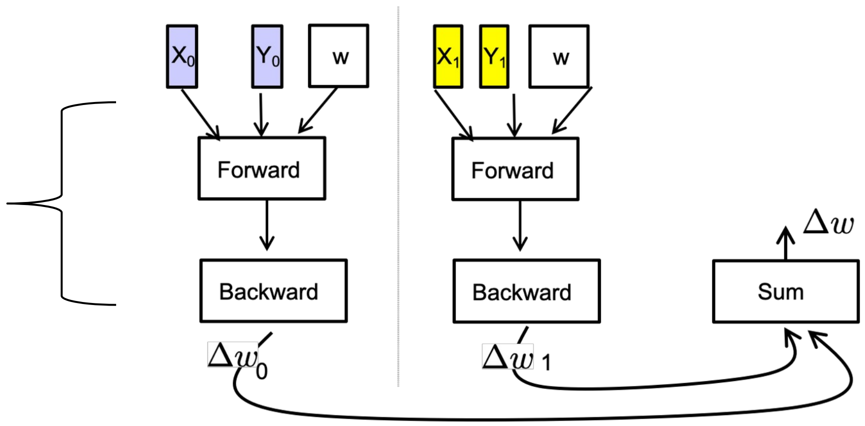
## Synchronous Data Parallelism



## Pseudocode for (iterative) data parallel in action

```
// execute on a master
for iter in num_iters:

        iter+1 iter
```

iter



**The master will do the coordination**

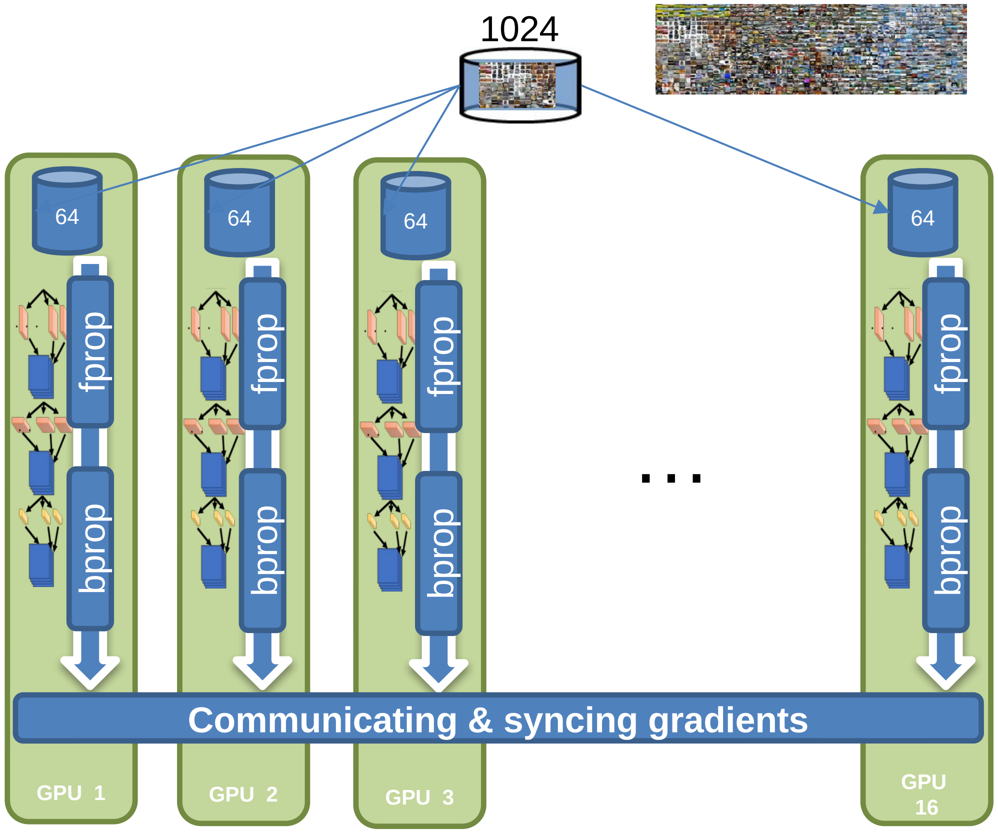
- Similar to the Job manager in Dryad

## Synchronous Data Parallelism

**Store the entire model (W) on each processor**

- Then distribute the batch evenly across each processor
- E.g., 64 images per GPU

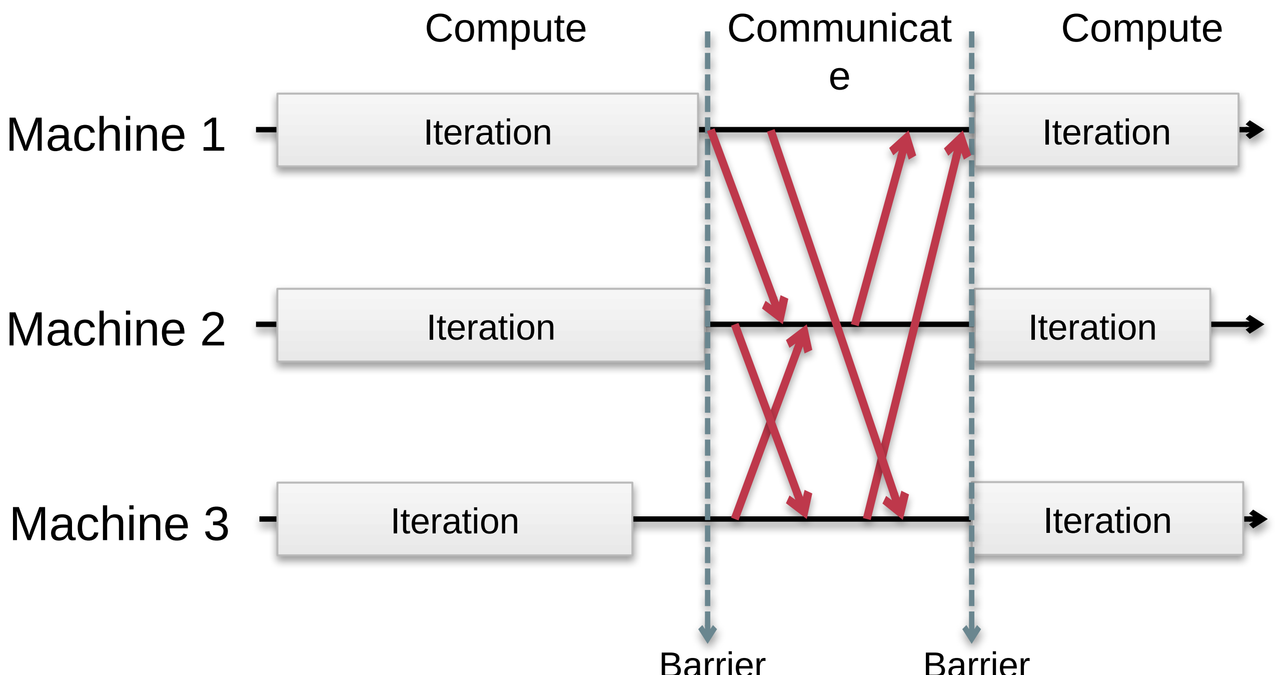
**Question: how to coordinate between iterations?**



## How to parallelize the partitioned graphs? BSP

**Bulk Synchronous Parallel (BSP) Execution**

- Using a strict barrier to coordinator processes in a distributed computing
- Barrier can be implemented in a centralized way (master) or decentralized



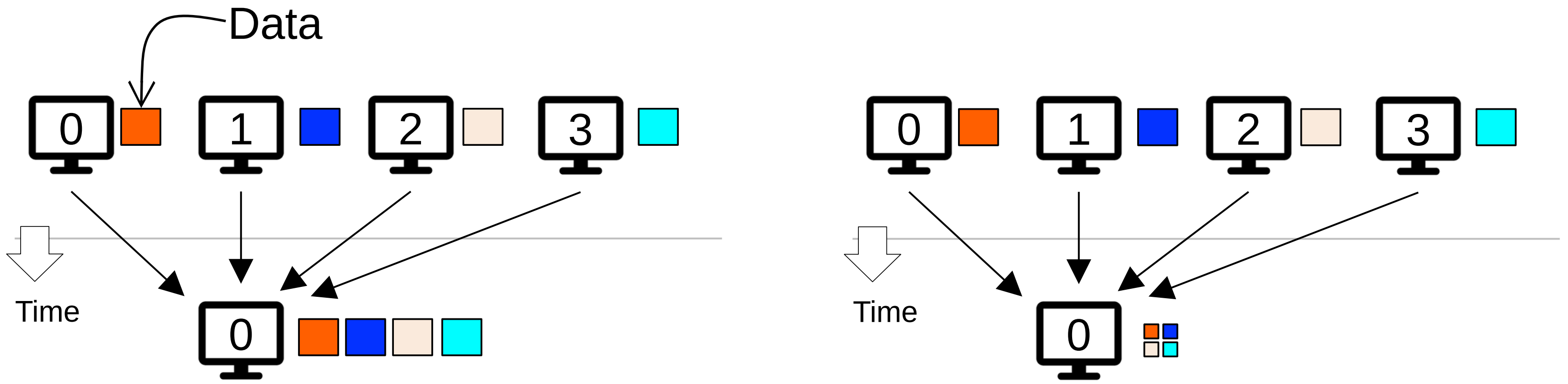
## Communication primitives in distributed computing

**Gather:** Collect data from all the other processes

– Example: [1][2][3][4] -> (gather) -> [1,2,3,4]

**Reduce:** Gather and aggregate a piece of data from all the other processes

– Example: [1][2][3][4] -> (reduce) -> [1+2+3+4] = [10]



**Gather vs. Reduce**

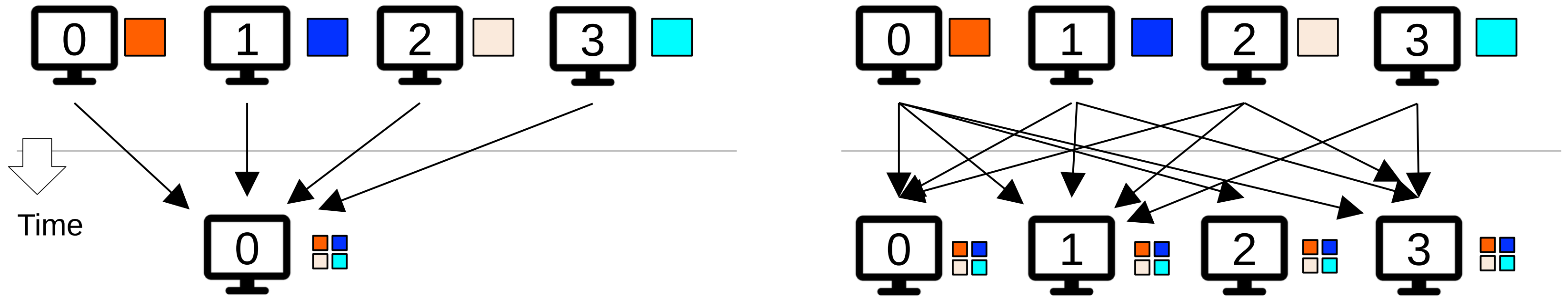
## Communication primitives in distributed computing

**Reduce:** Gather and aggregate a piece of data from all the other processes

- Example: [1][2][3][4] -> (reduce) -> [1+2+3+4] = [10]

**Allreduce = reduce on all processes**

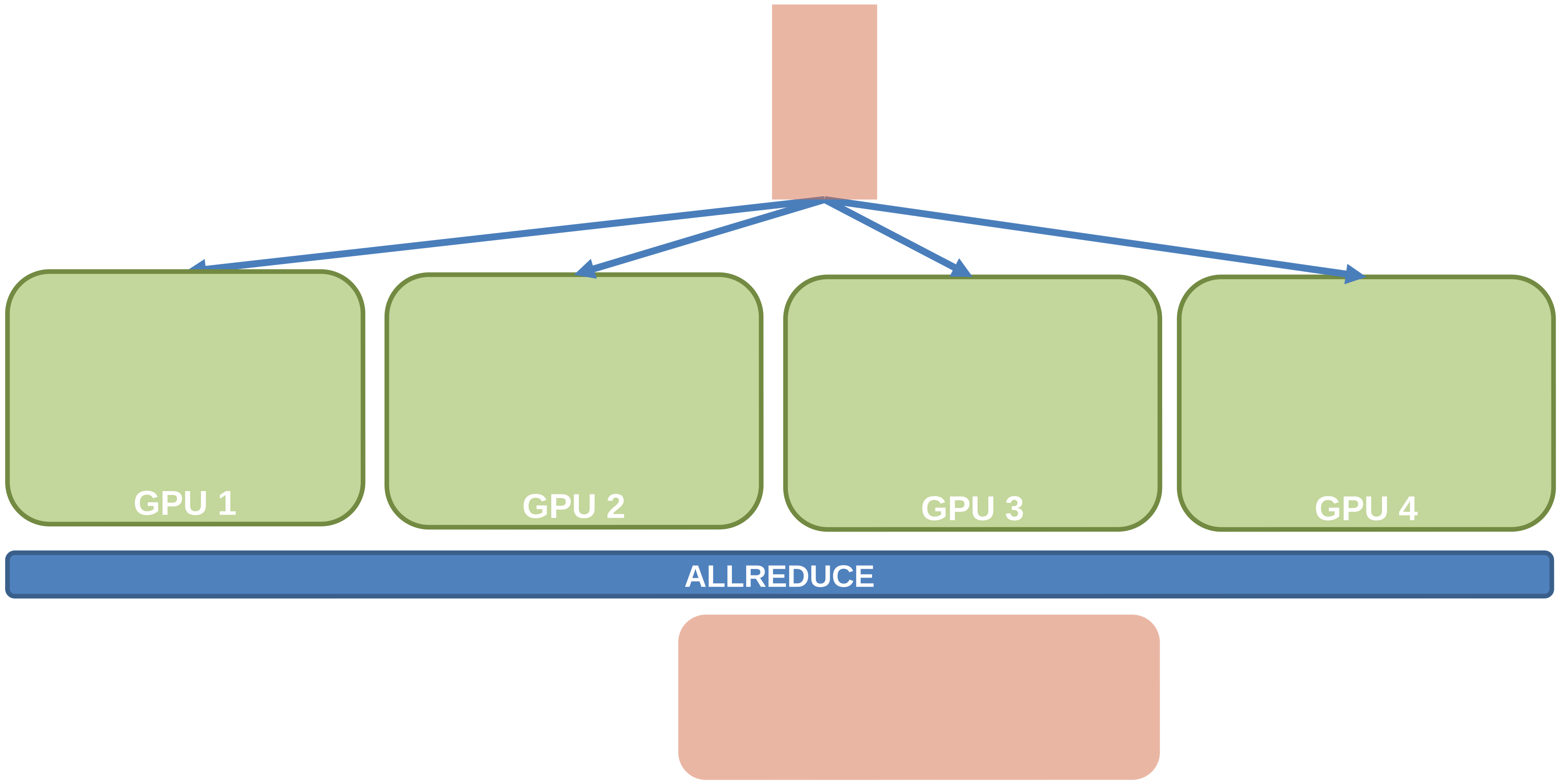
**Question:** how to efficiently implement allreduce?



**Reduce vs. AllReduce**

* The messages of allreduce may be missing due to space limitations

## Key communication operator: allreduce



## System requirements for AllReduce

**Overhead analysis: computation + network**

**The overhead of computation is typically small**

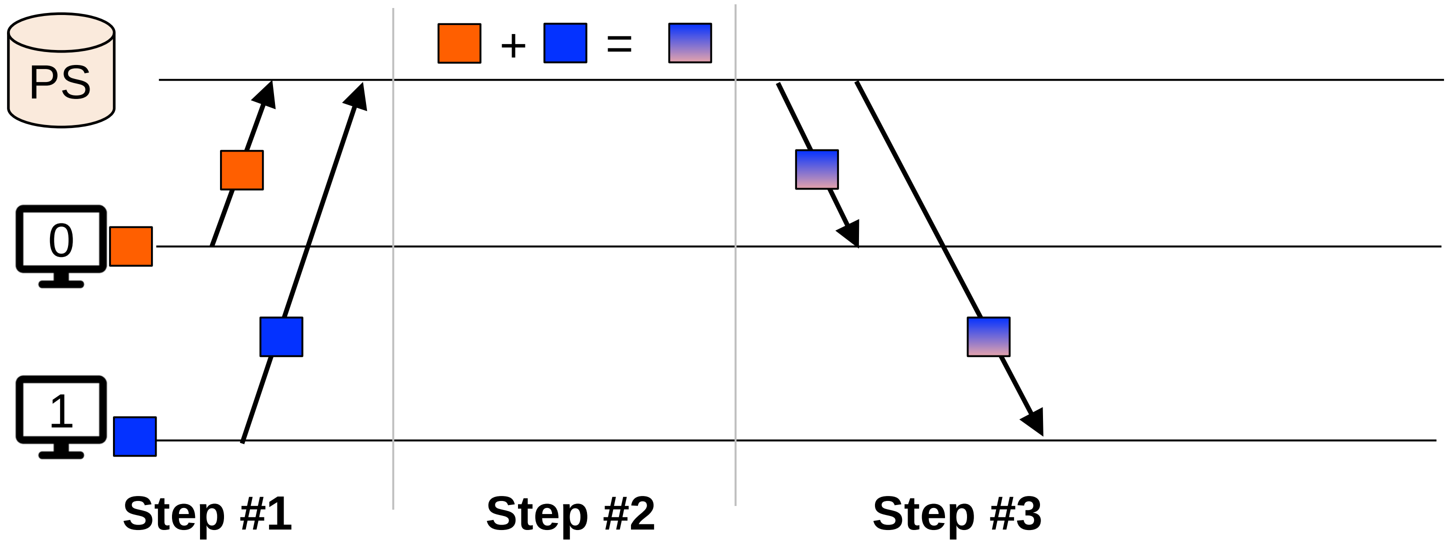
- Compute a sum operation on device is highly optimized, e.g., SIMD, GPU, TPU, etc.

**Goal: reduce network overhead**

- Reduce bytes sent per-message
- Reduce concurrent messages sent to a server

## Try #1: parameter server

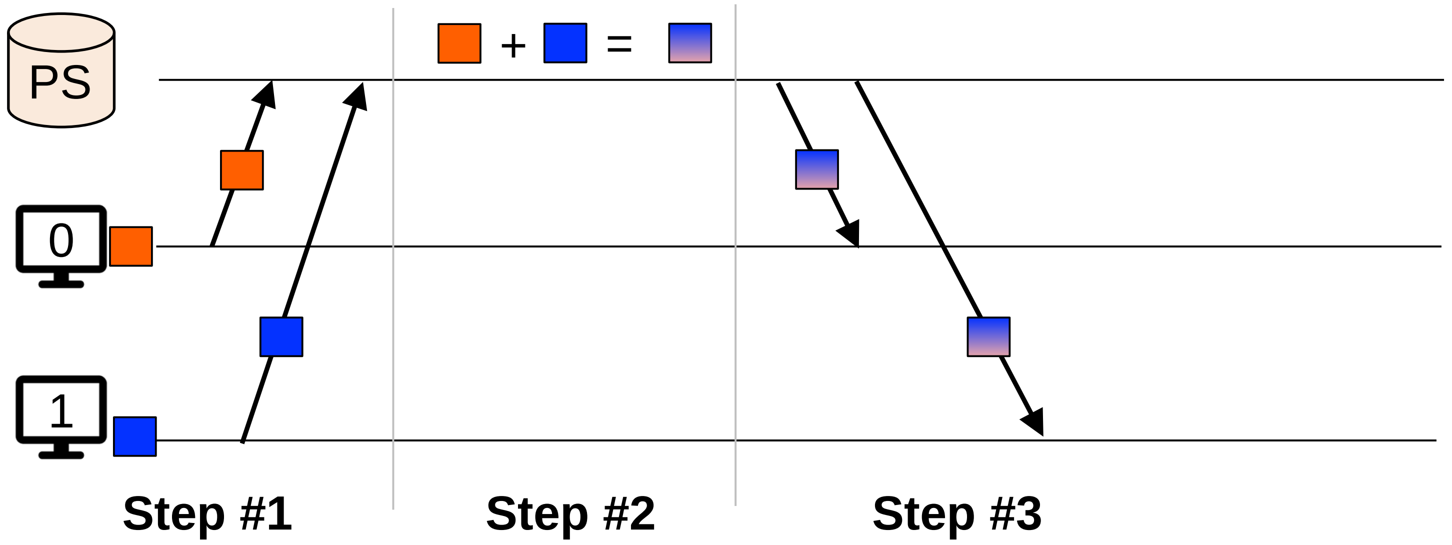
1. Each process pushes the data to the parameter server (PS)
2. After receiving all data, aggregate the data at the PS
3. The PS pushes the data back to the processes



## Allreduce with a single parameter server (PS)

**Question**: what is the network requirements at each stage?

- N: the size of the parameters; P: the number of processors
- Step #1: O(N) at each process, O(N * P) at the PS
- Step #3: O(N) at each process, O(n * P) at the PS



## Allreduce with a single parameter server (PS)

**Problem: huge bandwidth requirement at the PS (O(P * N))**

- As well as huge contention at the master machine

**Example: training ResNET50 on V100**

- ResNET50: 24.4 M params ~= 97.5MB storage
- On V100, each node can train 3 iterations / second (forward + backward path), w/o communication
- #Processing Node: 256
- Total requirement bandwidth: 256 * 3 * 97.5 = 73.1GB/s

**The state-of-the-art NIC: 400Gbps RDMA ~= 50GB/s**

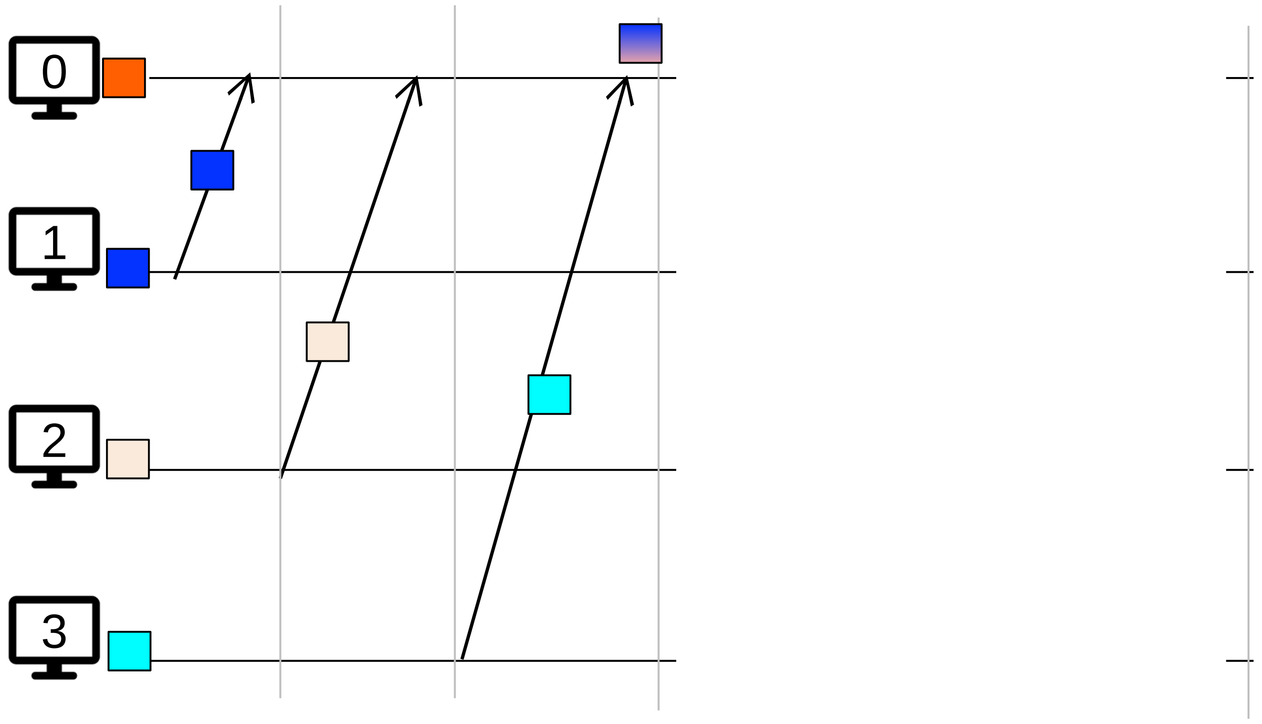
- The network would become the bottleneck!

**Goal**: reduce messages send to a centralized server

## Try #2: De-centralized approach for allreduce

**Each node finishes its allreduce, one-by-one**

– E.g., with a careful designed scheduler



## Analysis of the naïve decentralized reduce

**N: total parameters, P: # processors**

**Total communication per-machine**

- N * (P – 1) data (Bad …)
- O(1) fan-in (Good!)

**Total communication rounds: O(P * P)**

- Much higher than the parameter server approach (O(1))

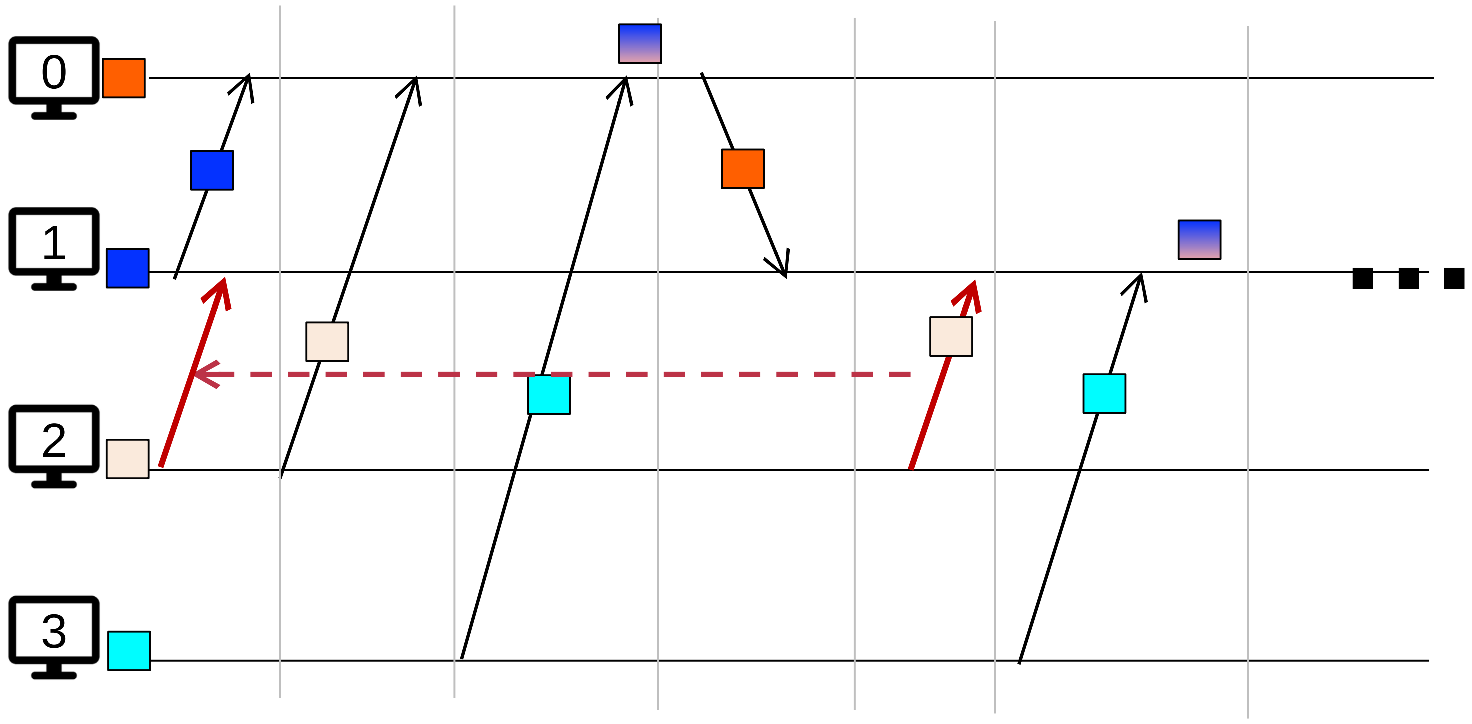
**Question**

- Can we do better?

# Try#2: De-centralized approach for allreduce

**Each node finishes its allreduce, one-by-one**

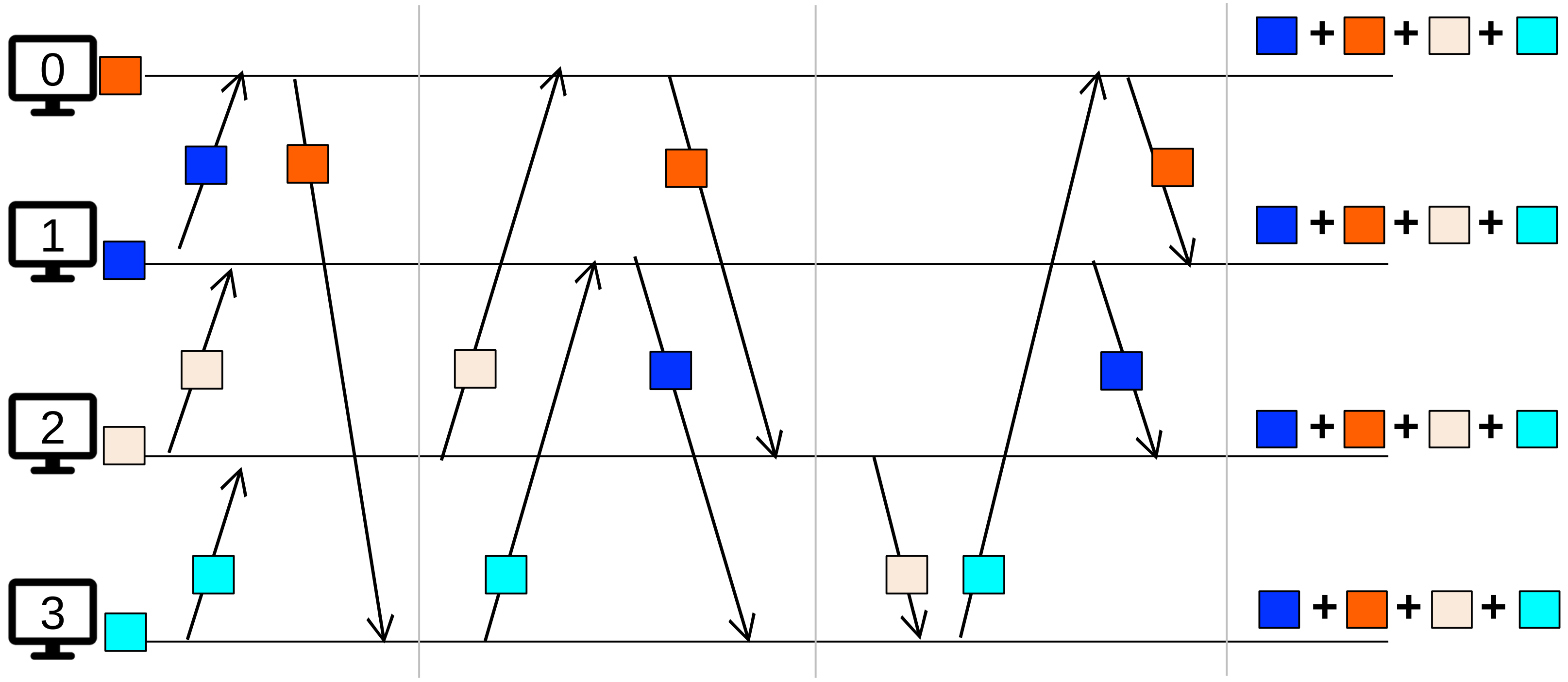
- Observation: communications from future rounds can be moved earlier if w/o contention



## De-centralized approach for allreduce

**Each node finishes its allreduce, one-by-one**

– Communications from future rounds can be moved earlier if w/o contention



## Analysis of the naïve decentralized reduce

**N: total parameters, P: # processors**

**Total communication per-machine**

- N * (P – 1) data (Bad …)
- O(1) fan-in (Good!)

**Total communication rounds: ~~O(P * P)~~ =》 O(P)**

- Higher than the parameter server approach (O(1)) , but w/o contention

**Question**

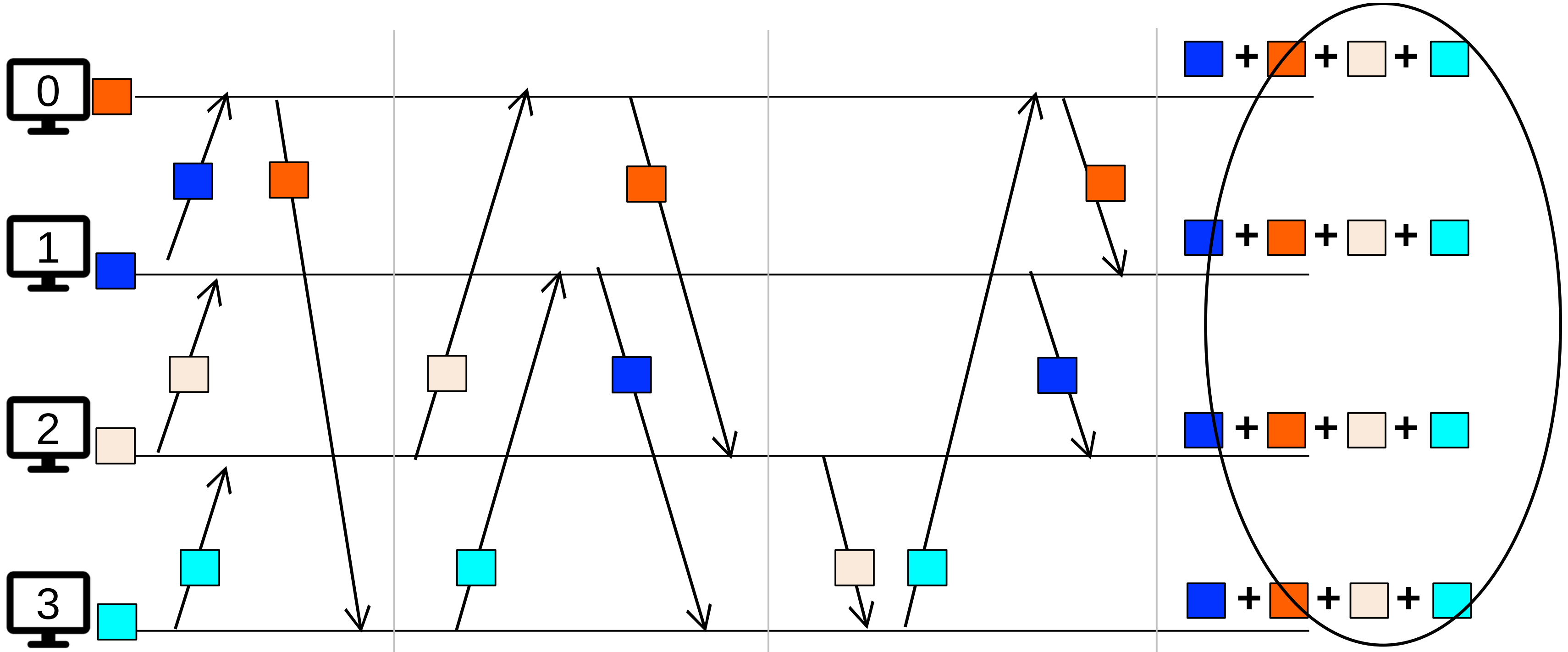
- Can we do better?

**Goal**: reduce the payload sent per-process

## Idea: partition the data to reduce traffic

**Each node finishes its (a partition of its) allreduce, one-by-one**

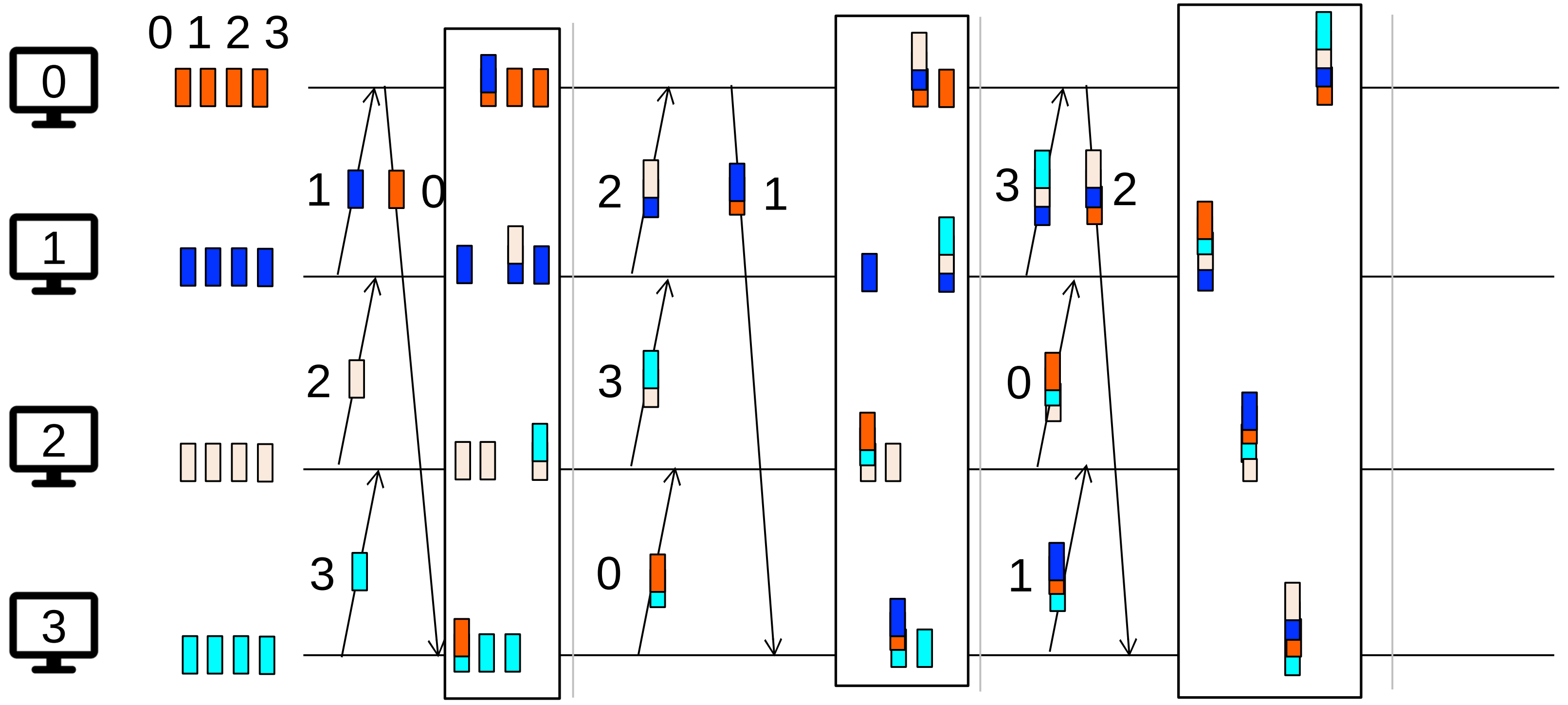
– Finally, using a decentralized approach to gather all the data



## Try #3: Ring allreduce

**Decentralized reduce, each process reduce on a partitioned data (prev/next to it)**

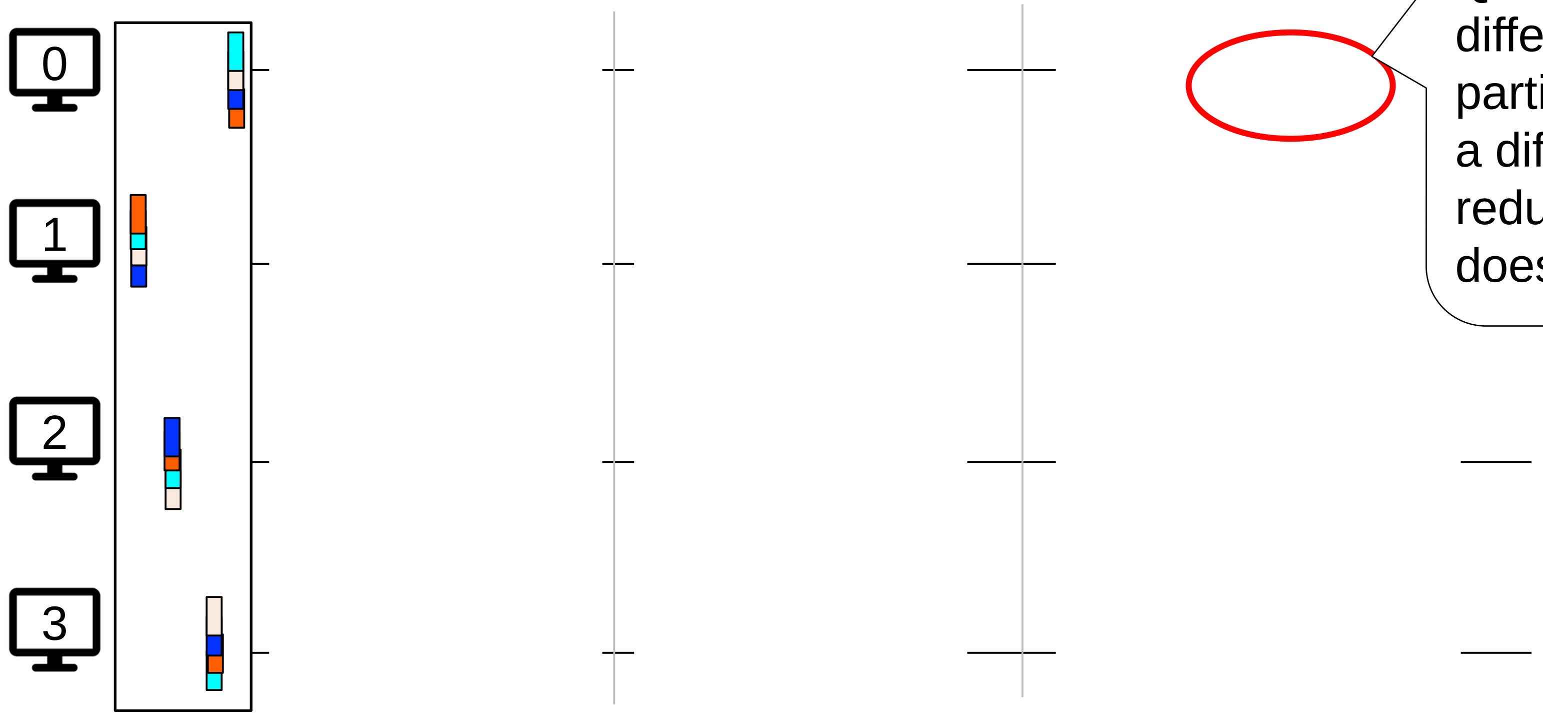
– i.e., each process send one partition, compute it, & then sends to the next process



## Ring allreduce

**After the individual reduce, each process can scatter to collect results**

– Eventually, all the processes will have the same reduced data



**Question**: different partition has a different reduce order, does it ok?

## Why ring allreduce can work?

**Assumption: reduce op is associative & communicative**

- So we can reorder them to reduce sudden loads at a node
- E.g., we don’t need to collect all the data to do the reduce

**Otherwise, we cannot reorder the computation**

## Analysis of the communication cost of ring allreduce

**N: total parameters, P: # processors**

**Total communication per-machine**

- 2 * (P-1) * N / P: the same as partitioned
- O(1) fan-in

**Total communication rounds: O(P)**

- Much higher than the parameter server approach (O(1))
- A trade-off for reducing network contention due to high fan-in

**What are the drawback?**

- High latency due to extra rounds

## Allreduce: put it all together

**Trade-off between rounds vs. fan-in performance**

- P: # processors/machines/GPUs participated in the computation
- N: the size of the data to be reduced

| | Round | Peak node bandwidth | Per-node fan-in |
|---|---|---|---|
| Parameter server (PS) | O(1) | O(N * P) | O(P) |
| Decentralized Allreduce | O(P) | O(N) | O(1) |
| Ring allreduce | O(2 * P) | O(N/P) | O(1) |

## More reduce methods exist

**Key design goals**

- Reduce payload sent per-node
- Reduce fan-in to avoid contention on network resource
- (new) Reduce rounds (grow linearly w/ the number of processors)
- (new) Better suits the underlying hardware (Cloud & HPC)

**Still an active research field**

- E.g., what if the hardware speed between different machines diff?
- What if the link speed changes overtime?

## Case study: NVLink + RDMA

**Setup**

- Each machine has multiple GPUs for computation
- These GPUs essentially form a distributed (processing) system

**NVLink**

- Fast interconnects that connect different GPUs

**RDMA**

- Fast networking feature that connects different machines

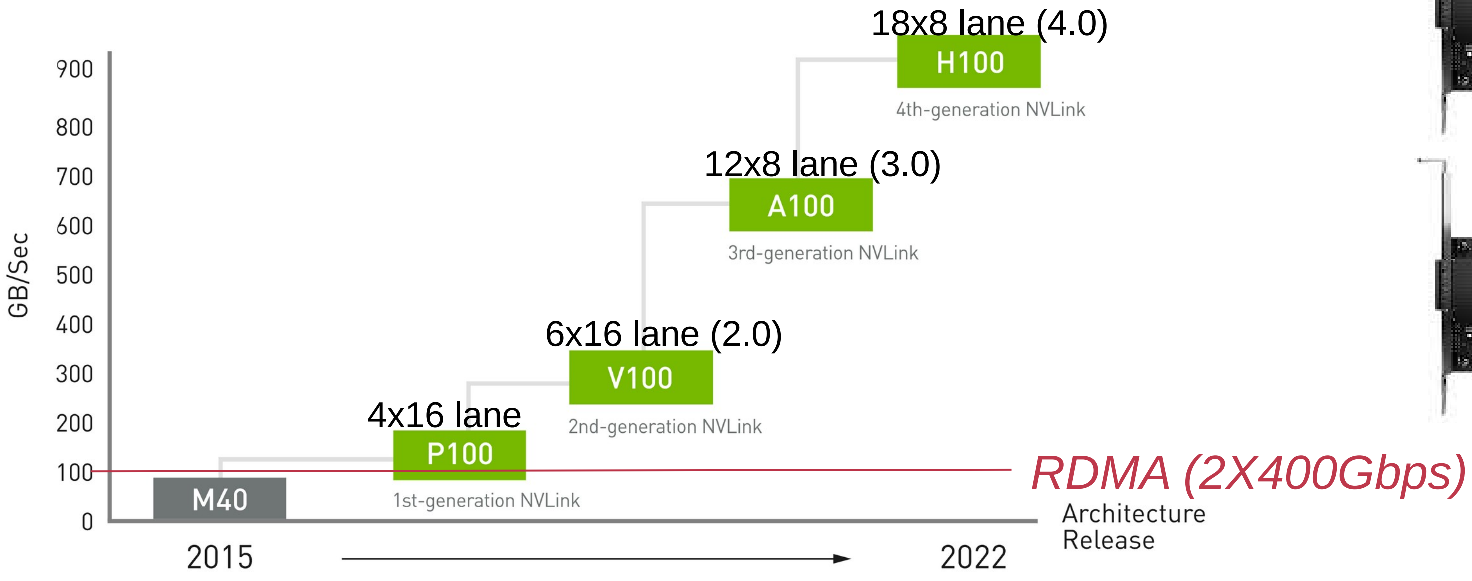
## What does NVLink Look Like?

**8-GPU on single board**

## Why is NVLink Fast?

**Massive Parallelism (thanks to the close placement of hardware)**

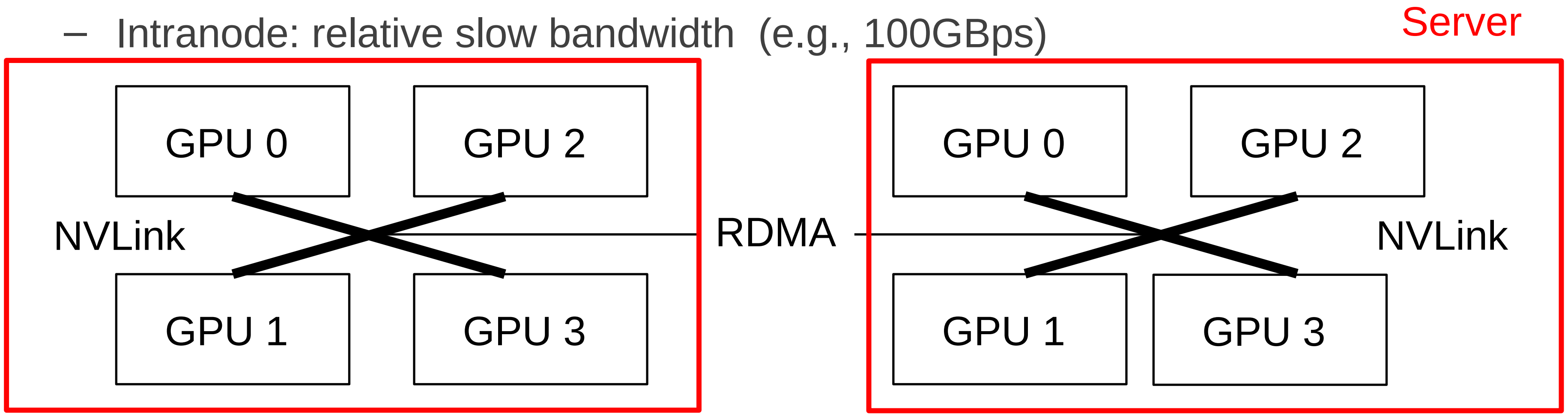
– GPUs are connected w/ *multiple lanes, w/ high frequencie*



# Communication heterogeneity

**Different nodes' link capacity differs**

- Internode: high bandwidth (e.g., 450GBps)
- Intranode: relative slow bandwidth  (e.g., 100GBps)



**Optimization opportunities**

- We can do a parameter server within a server, and do rings across servers

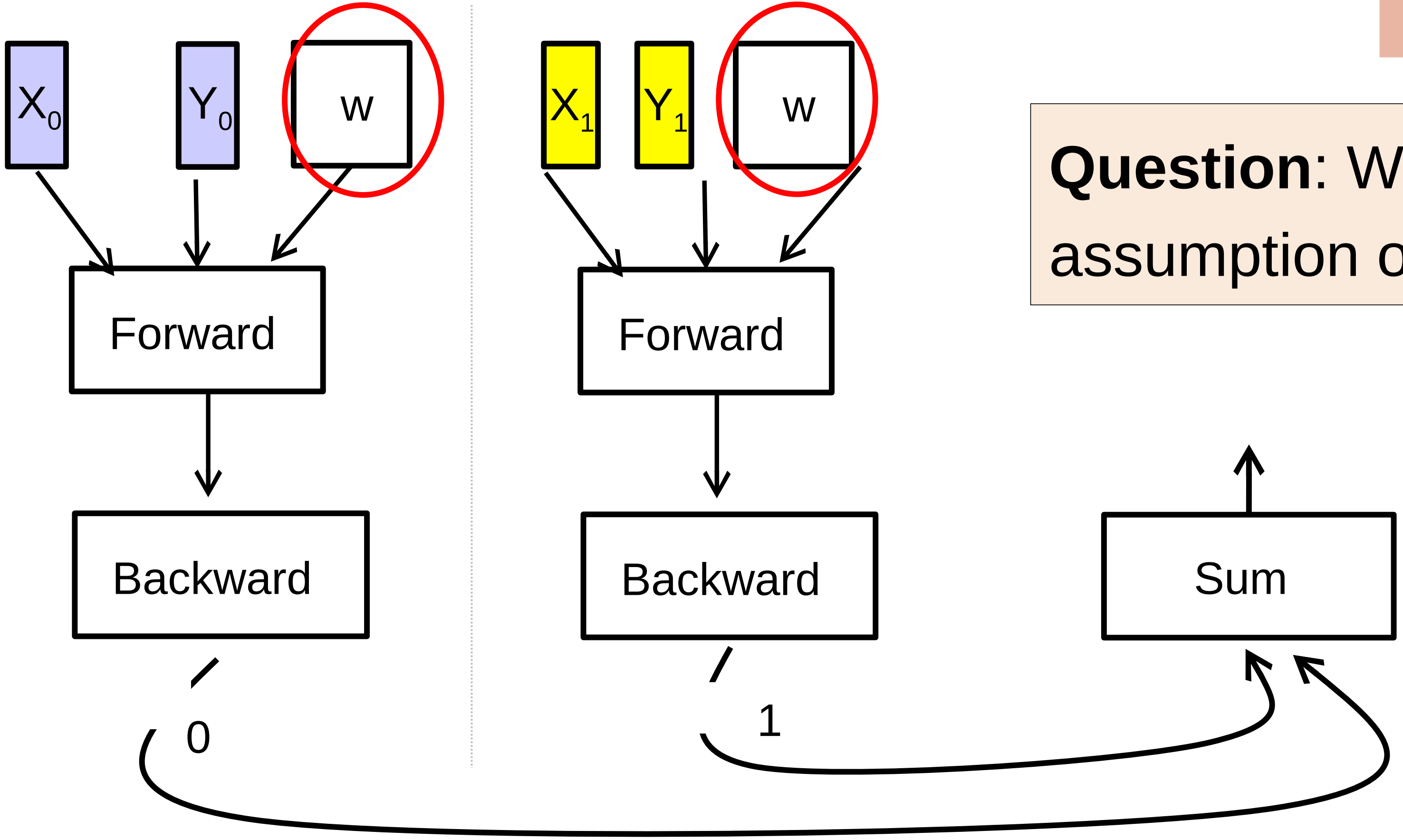
## Parallelization Opportunities

***Data Parallelism:*** *Distribute the processing of data to multiple PEs.*

***Model Parallelism:*** *Break the model and distribute processing of every layer to multiple PEs*

*For either approach it is also possible to use **synchronous** or **asynchronous** updates*

## Recall: Data Parallelism



**Question**: What is the assumption of W?

## Drawback of data parallelism: replicated model

**Data parallelism assumes the model (parameters) are replicated on all the processes**

- Such that they can do the forward & backward pass dependently
- What if the model cannot fit onto the device?

**Current trends: models are becoming larger and larger**

- No country (or company) can tolerate the risk of falling behind in the AI race

## Trend: Training Large Models

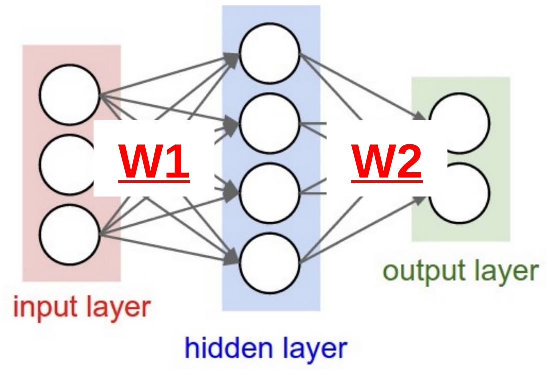
Beyond single chip memory

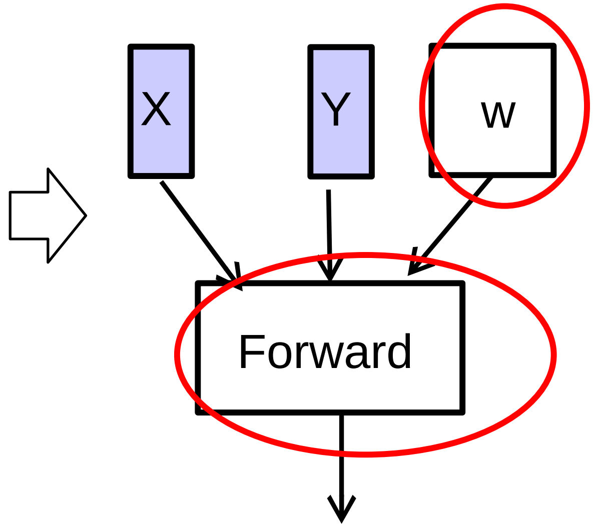
Amir Gholami, Zhewei Yao, Sehoon Kim, Michael W. Mahoney, Kurt Keutzer, AI and Memory Wall, Riselab Medium Blogpost, 2021.

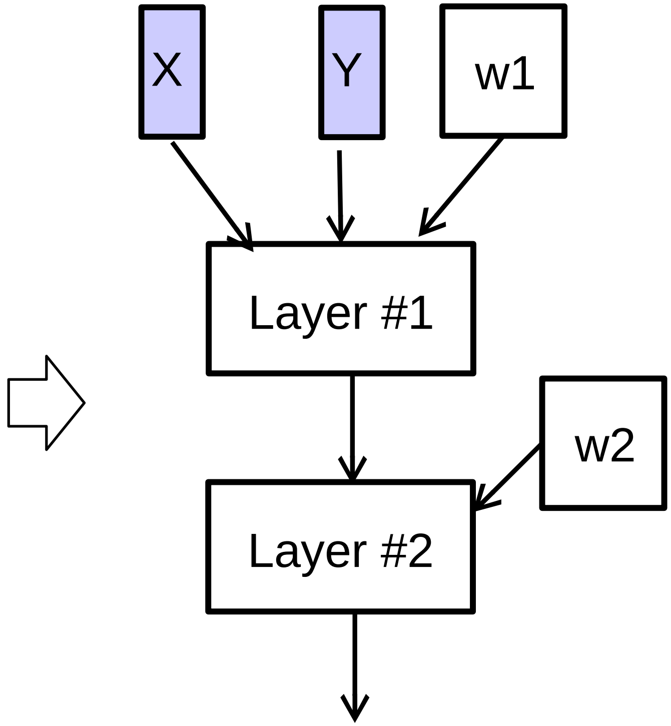
## Model parallelism

**Partition the parameters of a model, typically done in two ways:**

- **Partition on the layer**: pipeline parallelism
- **Partition on the W**: tensor parallelism



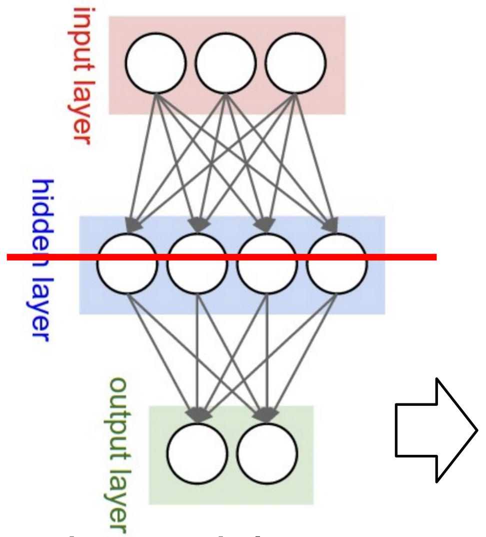




## Pipeline parallelism

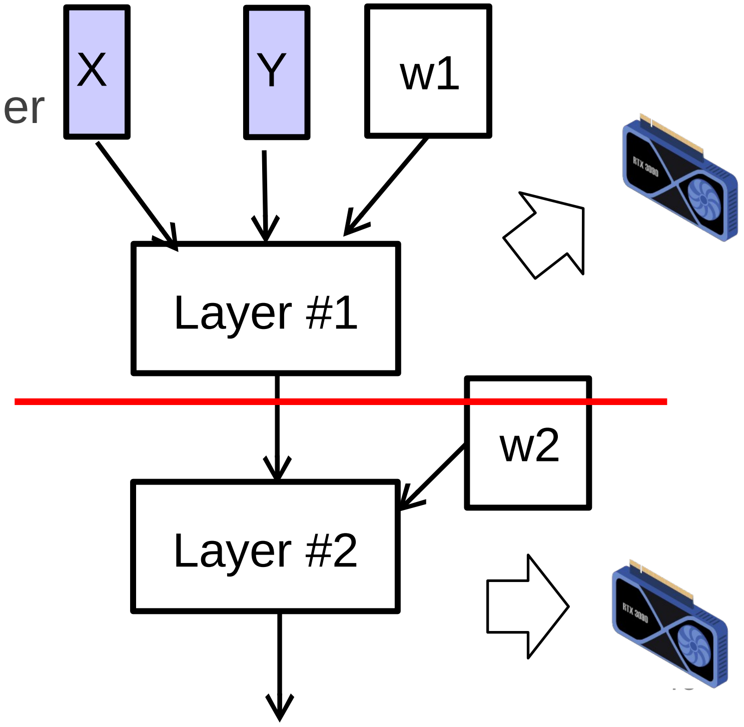
**Partition the computation layer-by-layer, each partition is deployed on a device**

- Note that different layers can be grouped together
- The strategy is out of the scope of this course

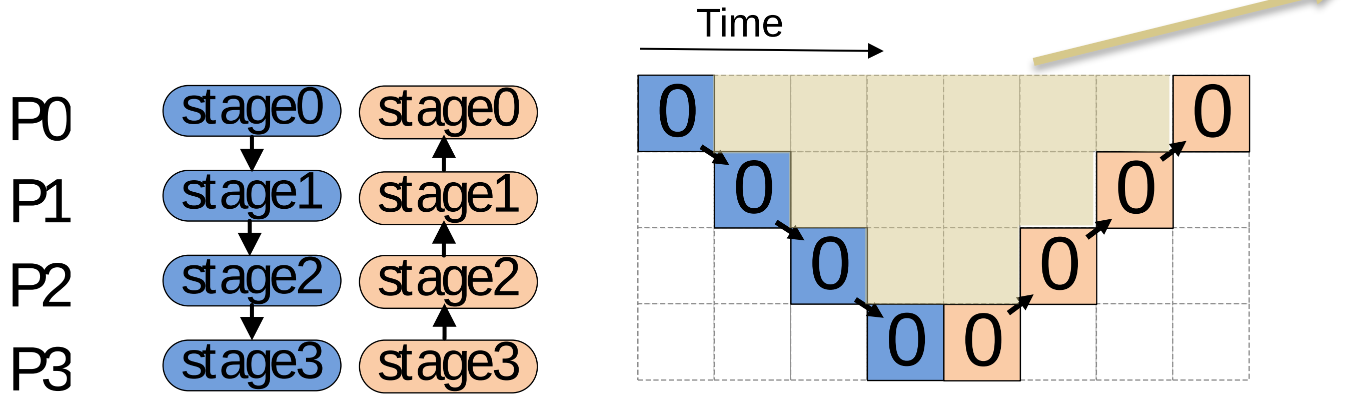


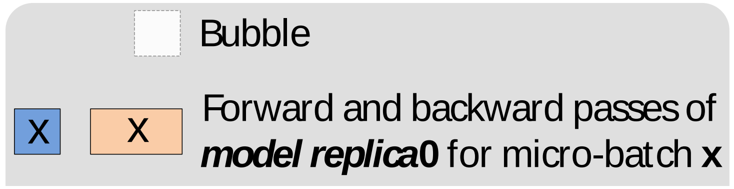
The model

The partitioned computation graph



## Pipeline parallelism




Bubble
x
x
Forward and backward passes of ***model replica*0** for micro-batch **x**

**How to reduce bubble?**

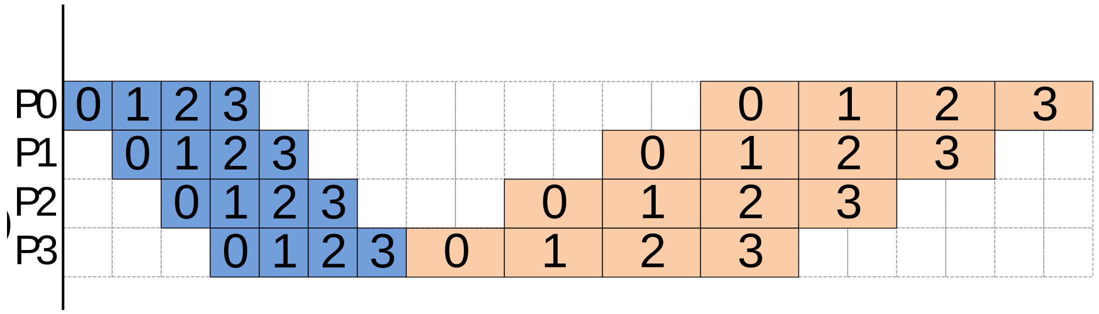
## Technique: micro-batching

**Reducing the bubble size by breaking the batch size into smaller pieces to reduce the idle time of the processes**

- Reduces bubble size in an easy way to implement

**Question: to what extent can micro-batching improves performance?**

- The bubble size depends on the #stages
- The ratio bubble overhead depends on the overall pipeline numbers



## Pipeline parallelism & the number of micro-batch

**Question: what is the bubble size ?**

- p – 1 (p == #devices)

**Question: what is the overhead?**

- : time of a forward of a micro-batch (m)
- : time of a backward of a micro-batch (m)
- Ideal process time:
- Wasted time of bubble:
- Bubble time fraction:

## Pipeline parallelism & the number of micro-batch

**Question: what is the overhead of pipeline parallelism?**

– Bubble time fraction:

**Optimization directions**

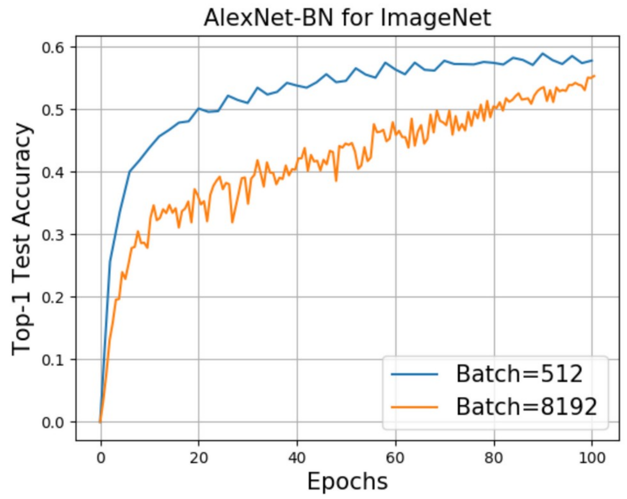
– 1. increase m (e.g., increase the batch size)

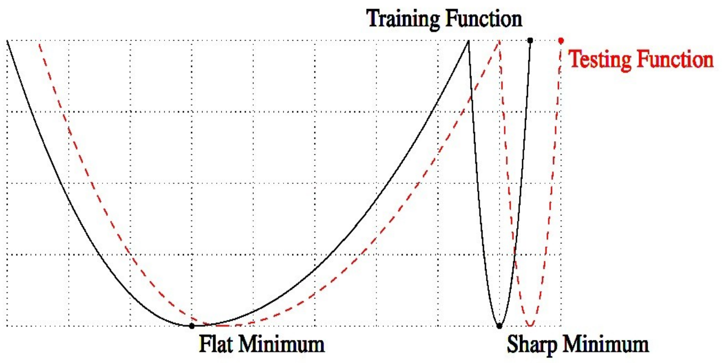
– 2. reduce p (reduce the #partitions)

## Problems with Large Batch Training

Larger Batch leads to sub-optimal generalization

A common belief is that large batch training gets attracted to “sharp minimas”





**Key takeaway:** improving system should consider application properties

Keskar et al., On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima, ICLR’16.
Z. Yao, A. Gholami, Q. Lei, K. Keutzer, M. Mahoney. Hessian-based Analysis of Large Batch Training and Robustness to Adversaries, NeurIPS’18.
Ginsburg, Boris, Igor Gitman, and Yang You. "Large Batch Training of Convolutional Networks with LARS." arXiv:1708.03888, 2018.

## Problems with large batch size

**P#0. Decreased accuracy**

**P#1. Increased memory at each device**

- Memory used for storing the activation memory

**Though some optimizations exist**

- E.g., Re-computation, throw away the intermedia activations generated during the forward path, re-compute it during the backward path
- **Cons**: trades performance for memory usage

## Can we reduce the number of pipeline stages (p)?

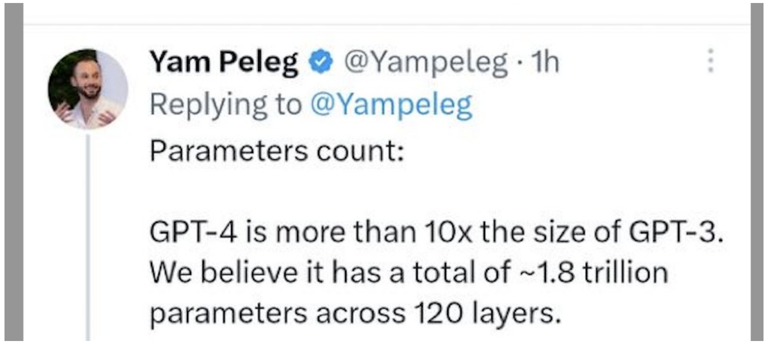
**Typically, also challenging because AI models are really really large**

**Example: GPT-4**

- Setup: 1.8 trillion parameters
- 120 layers
- Assuming 4B for each parameter, we have 60 GB per-layer

**What are the memory capacity of GPU?**

- 80GB for H100, 140GB for H200
- Note that we also need to store activations & input data ( & optimization state)


Yam Peleg @Yampeleg · 1h
Replying to @Yampeleg
Parameters count:

GPT-4 is more than 10x the size of GPT-3. We believe it has a total of ~1.8 trillion parameters across 120 layers.

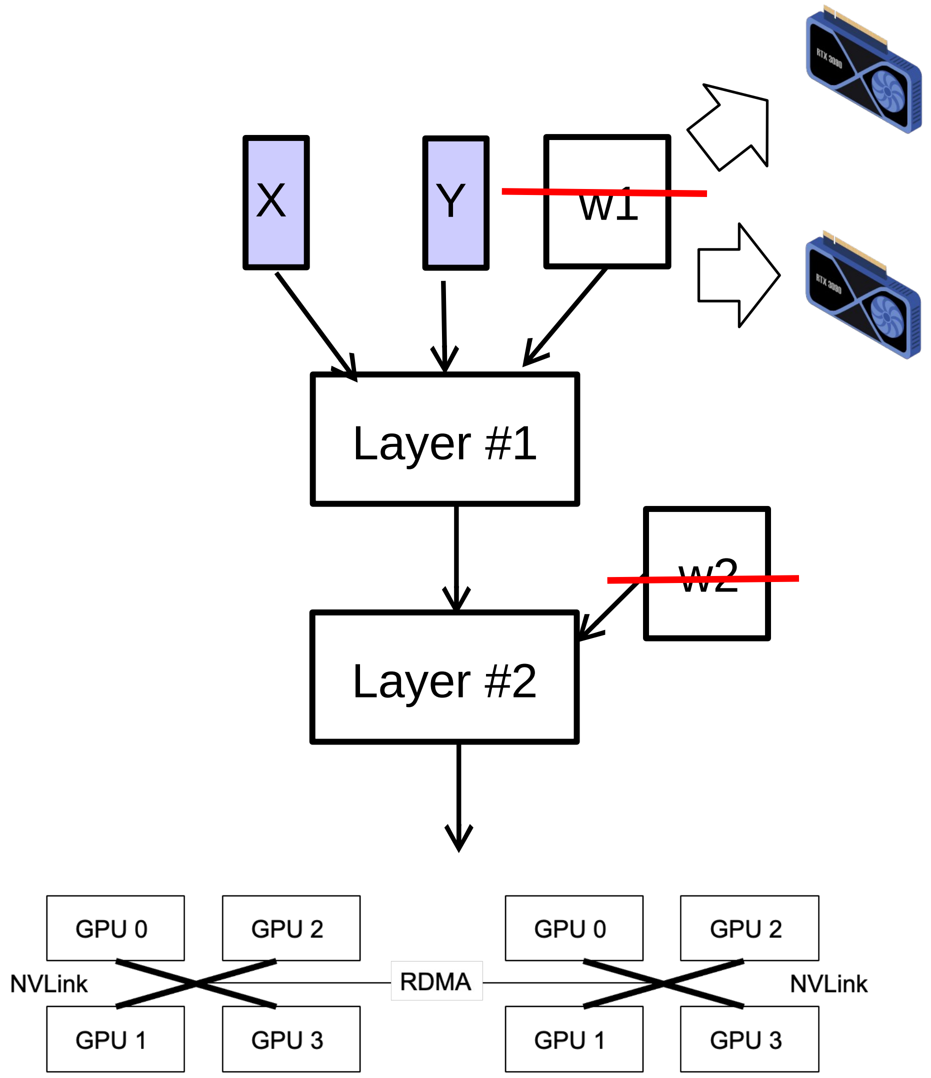
## Tensor parallelism

**Partition the parameters of a layer**

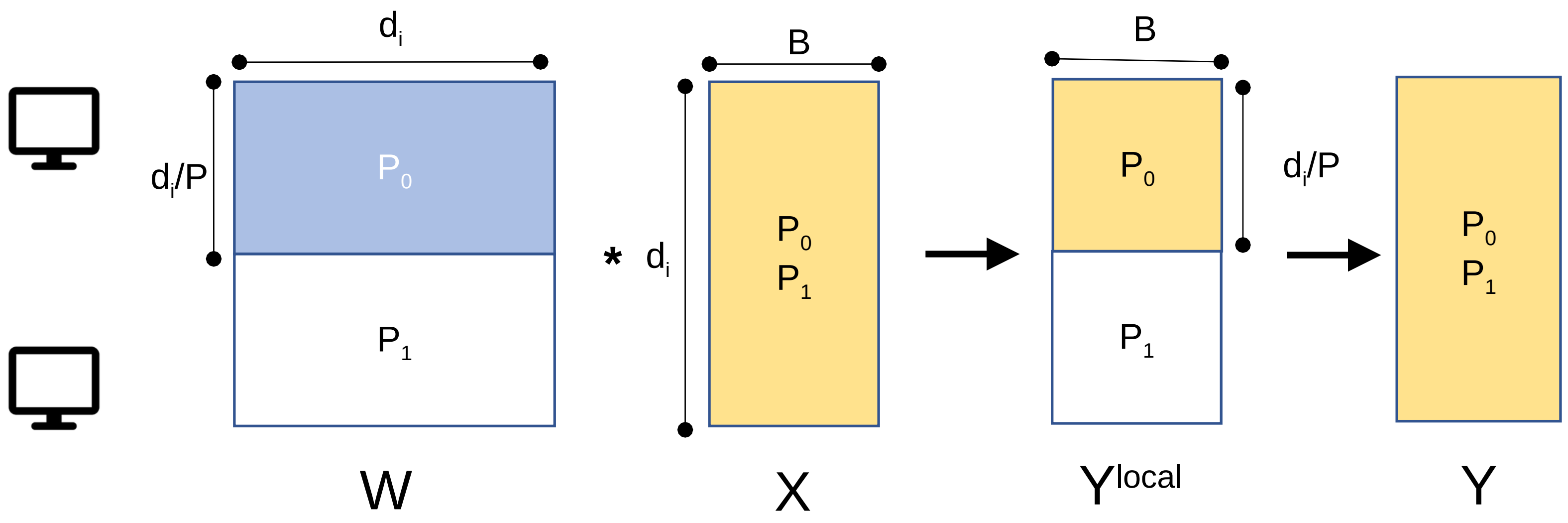
- Each partition is deployed on a separate GPU

**Pros**

- Support pipeline parallelism with a large layer
- Fit the hardware architecture of modern servers

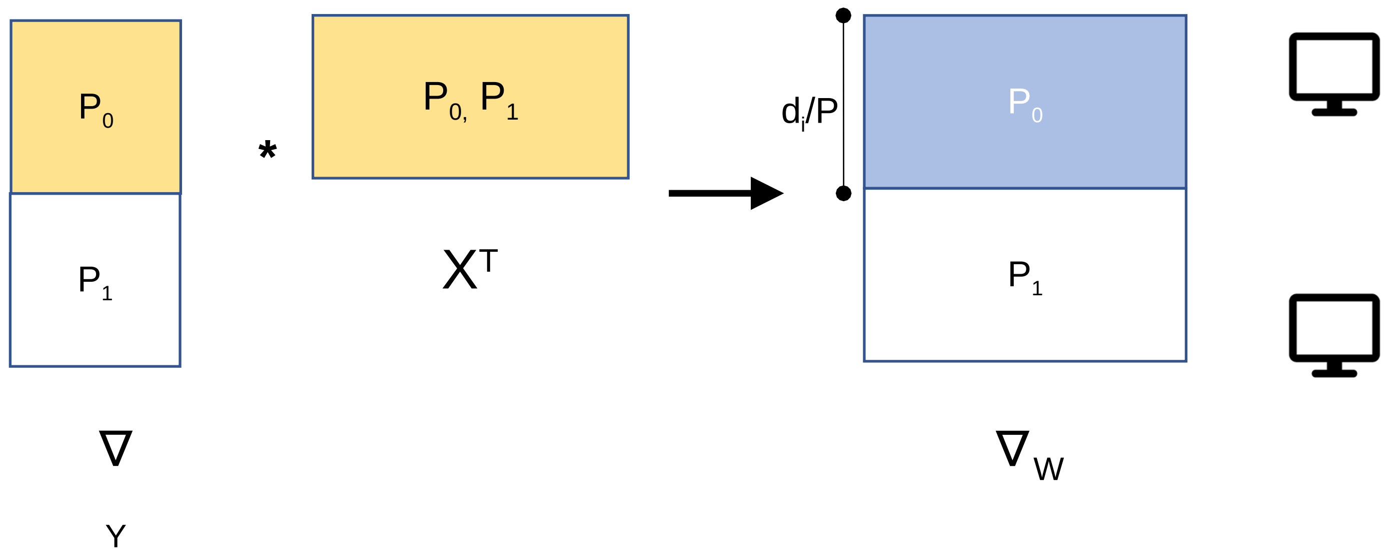


## Model Parallelism: Forward Pass



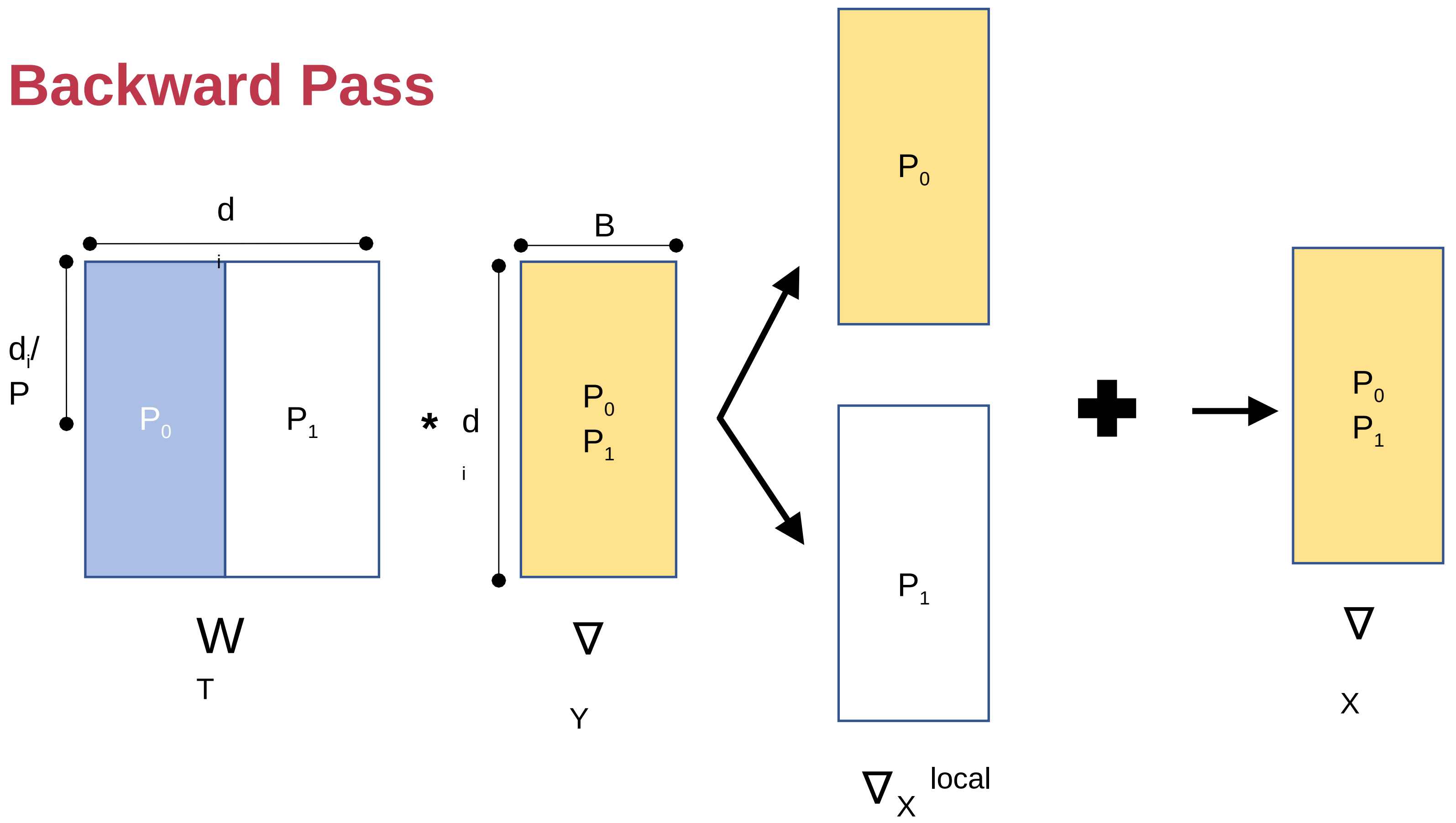
- Requires an all gather communication so that all processes get each others activation data
- Suppose with a simple forward, each node needs to send ~=

## Model Parallelism: Backward Pass



No communication needed as every processor only needs the gradient of its own parameters

## Backward Pass



- Aggregating input gradient requires an allreduce operation
- Cost (using ring): )

## Tensor parallelism: communication analysis

**Suppose W is di, and X is di, we have P partitions**

**Setup: partition the graph into P partitions w/ model parallelism**

- What is the expected communication needed for a link?
- Typically, much higher than pipeline parallelism

**Forward pass of tensor parallelism:**

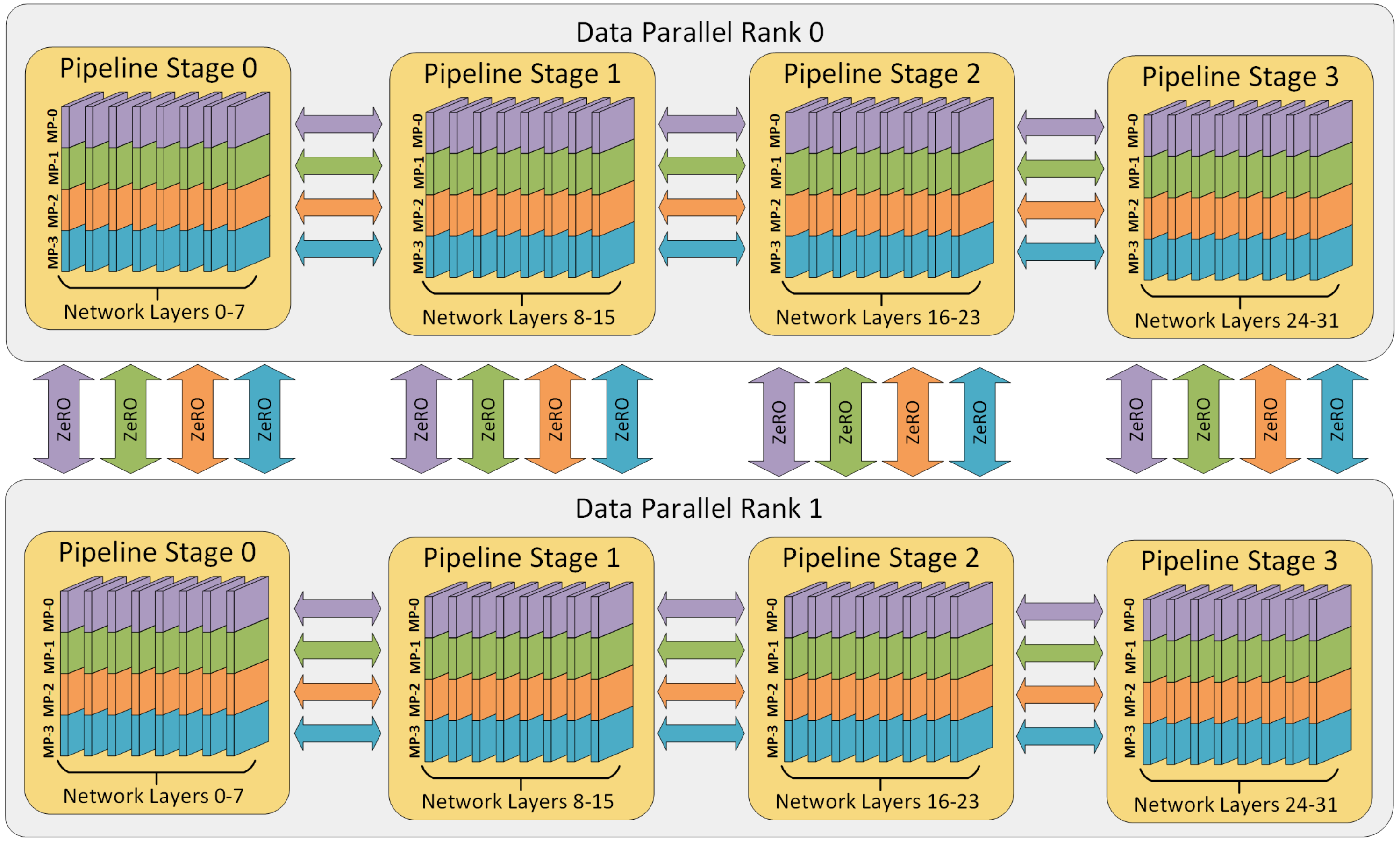
- (di * B / P) * ( P – 1) // must communicates with all the partitions

**Forward pass of pipeline parallelism:**

- (di * B / P) // only needs to pass to the next stage

**In general, tensor parallelism has a higher communication cost!**

## Put it altogether: parallelize distributed training



## Summary: model parallelism

**Two forms: pipeline parallelism + tensor parallelism**

**Pipeline parallelism**

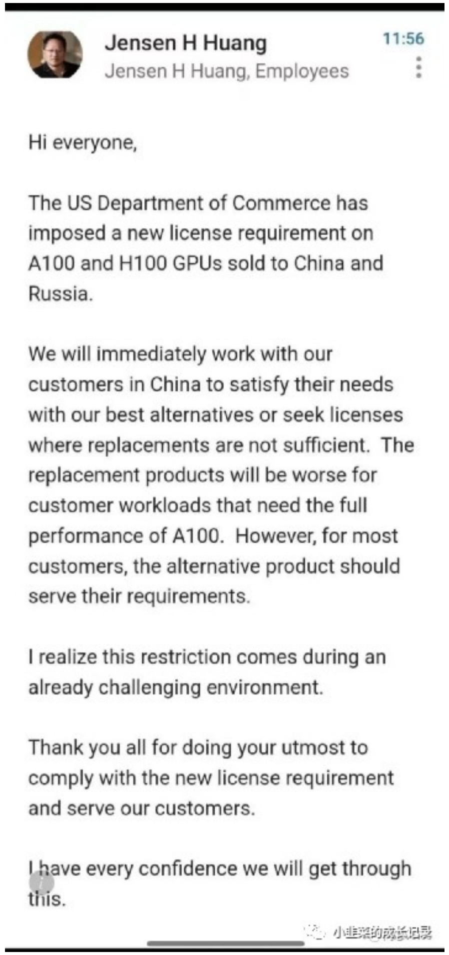
- Partition the computation graph by layers
- Pros: reduced communication
- Cons: bubbles

**Tensor parallelism**

- Partition the parameters of a layer (or a set of layers )
- Pros: better support for large models
- Cons: high communication cost

## Case study: A100 vs. A800

## The US conducted a GPU ban on China, mainly restricting GPU communication performance (some kind of out-dated)



Jensen H Huang 11:56
Jensen H Huang, Employees

Hi everyone,

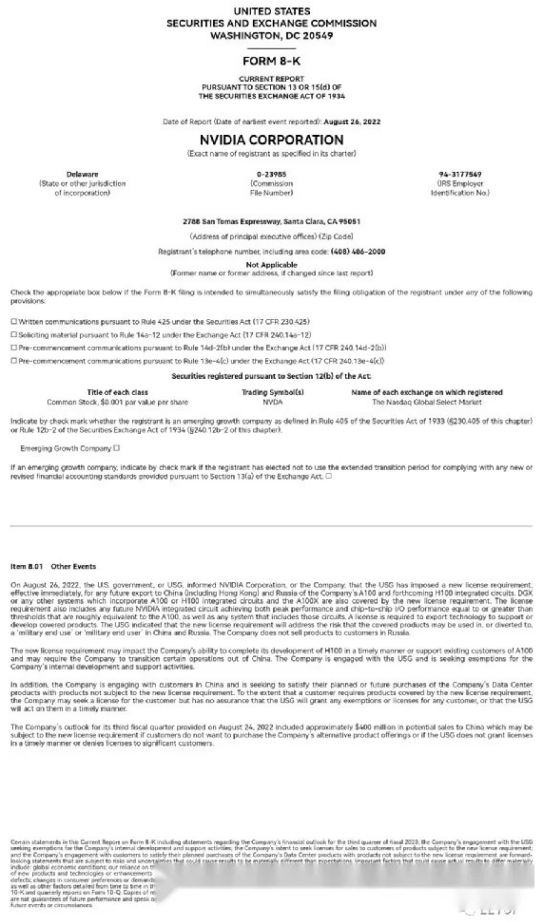
The US Department of Commerce has imposed a new license requirement on A100 and H100 GPUs sold to China and Russia.

We will immediately work with our customers in China to satisfy their needs with our best alternatives or seek licenses where replacements are not sufficient. The replacement products will be worse for customer workloads that need the full performance of A100. However, for most customers, the alternative product should serve their requirements.

I realize this restriction comes during an already challenging environment.

Thank you all for doing your utmost to comply with the new license requirement and serve our customers.

I have every confidence we will get through this.



UNITED STATES
SECURITIES AND EXCHANGE COMMISSION
WASHINGTON, DC 20549

FORM 8-K

CURRENT REPORT
PURSUANT TO SECTION 13 OR 15(d) OF
THE SECURITIES EXCHANGE ACT OF 1934

Date of Report (Date of earliest event reported): **August 26, 2022**

**NVIDIA CORPORATION**
(Exact name of registrant as specified in its charter)

| | | |
|---|---|---|
| **Delaware** | **0-23985** | **94-3177549** |
| (State or other jurisdiction of incorporation) | (Commission File Number) | (IRS Employer Identification No.) |

**2788 San Tomas Expressway, Santa Clara, CA 95051**
(Address of principal executive offices) (Zip Code)

Registrant's telephone number, including area code: **(408) 486-2000**

**Not Applicable**
(Former name or former address, if changed since last report)

Check the appropriate box below if the Form 8-K filing is intended to simultaneously satisfy the filing obligation of the registrant under any of the following provisions:

☐ Written communications pursuant to Rule 425 under the Securities Act (17 CFR 230.425)
☐ Soliciting material pursuant to Rule 14a-12 under the Exchange Act (17 CFR 240.14a-12)
☐ Pre-commencement communications pursuant to Rule 14d-2(b) under the Exchange Act (17 CFR 240.14d-2(b))
☐ Pre-commencement communications pursuant to Rule 13e-4(c) under the Exchange Act (17 CFR 240.13e-4(c))

**Securities registered pursuant to Section 12(b) of the Act:**

| Title of each class | Trading Symbol(s) | Name of each exchange on which registered |
|---|---|---|
| Common Stock, $0.001 par value per share | NVDA | The Nasdaq Global Select Market |

Indicate by check mark whether the registrant is an emerging growth company as defined in Rule 405 of the Securities Act of 1933 (§230.405 of this chapter) or Rule 12b-2 of the Securities Exchange Act of 1934 (§240.12b-2 of this chapter).

Emerging Growth Company ☐

If an emerging growth company, indicate by check mark if the registrant has elected not to use the extended transition period for complying with any new or revised financial accounting standards provided pursuant to Section 13(a) of the Exchange Act. ☐

**Item 8.01 Other Events**

On August 26, 2022, the U.S. government, or USG, informed NVIDIA Corporation, or the Company, that the USG has imposed a new license requirement, effective immediately, for any future export to China (including Hong Kong) and Russia of the Company's A100 and forthcoming H100 integrated circuits. DGX or any other systems which incorporate A100 or H100 integrated circuits and the A100X are also covered by the new license requirement. The license requirement also includes any future NVIDIA integrated circuit achieving both peak performance and chip-to-chip I/O performance equal to or greater than thresholds that are roughly equivalent to the A100, as well as any system that includes those circuits. A license is required to export technology to support or develop covered products. The USG indicated that the new license requirement will address the risk that the covered products may be used in, or diverted to, a 'military end use' or 'military end user' in China and Russia. The Company does not sell products to customers in Russia.

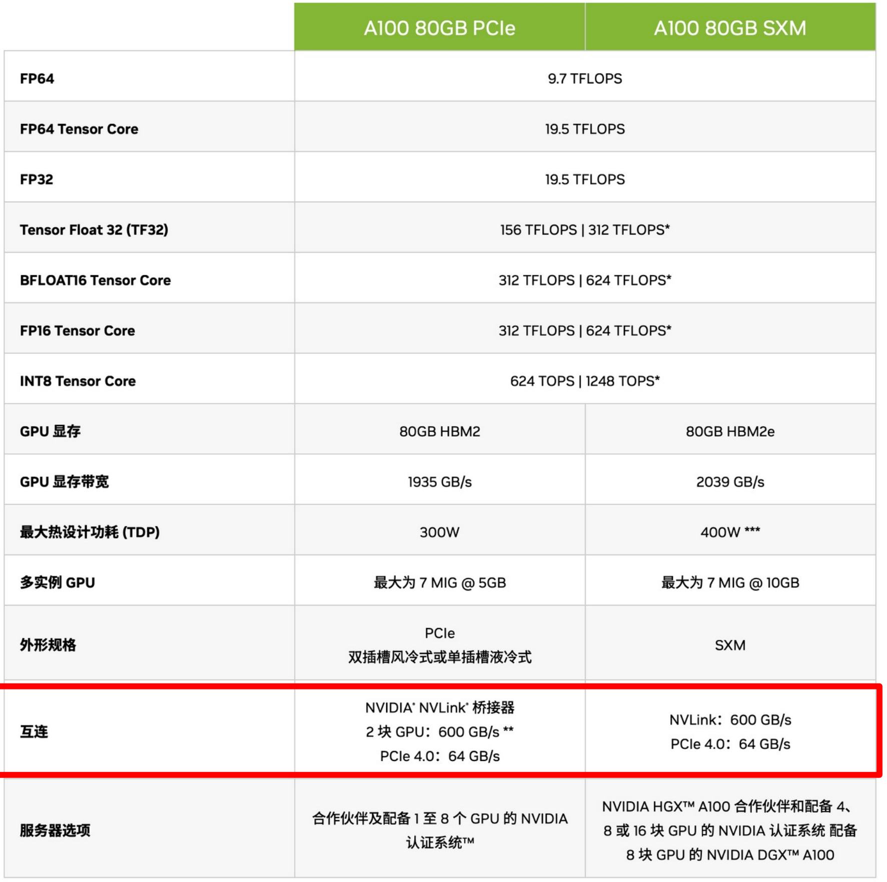
The new license requirement may impact the Company's ability to complete its development of H100 in a timely manner or support existing customers of A100 and may require the Company to transition certain operations out of China. The Company is engaged with the USG and is seeking exemptions for the Company's internal development and support activities.

In addition, the Company is engaging with customers in China and is seeking to satisfy their planned or future purchases of the Company's Data Center products with products not subject to the new license requirement. To the extent that a customer requires products covered by the new license requirement, the Company may seek a license for the customer but has no assurance that the USG will grant any exemptions or licenses for any customer, or that the USG will act on them in a timely manner.

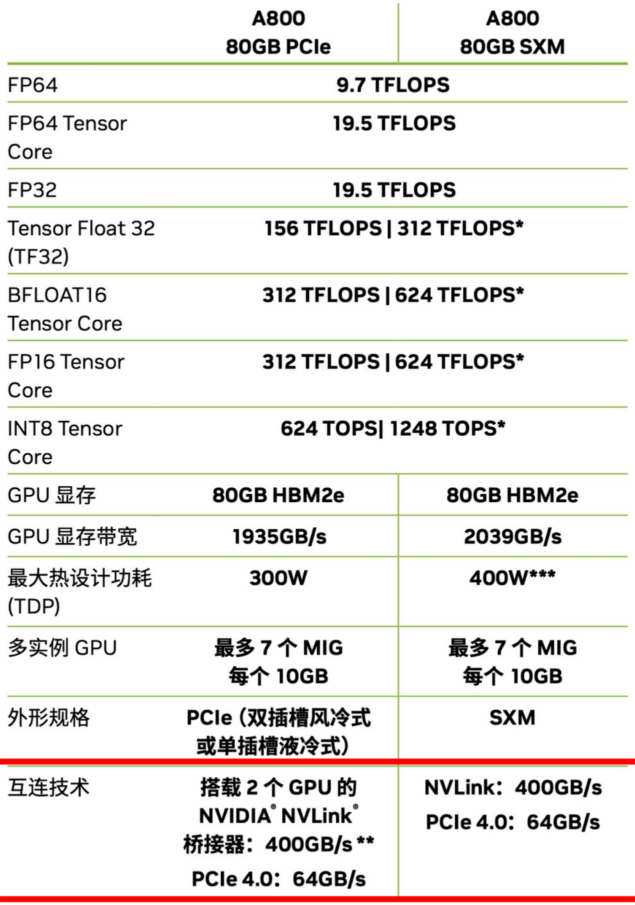
The Company's outlook for its third fiscal quarter provided on August 24, 2022 included approximately $400 million in potential sales to China which may be subject to the new license requirement if customers do not want to purchase the Company's alternative product offerings or if the USG does not grant licenses in a timely manner or denies licenses to significant customers.

Certain statements in this Current Report on Form 8-K including statements regarding the Company's financial outlook for the third quarter of fiscal 2023; the Company's engagement with the USG seeking exemptions for the Company's internal development and support activities; the Company's intent to seek licenses for sales to customers of products subject to the new license requirement; and the Company's engagement with customers to satisfy their planned purchases of the Company's Data Center products with products not subject to the new license requirement are forward-looking statements that are subject to risks and uncertainties that could cause results to be materially different than expectations. Important factors that could cause actual results to differ materially include: global economic conditions; our reliance on th [illegible]
of new products and technologies or enhancements [illegible]
defects; changes in consumer preferences or demands [illegible]
as well as other factors detailed from time to time in th [illegible]
10-K and quarterly reports on Form 10-Q. Copies of re [illegible]
are not guarantees of future performance and speak o [illegible]
future events or circumstances.

# Case study: A100 vs. A800



| | A100 80GB PCIe | A100 80GB SXM |
|---|---|---|
| **FP64** | 9.7 TFLOPS | |
| **FP64 Tensor Core** | 19.5 TFLOPS | |
| **FP32** | 19.5 TFLOPS | |
| **Tensor Float 32 (TF32)** | 156 TFLOPS \| 312 TFLOPS* | |
| **BFLOAT16 Tensor Core** | 312 TFLOPS \| 624 TFLOPS* | |
| **FP16 Tensor Core** | 312 TFLOPS \| 624 TFLOPS* | |
| **INT8 Tensor Core** | 624 TOPS \| 1248 TOPS* | |
| **GPU 显存** | 80GB HBM2 | 80GB HBM2e |
| **GPU 显存带宽** | 1935 GB/s | 2039 GB/s |
| **最大热设计功耗 (TDP)** | 300W | 400W *** |
| **多实例 GPU** | 最大为 7 MIG @ 5GB | 最大为 7 MIG @ 10GB |
| **外形规格** | PCIe<br>双插槽风冷式或单插槽液冷式 | SXM |
| **互连** | NVIDIA® NVLink® 桥接器<br>2 块 GPU：600 GB/s **<br>PCIe 4.0：64 GB/s | NVLink：600 GB/s<br>PCIe 4.0：64 GB/s |
| **服务器选项** | 合作伙伴及配备 1 至 8 个 GPU 的 NVIDIA 认证系统™ | NVIDIA HGX™ A100 合作伙伴和配备 4、8 或 16 块 GPU 的 NVIDIA 认证系统 配备 8 块 GPU 的 NVIDIA DGX™ A100 |



| | A800 80GB PCIe | A800 80GB SXM |
|---|---|---|
| FP64 | **9.7 TFLOPS** | |
| FP64 Tensor Core | **19.5 TFLOPS** | |
| FP32 | **19.5 TFLOPS** | |
| Tensor Float 32 (TF32) | **156 TFLOPS \| 312 TFLOPS*** | |
| BFLOAT16 Tensor Core | **312 TFLOPS \| 624 TFLOPS*** | |
| FP16 Tensor Core | **312 TFLOPS \| 624 TFLOPS*** | |
| INT8 Tensor Core | **624 TOPS\| 1248 TOPS*** | |
| GPU 显存 | **80GB HBM2e** | **80GB HBM2e** |
| GPU 显存带宽 | **1935GB/s** | **2039GB/s** |
| 最大热设计功耗 (TDP) | **300W** | **400W*****|
| 多实例 GPU | **最多 7 个 MIG 每个 10GB** | **最多 7 个 MIG 每个 10GB** |
| 外形规格 | **PCIe（双插槽风冷式或单插槽液冷式）** | **SXM** |
| 互连技术 | **搭载 2 个 GPU 的 NVIDIA® NVLink® 桥接器：400GB/s ****<br>**PCIe 4.0：64GB/s** | **NVLink：400GB/s**<br>**PCIe 4.0：64GB/s** |

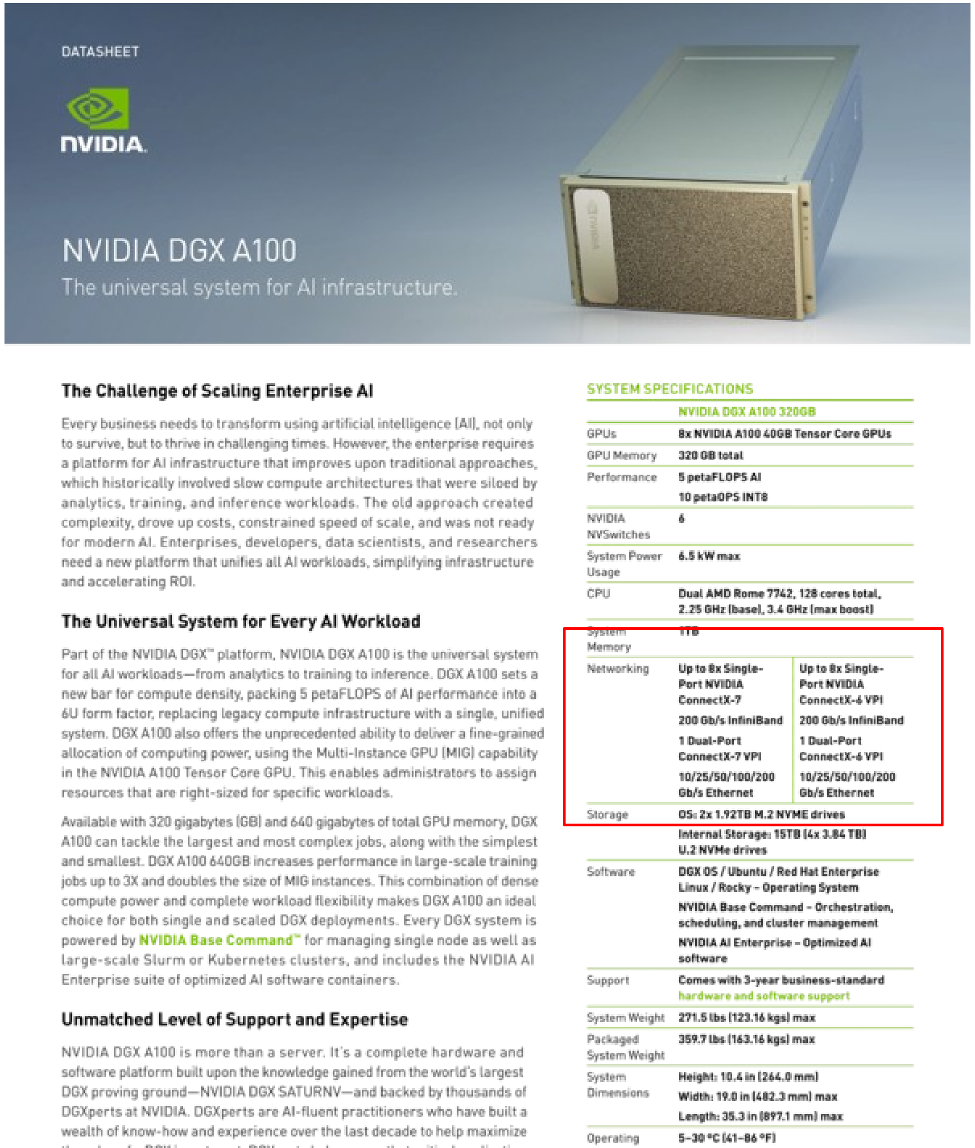
## Case study: Distributed training on DGX A100 (8 X A100 server)

### Key components

- 8 X A100 GPUs
- Up to 8 X ConnectX-6/7 NIC (50/100/200 Gbps, unidirectional)

### GPUs are connected via NVLink

- 600GBps (bi-directional) w/ A100
- 400GBps (bi-directional) w/ A800



DATASHEET

NVIDIA DGX A100

The universal system for AI infrastructure.

**The Challenge of Scaling Enterprise AI**

Every business needs to transform using artificial intelligence (AI), not only to survive, but to thrive in challenging times. However, the enterprise requires a platform for AI infrastructure that improves upon traditional approaches, which historically involved slow compute architectures that were siloed by analytics, training, and inference workloads. The old approach created complexity, drove up costs, constrained speed of scale, and was not ready for modern AI. Enterprises, developers, data scientists, and researchers need a new platform that unifies all AI workloads, simplifying infrastructure and accelerating ROI.

**The Universal System for Every AI Workload**

Part of the NVIDIA DGX™ platform, NVIDIA DGX A100 is the universal system for all AI workloads—from analytics to training to inference. DGX A100 sets a new bar for compute density, packing 5 petaFLOPS of AI performance into a 6U form factor, replacing legacy compute infrastructure with a single, unified system. DGX A100 also offers the unprecedented ability to deliver a fine-grained allocation of computing power, using the Multi-Instance GPU (MIG) capability in the NVIDIA A100 Tensor Core GPU. This enables administrators to assign resources that are right-sized for specific workloads.

Available with 320 gigabytes (GB) and 640 gigabytes of total GPU memory, DGX A100 can tackle the largest and most complex jobs, along with the simplest and smallest. DGX A100 640GB increases performance in large-scale training jobs up to 3X and doubles the size of MIG instances. This combination of dense compute power and complete workload flexibility makes DGX A100 an ideal choice for both single and scaled DGX deployments. Every DGX system is powered by NVIDIA Base Command™ for managing single node as well as large-scale Slurm or Kubernetes clusters, and includes the NVIDIA AI Enterprise suite of optimized AI software containers.

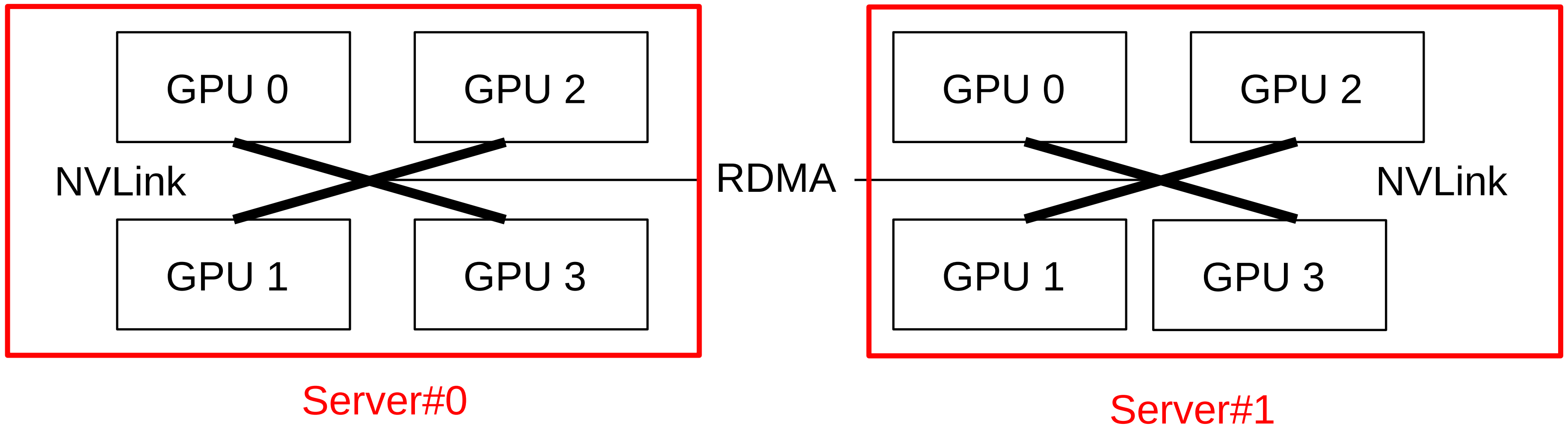
**Unmatched Level of Support and Expertise**

NVIDIA DGX A100 is more than a server. It's a complete hardware and software platform built upon the knowledge gained from the world's largest DGX proving ground—NVIDIA DGX SATURNV—and backed by thousands of DGXperts at NVIDIA. DGXperts are AI-fluent practitioners who have built a wealth of know-how and experience over the last decade to help maximize

SYSTEM SPECIFICATIONS

| | NVIDIA DGX A100 320GB | |
|---|---|---|
| GPUs | 8x NVIDIA A100 40GB Tensor Core GPUs | |
| GPU Memory | 320 GB total | |
| Performance | 5 petaFLOPS AI<br>10 petaOPS INT8 | |
| NVIDIA NVSwitches | 6 | |
| System Power Usage | 6.5 kW max | |
| CPU | Dual AMD Rome 7742, 128 cores total, 2.25 GHz (base), 3.4 GHz (max boost) | |
| System Memory | 1TB | |
| Networking | Up to 8x Single-Port NVIDIA ConnectX-7<br>200 Gb/s InfiniBand<br>1 Dual-Port ConnectX-7 VPI<br>10/25/50/100/200 Gb/s Ethernet | Up to 8x Single-Port NVIDIA ConnectX-6 VPI<br>200 Gb/s InfiniBand<br>1 Dual-Port ConnectX-6 VPI<br>10/25/50/100/200 Gb/s Ethernet |
| Storage | OS: 2x 1.92TB M.2 NVME drives<br>Internal Storage: 15TB (4x 3.84 TB) U.2 NVMe drives | |
| Software | DGX OS / Ubuntu / Red Hat Enterprise Linux / Rocky – Operating System<br>NVIDIA Base Command – Orchestration, scheduling, and cluster management<br>NVIDIA AI Enterprise – Optimized AI software | |
| Support | Comes with 3-year business-standard hardware and software support | |
| System Weight | 271.5 lbs (123.16 kgs) max | |
| Packaged System Weight | 359.7 lbs (163.16 kgs) max | |
| System Dimensions | Height: 10.4 in (264.0 mm)<br>Width: 19.0 in (482.3 mm) max<br>Length: 35.3 in (897.1 mm) max | |
| Operating | 5–30 °C (41–86 °F) | |

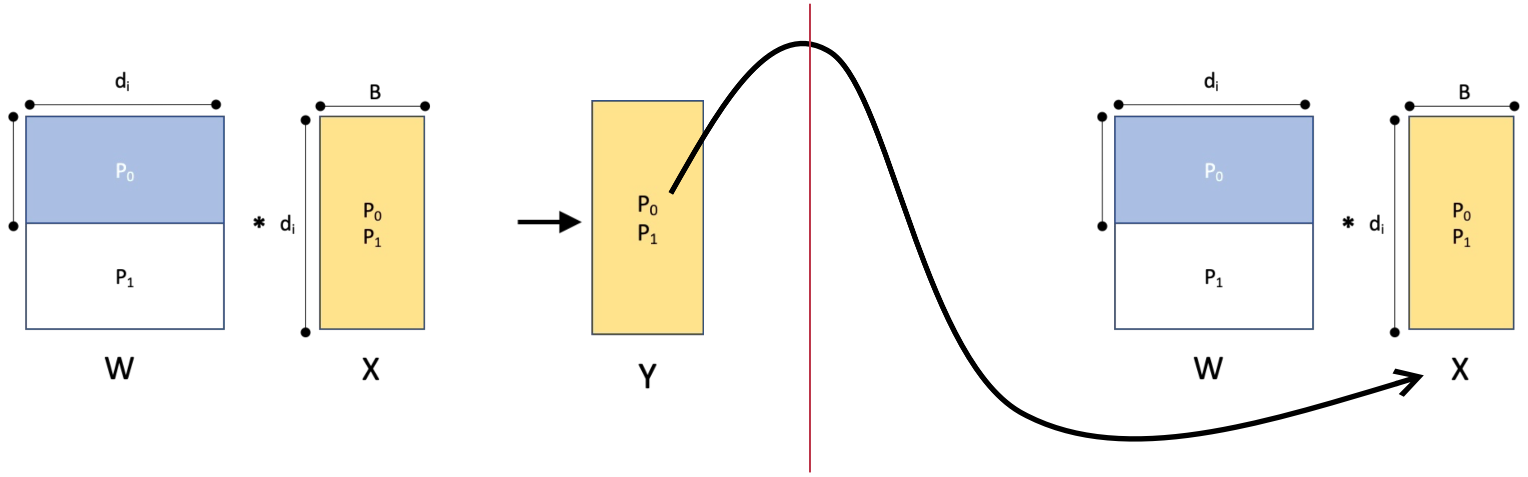
## Thought experiment: why 600 -> 400GB/s?



**Interconnect setup**

- Between servers, up to 8 X 200Gbps ConnectX-6/7 (200GBps, uni-directional); however, in real evaluation, **typically measured 160GBps (why?)**
- Between GPUs: 600/400 GBps NVLink (bi-directional) = 300/200 GBps NVLink (uni-directional)

## Thought experiment: why 600 -> 400GB/s?



Server #0

Server #1

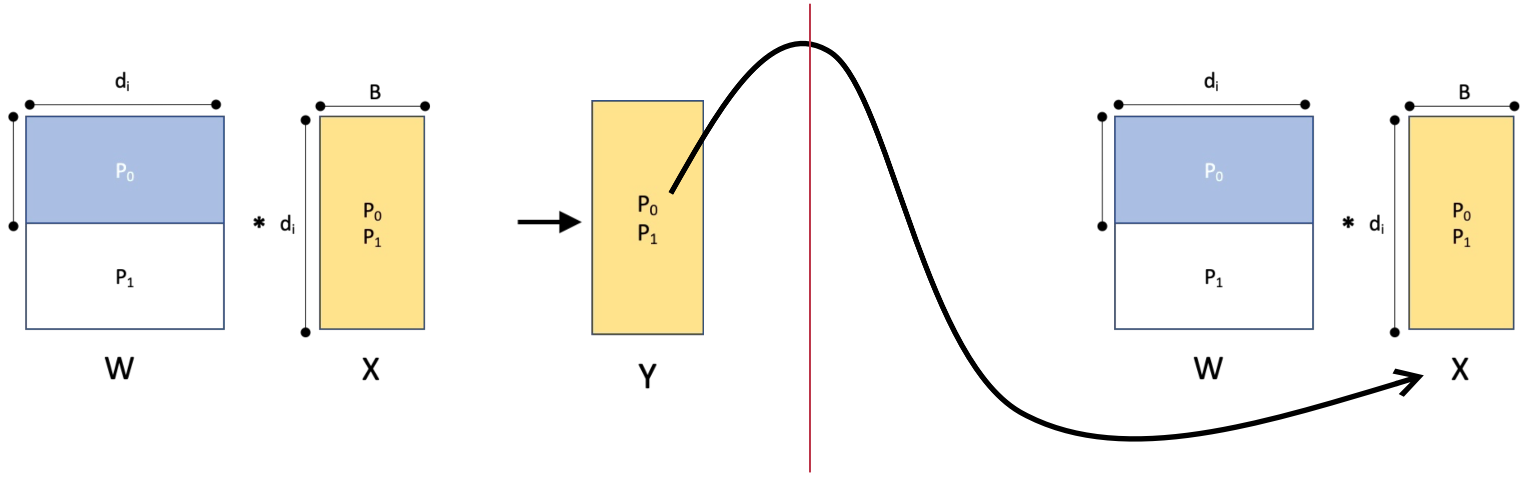
**Setup: pipeline + tensor parallelism**

– Two servers, each server: 8 GPUs (e.g., DGX A100)

– Within each server, tensor parallelism (model partitioned on 8 GPUs)

– Between servers, pipeline parallelism

## Thought experiment: why 600 -> 400GB/s?



Server #0 Server #1

**Communication analysis (Backward path, assuming 8 GPUs)**

- Between machine: bytes sent per iteration
- Between GPU: bytes sent per iteration

**The bandwidth of NVLink should be 1.75X larger to prevent being the bottleneck**

## Thought experiment: why 600 -> 400GB/s?

**Goal: NVLink should be 1.75X faster than RDMA**

– To prevent becoming the communication bottleneck in our pipeline—tensor hybrid parallelism

**A100 vs. RDMA**

–

**A800 vs. RDMA**

–

**With A800: the model parallelism will be bottlenecked by the NVLink !**

## Parallelization Opportunities

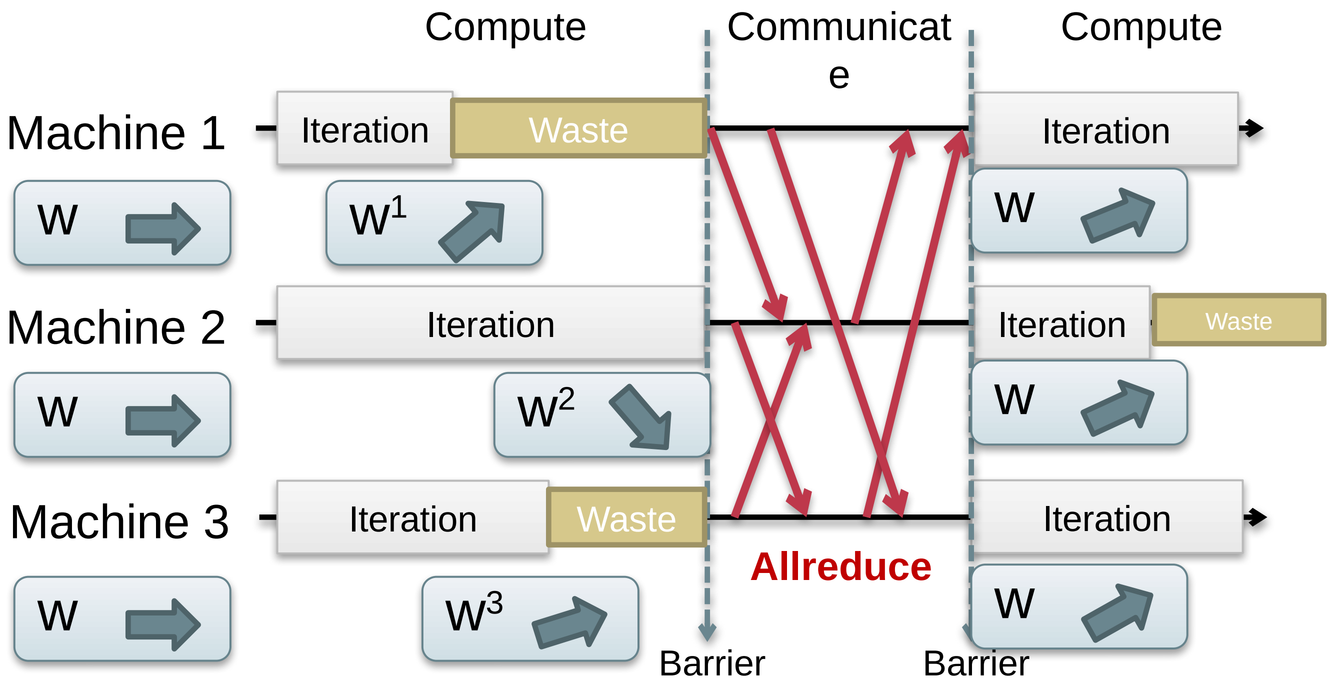
***Data Parallelism:*** *Distribute the processing of data to multiple PEs.*

***Model Parallelism:*** *Break the model and distribute processing of every layer to multiple PEs*

*For either approach it is also possible to use **synchronous** or **asynchronous** updates*

# Recall: BSP for data parallel execution

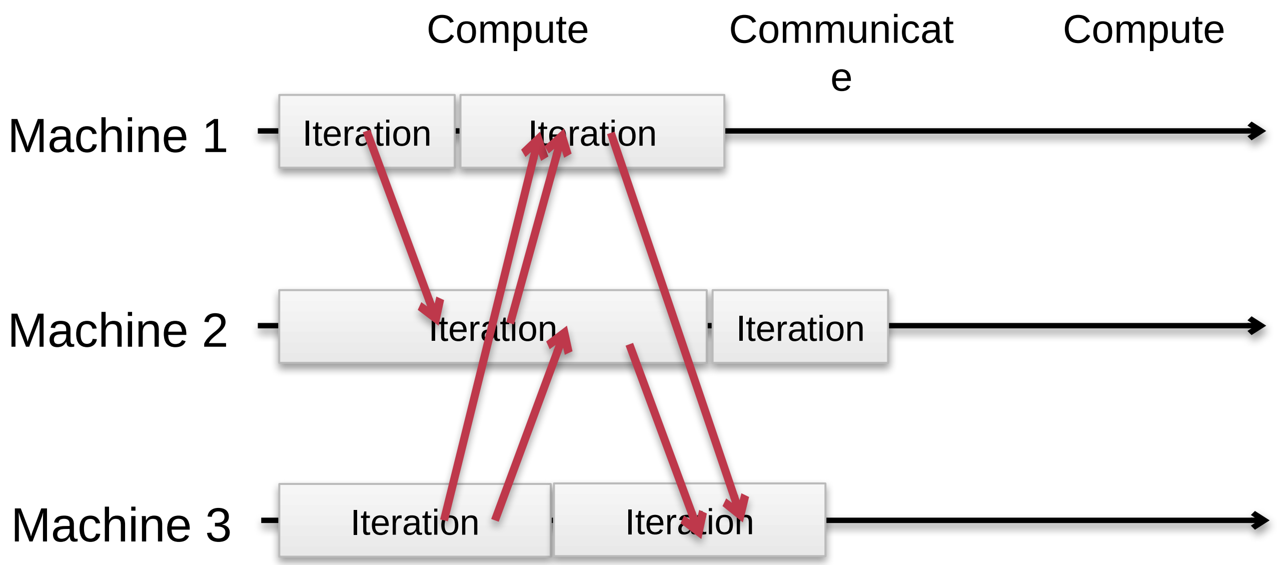
**Problem: may have idle time at each process**



## Asynchronous Execution

**Key idea: each process does not wait for others**

– Update the gradients upon receive a messages, no iteration barrier



## Async vs. Sync execution

**Common**: nearly all training currently adopts a synchronous approach

**Async**

- Pros: efficiency
- Cons: trade accuracy for performance

**Sync**

- Pros: Preserve the SGD training property
- Cons: poor performance

**Which is better?**

- Hard to tell. But, as a system designer, preserving the algorithms property is **very important**. Any design that alter the training property should be justified