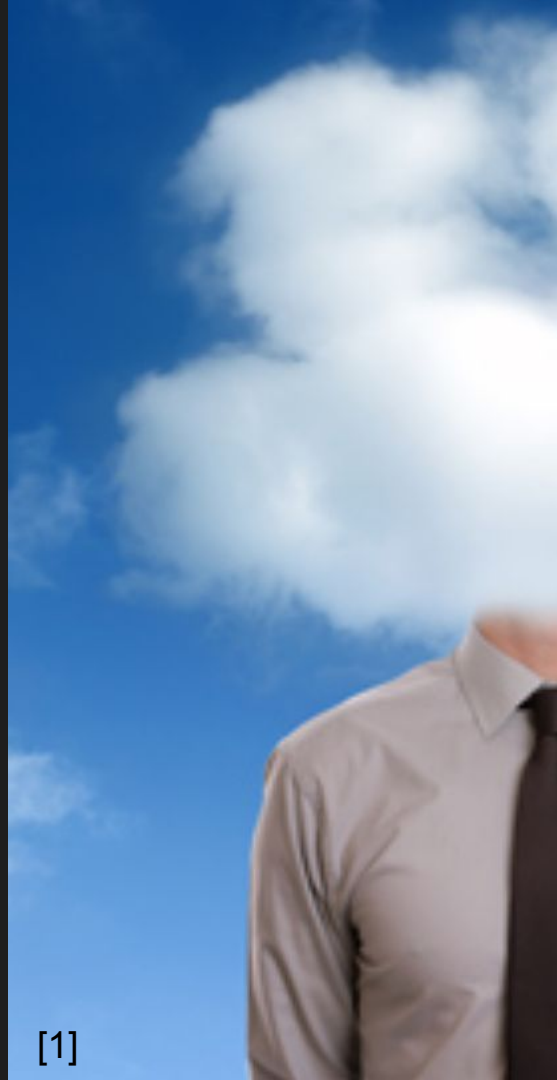
Multi-GPU Neural Network Training in the Cloud

Things to consider when determining if hosted GPUs are right for you?

Project Scope

Many different considerations

- Economy - Cost evaluation
  - Cost of purchasing machine
  - Cost of running in the cloud
- Workflow
  - VS Code
  - Lambda Cloud
- Leveraging Multiple GPUs
  - The task
  - The models
  - Performance

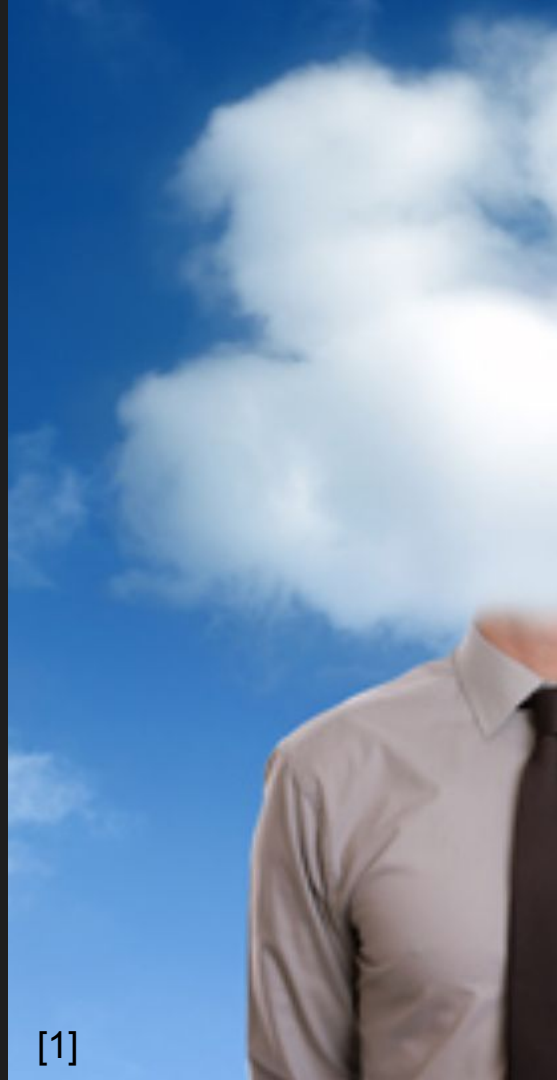


[1]

Project Scope

Many different considerations

- Economy - Cost evaluation
  - Cost of purchasing machine (Lambda)
  - Cost of running in the cloud (Lambda)
- Workflow
  - VS Code
  - Lambda Cloud
- Leveraging Multiple GPUs
  - The task
  - The models
  - Performance



[1]

Economy

Comparing the cost of owning dedicated hardware vs training on the cloud

Lambda Labs

- Physical Products
  - Range of technical specifications
  - Servers: $45,000 - $275,000 (8-16 GPUs)
  - Workstations: $5,500 - $33,000 (2-4 GPUs)
  - TensorBooks: $2,500 - $3,500 (1 GPU)
- Cloud Service
  - Cloud Machine: $1.50 / Hour (4 GPUs)
  - Multiple instances can be launched

[2]

What is the Cloud Service?

I appreciate your patience with my response.

You are connecting a single virtualized Lambda Quad workstation with 4x GPUs and 1 CPU. We have hundreds of these machines clustered together in our current version of the Cloud. Networking is dedicated.

The majority of our users are training language and vision models on our cloud instances.

Comparing Similar Computational Performance

No Exact Comparison

Choice	GPUs (4)	Cores / vCores	Ram
Cloud Instance	GTX 1080 Ti	8 vCores @ 3.5 Ghz	32 GB
Machine	RTX 2080 Ti	10 Cores @ 3.3Ghz	64 GB

GTX 1080 Ti ~ 30% less efficient than RTX 2080 Ti [4]

Choice	Cost
Cloud Instance	$1.50 / Hour
Machine	$8,000.00
Machine (adjusted)	$5,600.000 (-30%)

Cost Equality

~ 3,750 hours training to purchase adjusted machine
- That’s ~1.75 years full time

~ 5,250 hours training to purchase standard machine
- That’s ~2.5 years full time

Full time = 2000 hours / year

Economy: Final Notes

- Unlike workstations, Lambda instances can be scaled up as needed, effectively multiplying those hours of training you need when you need it.

- As far as I can tell, Lambda charges by the 1/100 of an hour.

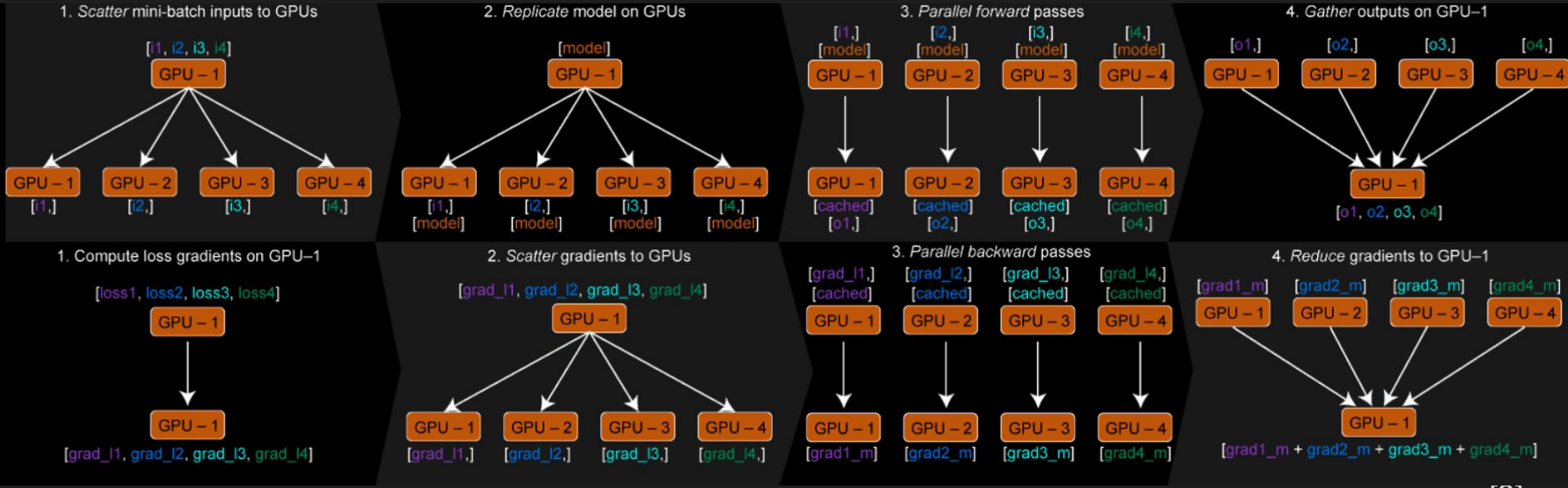
Resource Name	Spending	Rate	Resource type	Usage
c9772aaf02a9465d874e583696e370ec	$1.01	$1.50 / hour	gpu.4x	0.67 hours
dba0d575f02441d3b96f94d34dbf18c6	$1.92	$1.50 / hour	gpu.4x	1.28 hours
6927d2a7908c4f8ab896c71e71bb0a6b	$1.12	$1.50 / hour	gpu.4x	0.75 hours
e11948ab4c984f6d9bb35fbc3b3f3861	$1.45	$1.50 / hour	gpu.4x	0.97 hours
861f5a08bd724bca8a4b6f2a5ab62760	$1.96	$1.50 / hour	gpu.4x	1.31 hours
a5fa4ceca66846d4b81886d7acc4e4c7	$1.30	$1.50 / hour	gpu.4x	0.86 hours

[2]

Characterizing Multi-GPU Data Parallel NN Training

Model Architecture Considerations When Using PyTorch

How DataParallel() works in PyTorch

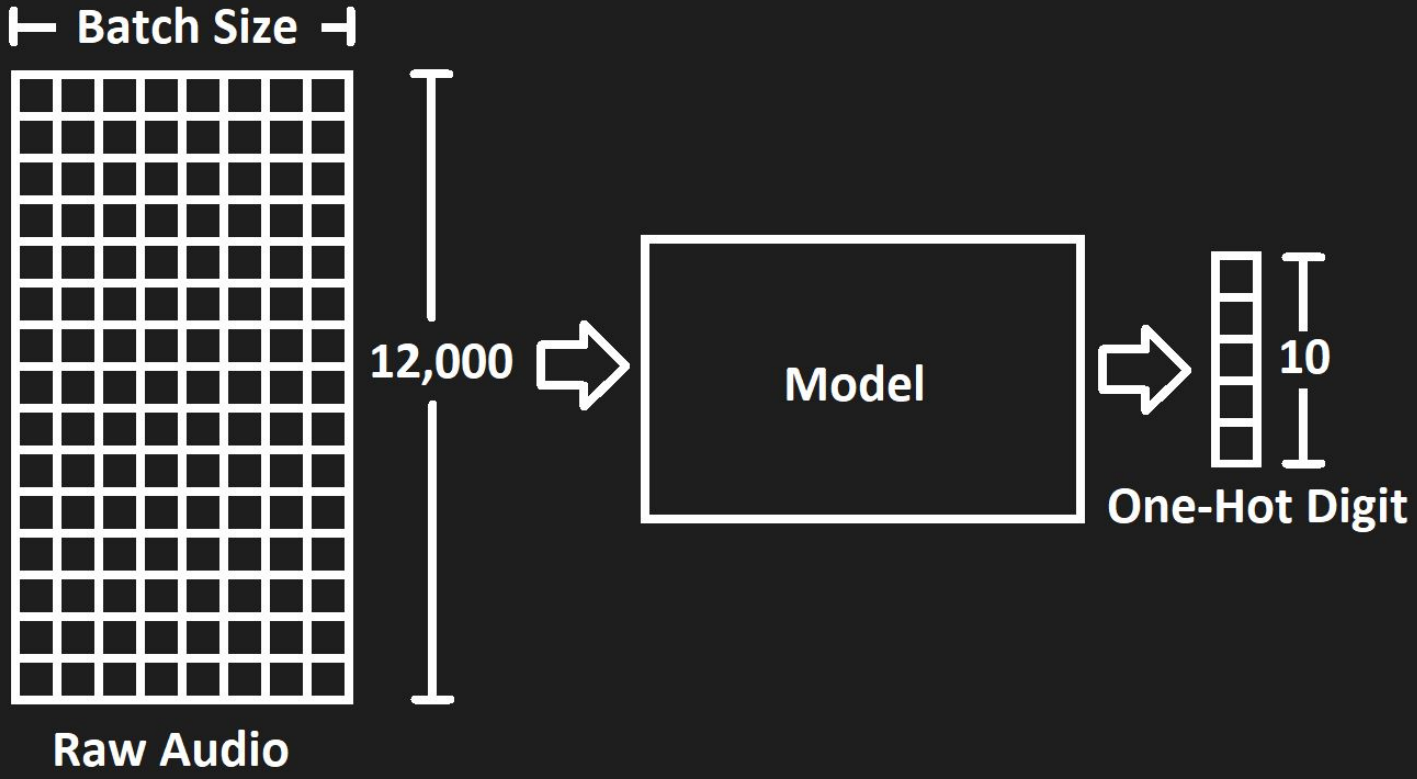


[3]

Machine Learning Task: Vocal Digit Recognition

- Data:
  - 2,000 1.5 second single digit clips
  - 4 speakers repeating each digit 50 times
  - Epoch: 92 MB
  - At: https://github.com/Jakobovski/free-spoken-digit-dataset
- Models
  - 1 fully connected
  - 1 convolutional
- Task Performance
  - Both perform with over 99% percent accuracy on training data
  - Probably extraordinary overfitting going on
  - Still pretty cool
  - Not the point of the project

Model Input / Output



Linear Model Architecture

12,000 → 1,000

1,000 → 100

100 → 10

10 → 10

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1              [-1, 1, 1000]      12,001,000
              ReLU-2              [-1, 1, 1000]               0
            Linear-3               [-1, 1, 100]         100,100
              ReLU-4               [-1, 1, 100]               0
            Linear-5                [-1, 1, 10]           1,010
           Sigmoid-6                [-1, 1, 10]               0
            Linear-7                [-1, 1, 10]             110
================================================================
Total params: 12,102,220
Trainable params: 12,102,220
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 0.02
Params size (MB): 46.17
Estimated Total Size (MB): 46.23
----------------------------------------------------------------
```

- Train Parameters: 12,102,220
- Size of Parameters: 46 MB

Convolutional Model Architecture

12,000 → 3,000 (50 x 200 kernel)

3,000 → 1,000 (30 x 100 kernel)

1,000 → 200 (10 x 50 kernel)

Flatten (10 x 200) → 2,000

2,000 → 100

100 → 10

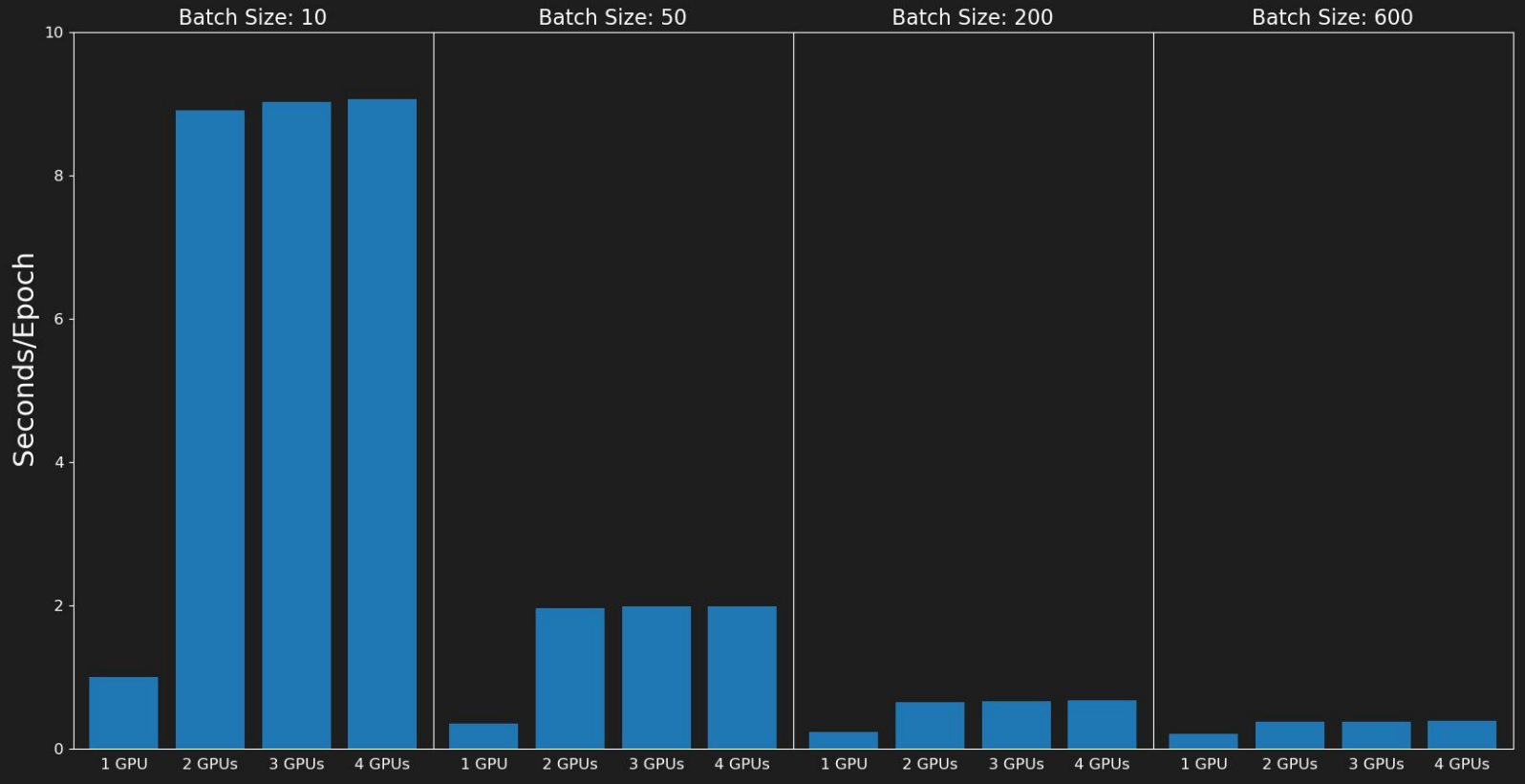
- Train Parameters: 376,200
- Size of Parameters: 1.4 MB

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv1d-1          [-1, 50, 12001]          10,050
              ReLU-2          [-1, 50, 12001]               0
         MaxPool1d-3           [-1, 50, 3000]               0
            Conv1d-4           [-1, 30, 3001]         150,030
              ReLU-5           [-1, 30, 3001]               0
         MaxPool1d-6           [-1, 30, 1000]               0
            Conv1d-7           [-1, 10, 1001]          15,010
              ReLU-8           [-1, 10, 1001]               0
         MaxPool1d-9            [-1, 10, 200]               0
           Linear-10                [-1, 100]         200,100
          Sigmoid-11                [-1, 100]               0
           Linear-12                 [-1, 10]           1,010
================================================================
Total params: 376,200
Trainable params: 376,200
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 12.07
Params size (MB): 1.44
Estimated Total Size (MB): 13.55
----------------------------------------------------------------
```

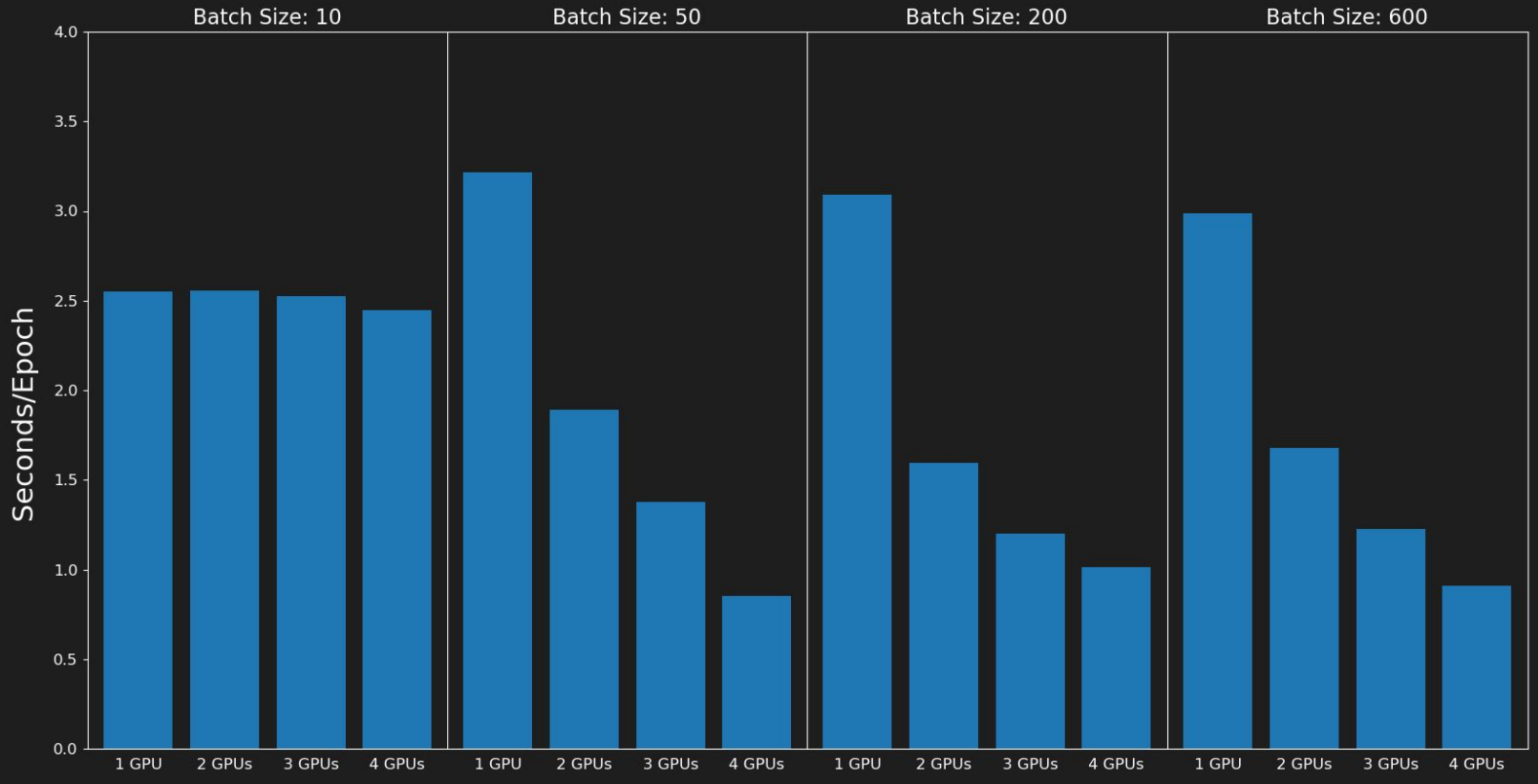
The Experiment

```
epochs = 20
dev_ids = [0,1,2,3]
batch_sizes = [10,20,30,50,70,100,150,200,300,400,500,600]

if __name__ == '__main__':
    linear_stats = get_stats(train_linear, epochs=epochs, dev_ids=dev_ids, batch_sizes=batch_sizes)
    conv_stats = get_stats(train_conv, epochs=epochs, dev_ids=dev_ids, batch_sizes=batch_sizes)
```

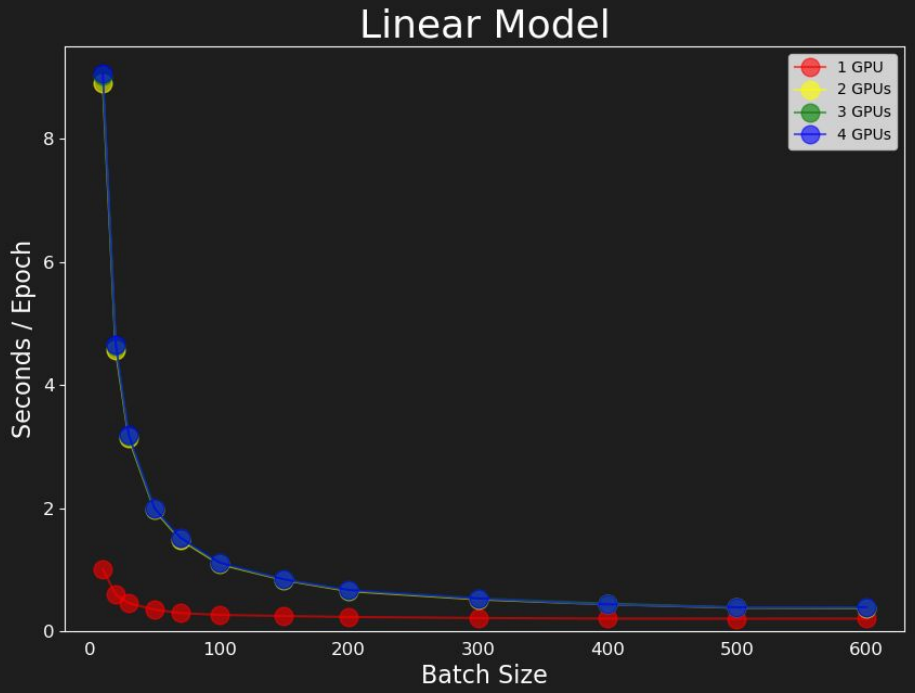
Linear Model Results

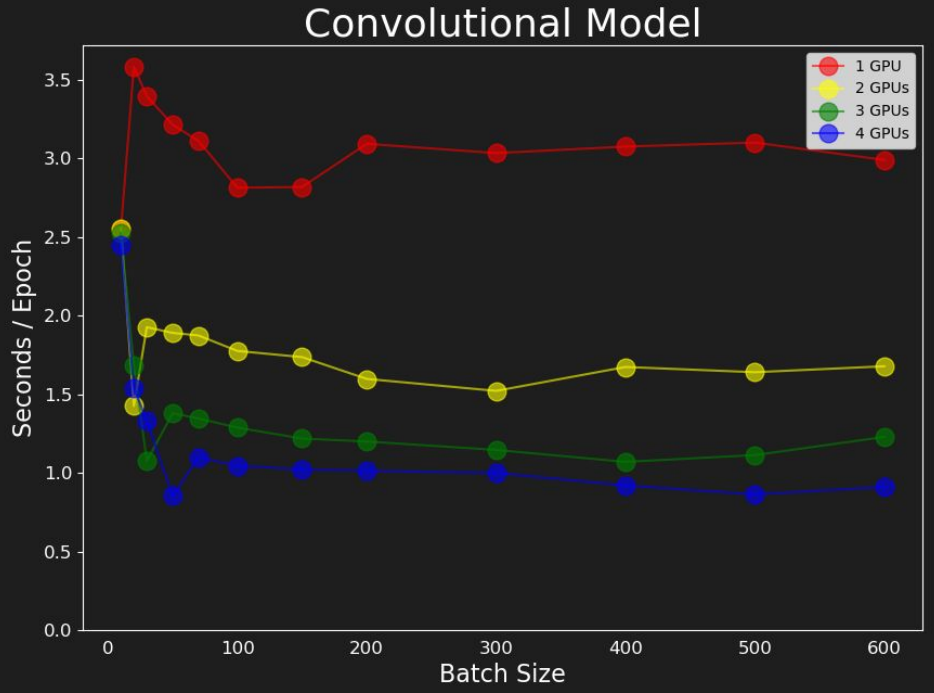


Convolutional Results

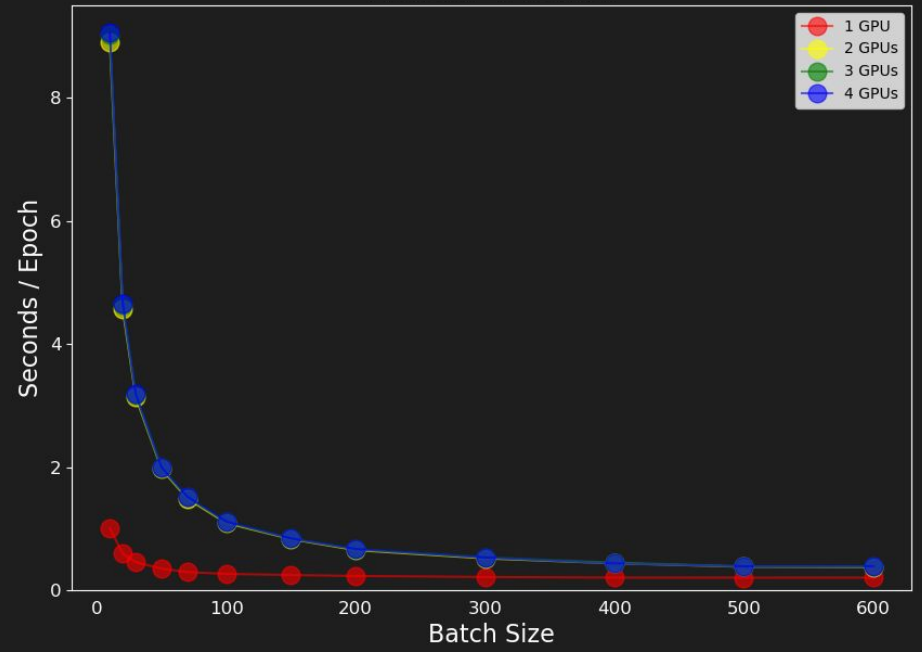


Model Comparison

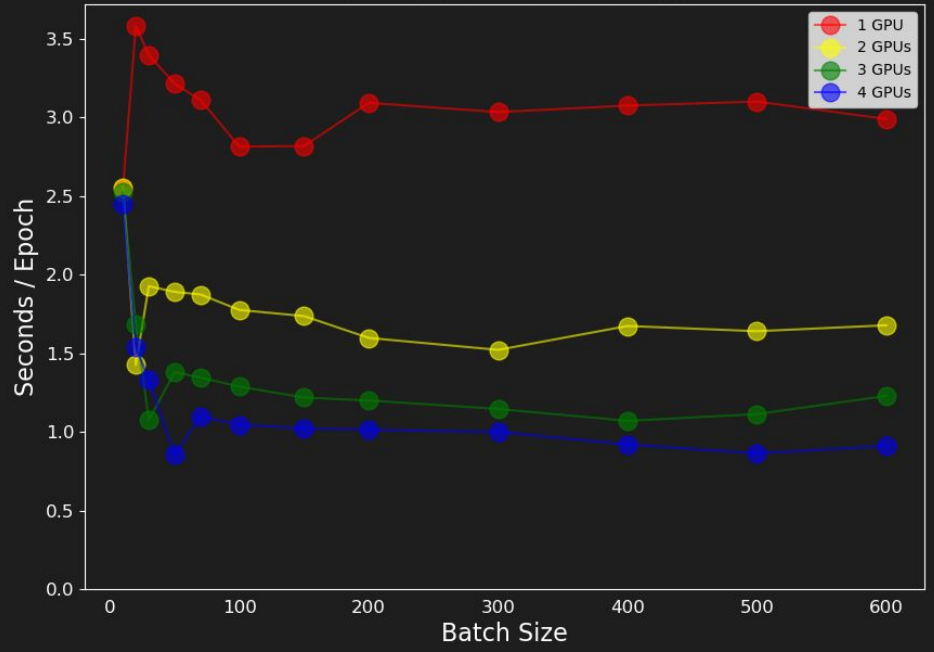




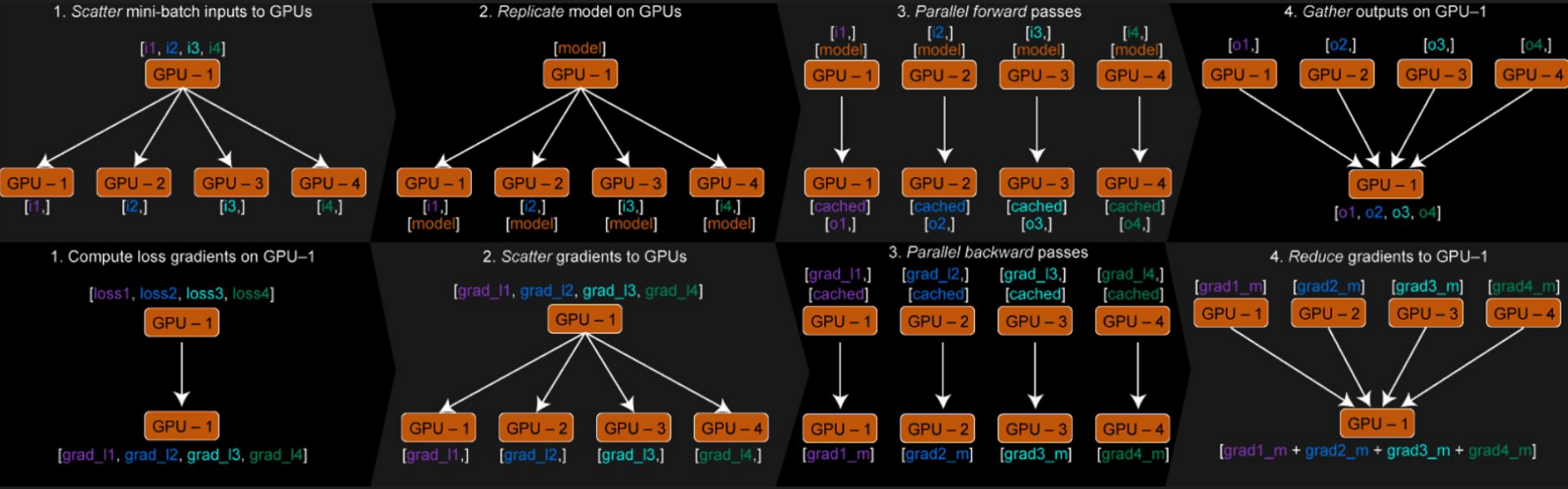
```
==========================================
Total params: 12,102,220
Trainable params: 12,102,220
Non-trainable params: 0
------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 0.02
Params size (MB): 46.17
Estimated Total Size (MB): 46.23
------------------------------------------
```




```
==========================================
Total params: 376,200
Trainable params: 376,200
Non-trainable params: 0
------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 12.07
Params size (MB): 1.44
Estimated Total Size (MB): 13.55
------------------------------------------
```

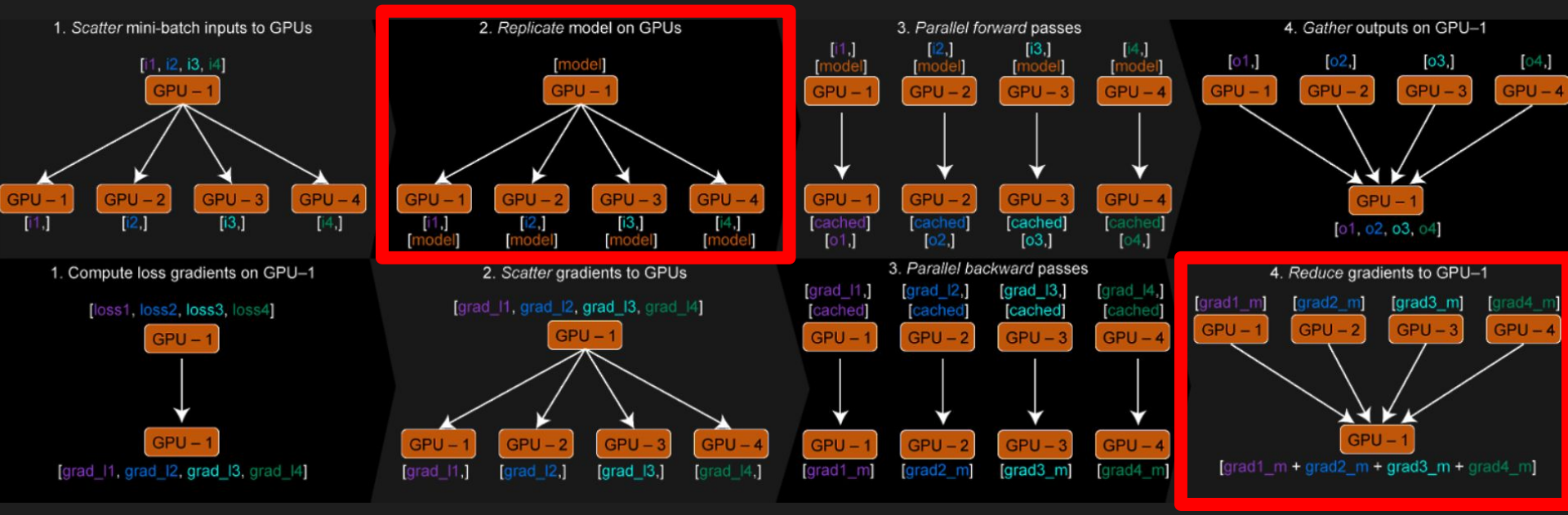


How DataParallel() Works



[3]

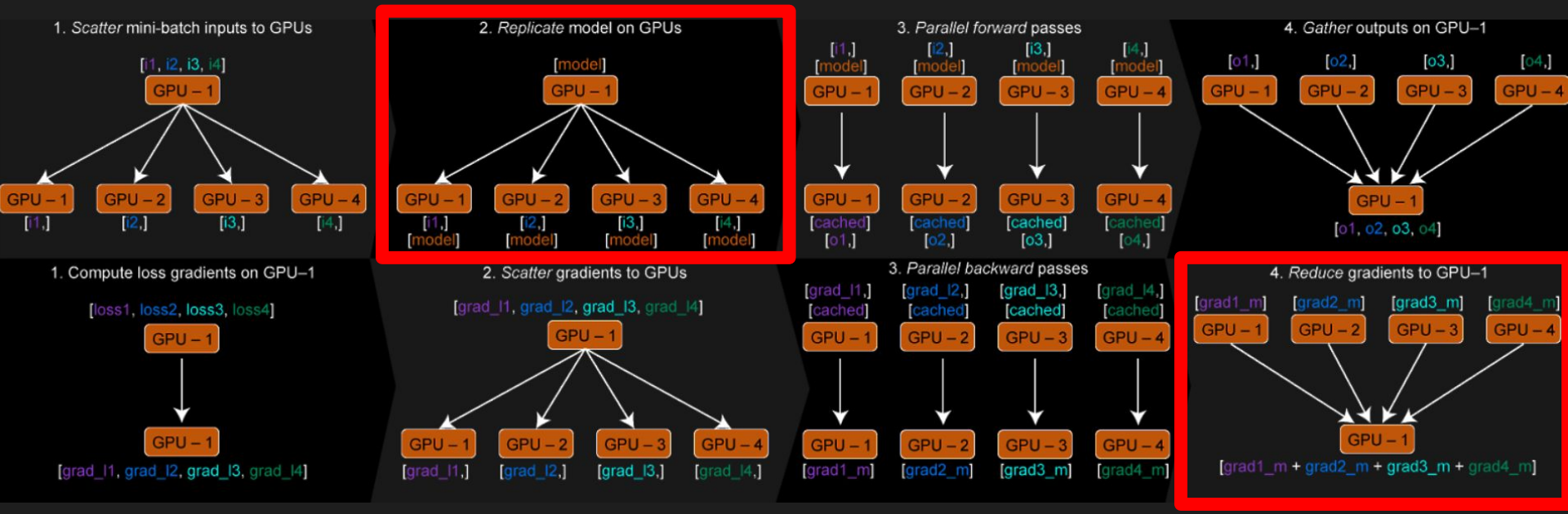
How DataParallel() Works



Assuming independent channels for each GPU-to-GPU communication:
All model parameters have to be transferred twice each batch!

[3]

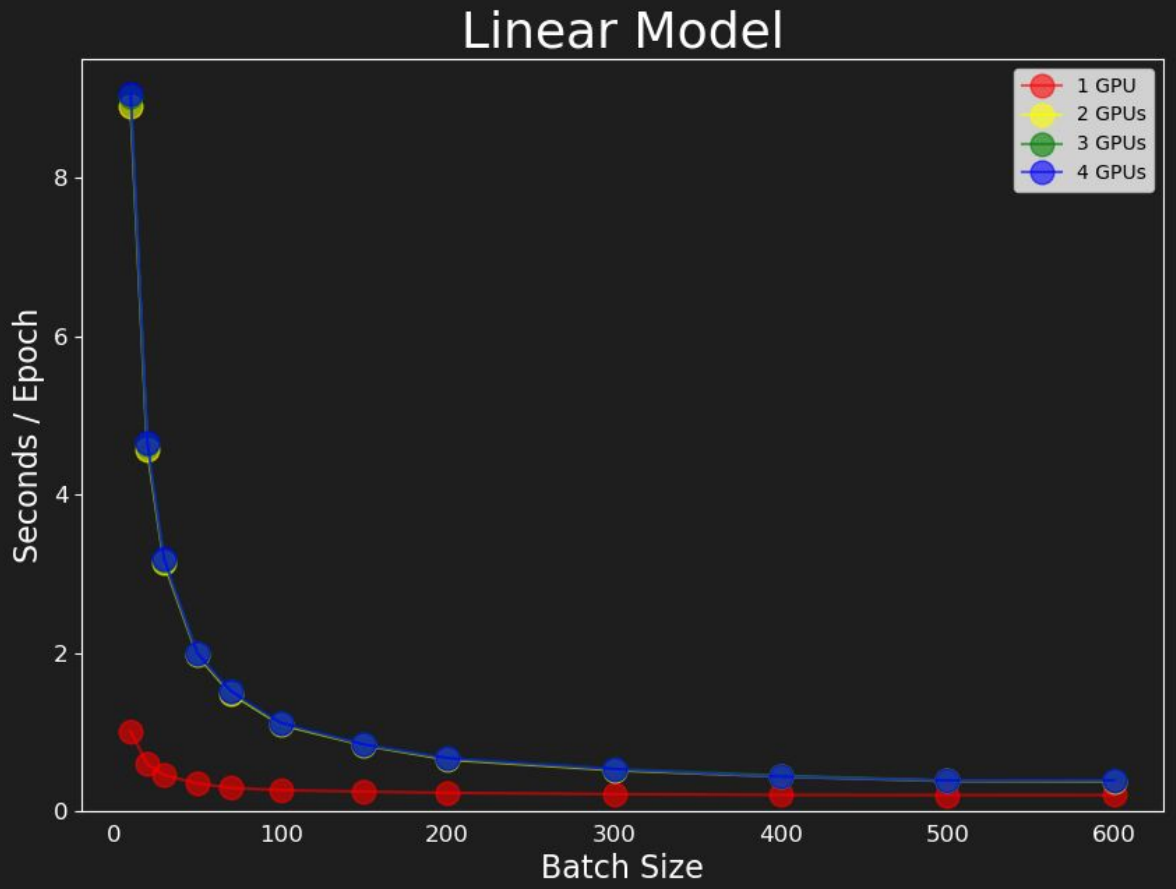
Linear Model: Parameter size ~ 50 MB
If the batch size is 20 then batches / epoch = 100



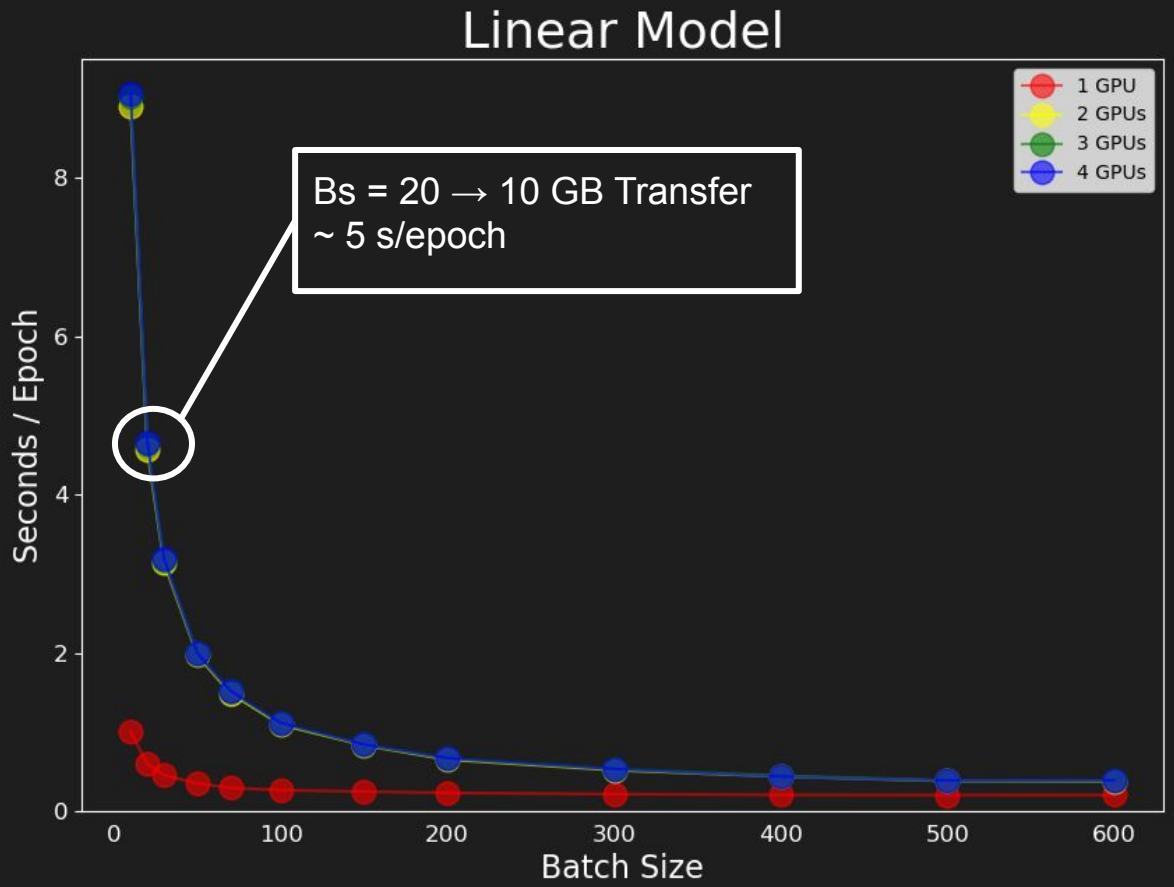
50MB x 2 transfers/batch x 100 batches/epoch = 10GB (transfer/epoch)

[3]

```
==========================================
Total params: 12,102,220
Trainable params: 12,102,220
Non-trainable params: 0
------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 0.02
Params size (MB): 46.17
Estimated Total Size (MB): 46.23
------------------------------------------
```

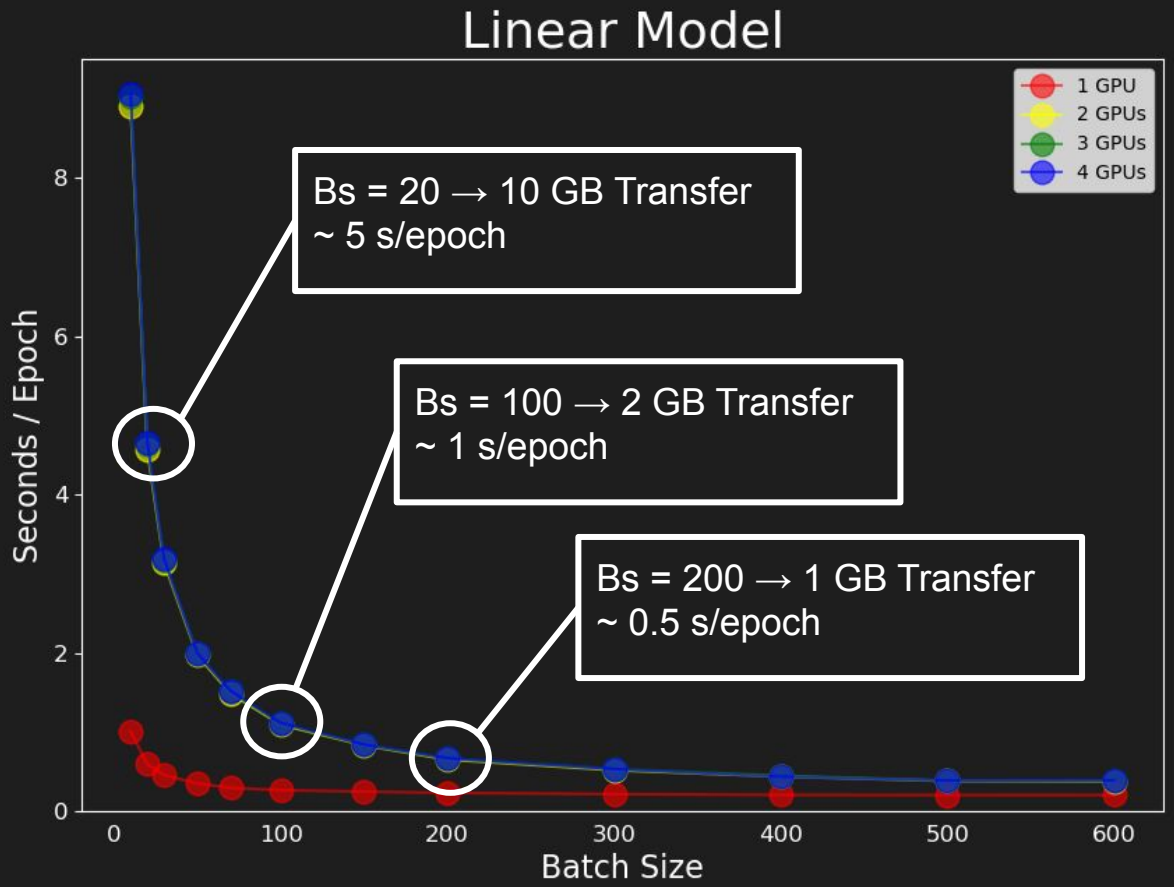


```
==========================================
Total params: 12,102,220
Trainable params: 12,102,220
Non-trainable params: 0
------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 0.02
Params size (MB): 46.17
Estimated Total Size (MB): 46.23
------------------------------------------
```




```
batch_sizes = [10,20,30,50,70,100,150,200,300,400,500,600]
```

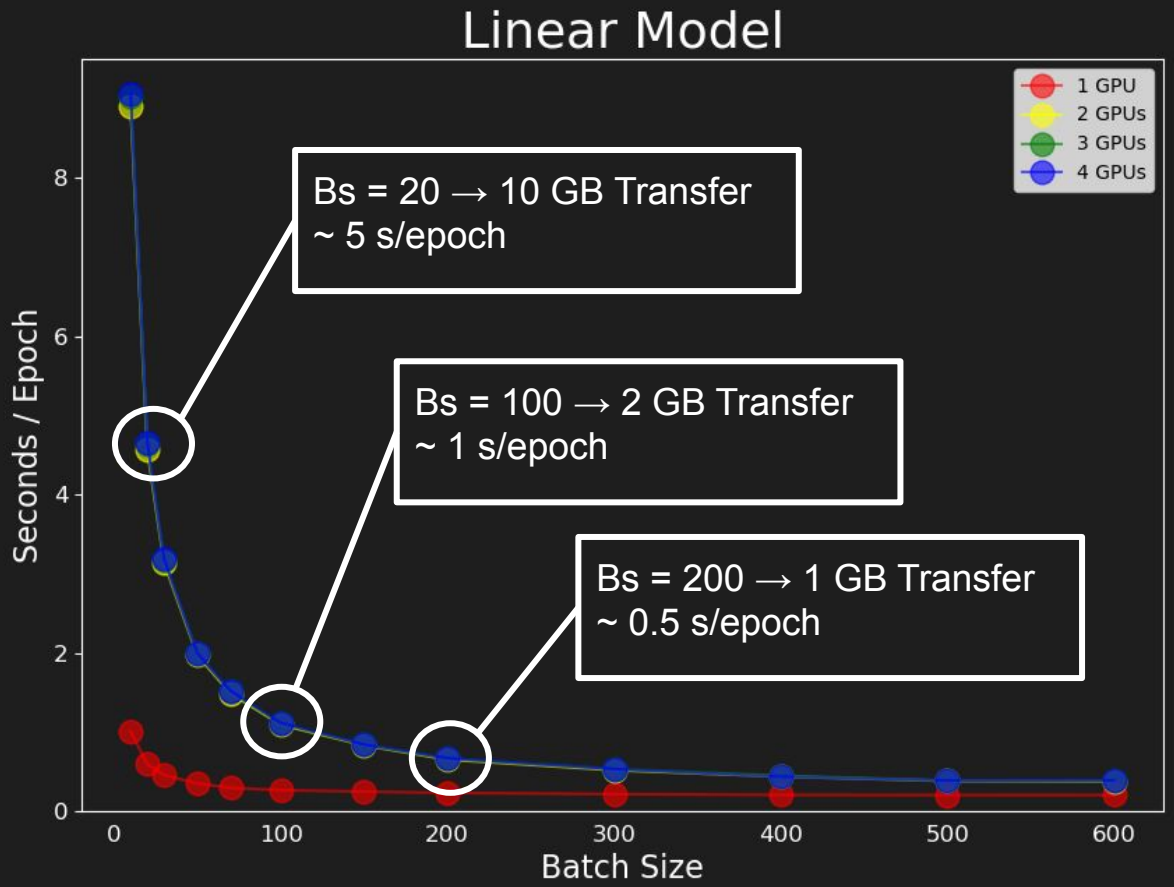
```
==========================================
Total params: 12,102,220
Trainable params: 12,102,220
Non-trainable params: 0
------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 0.02
Params size (MB): 46.17
Estimated Total Size (MB): 46.23
------------------------------------------
```




```
batch_sizes = [10,20,30,50,70,100,150,200,300,400,500,600]
```

Linear Relationship:
Time = (const)xTransfer

```
==========================================
Total params: 12,102,220
Trainable params: 12,102,220
Non-trainable params: 0
------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 0.02
Params size (MB): 46.17
Estimated Total Size (MB): 46.23
------------------------------------------
```

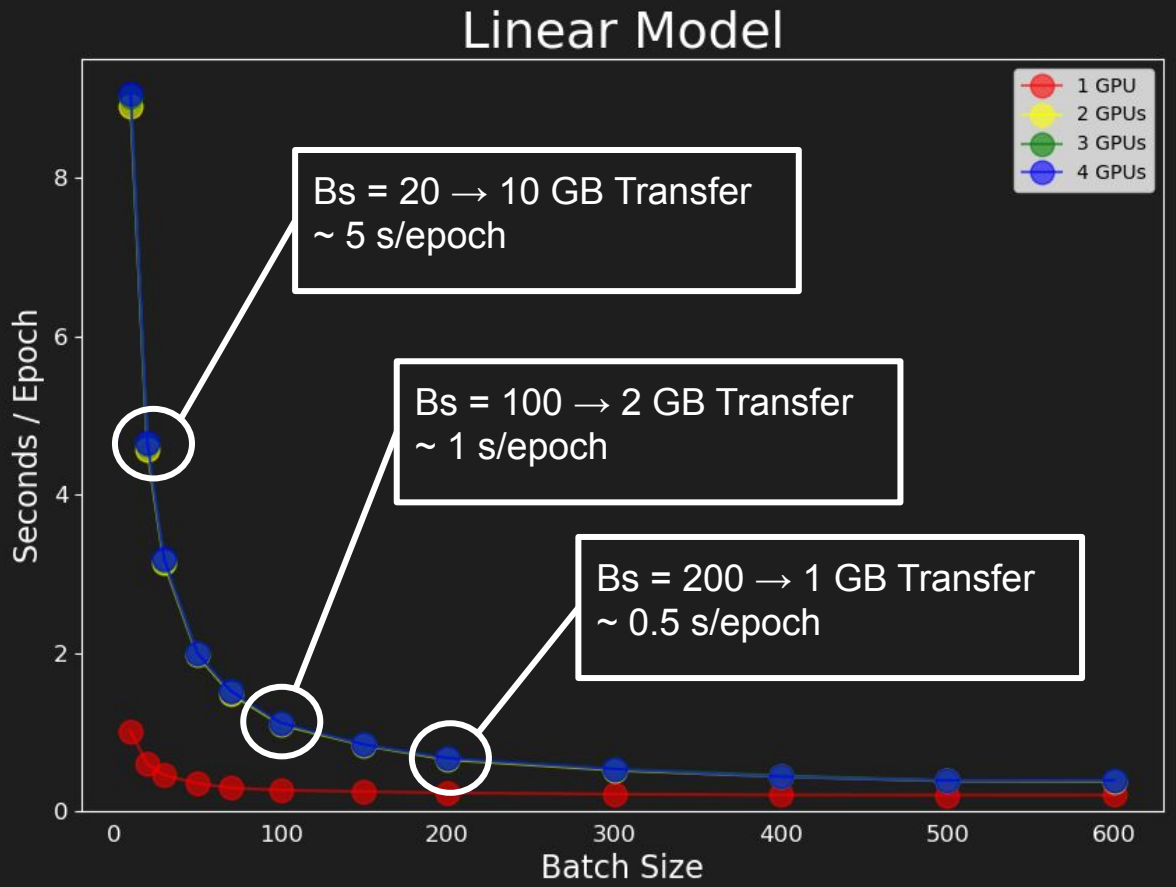



```
batch_sizes = [10,20,30,50,70,100,150,200,300,400,500,600]
```

Linear Relationship:
Time = Const x Transfer

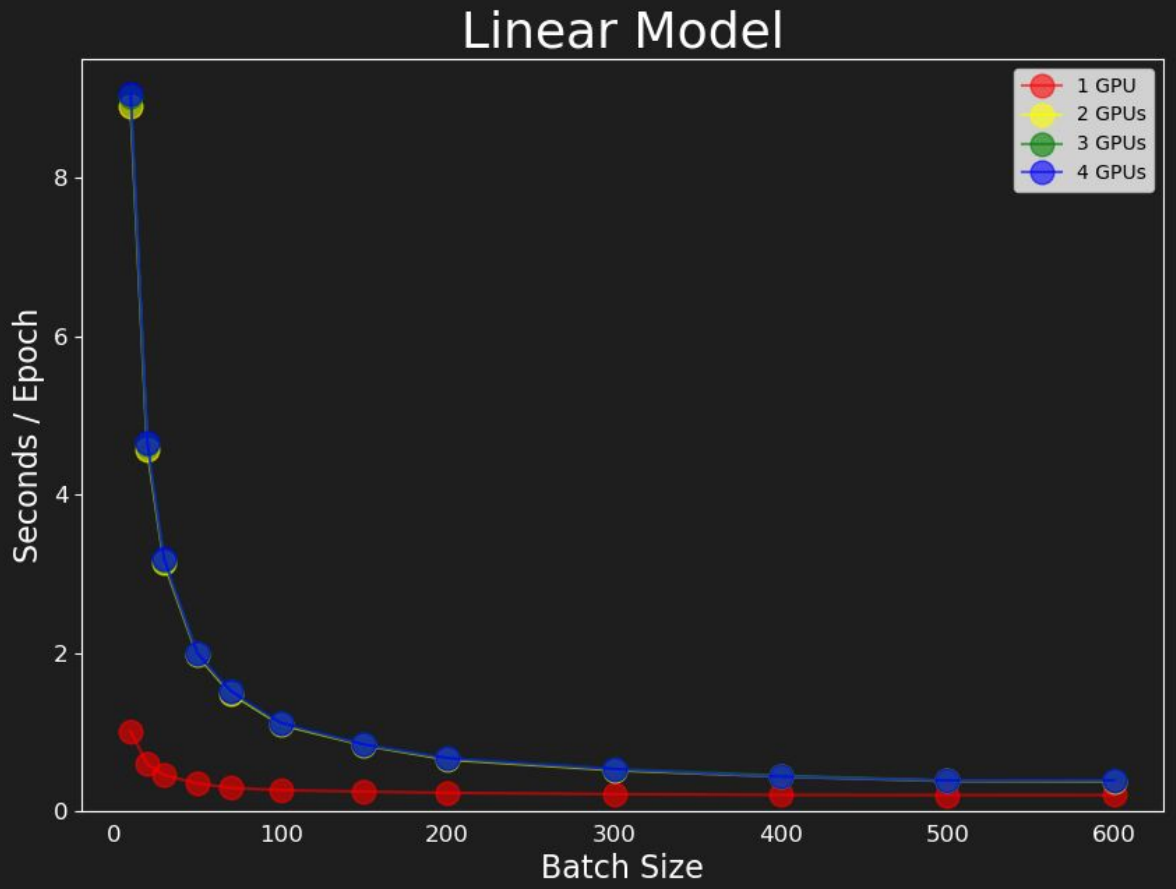
```
==========================================
Total params: 12,102,220
Trainable params: 12,102,220
Non-trainable params: 0
------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 0.02
Params size (MB): 46.17
Estimated Total Size (MB): 46.23
------------------------------------------
```

NETWORK CHOKE

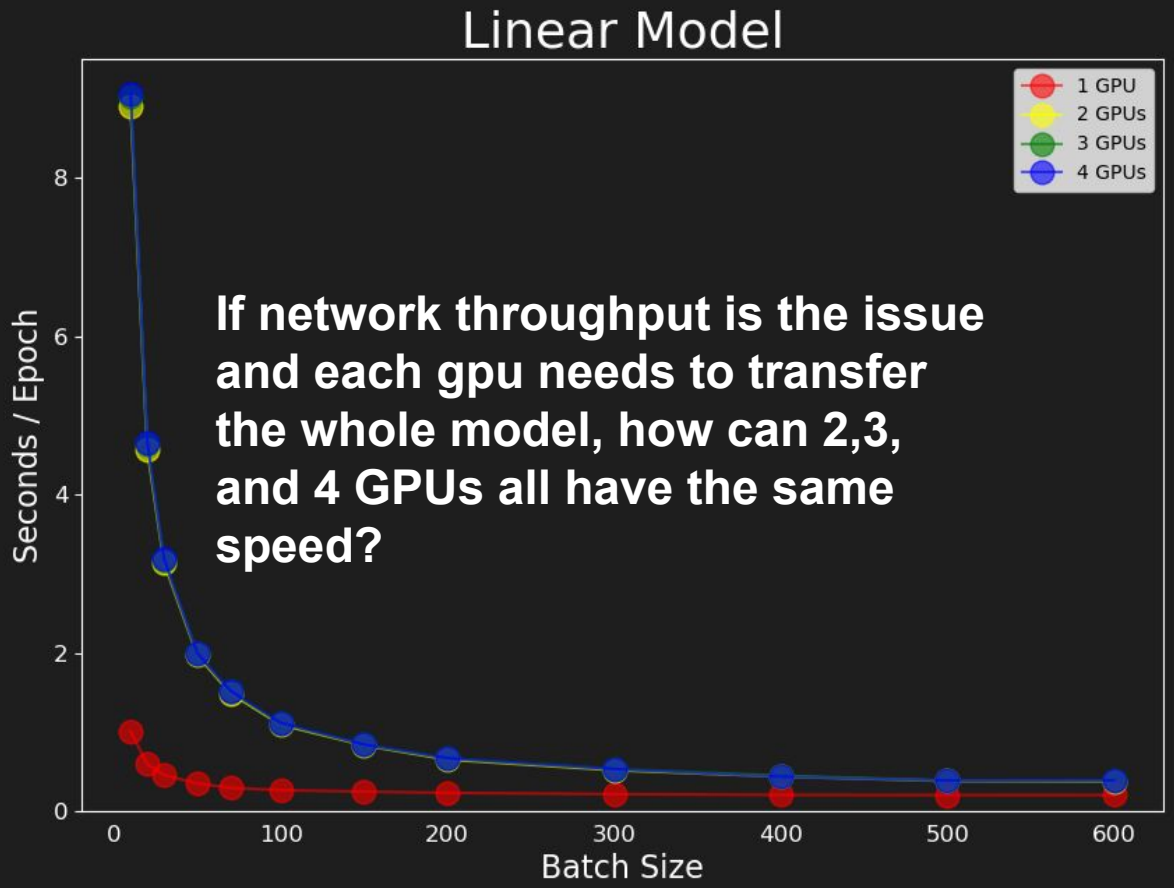



```
batch_sizes = [10,20,30,50,70,100,150,200,300,400,500,600]
```

```
==========================================
Total params: 12,102,220
Trainable params: 12,102,220
Non-trainable params: 0
------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 0.02
Params size (MB): 46.17
Estimated Total Size (MB): 46.23
------------------------------------------
```



```
==========================================
Total params: 12,102,220
Trainable params: 12,102,220
Non-trainable params: 0
------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 0.02
Params size (MB): 46.17
Estimated Total Size (MB): 46.23
------------------------------------------
```



If network throughput is the issue and each gpu needs to transfer the whole model, how can 2,3, and 4 GPUs all have the same speed?

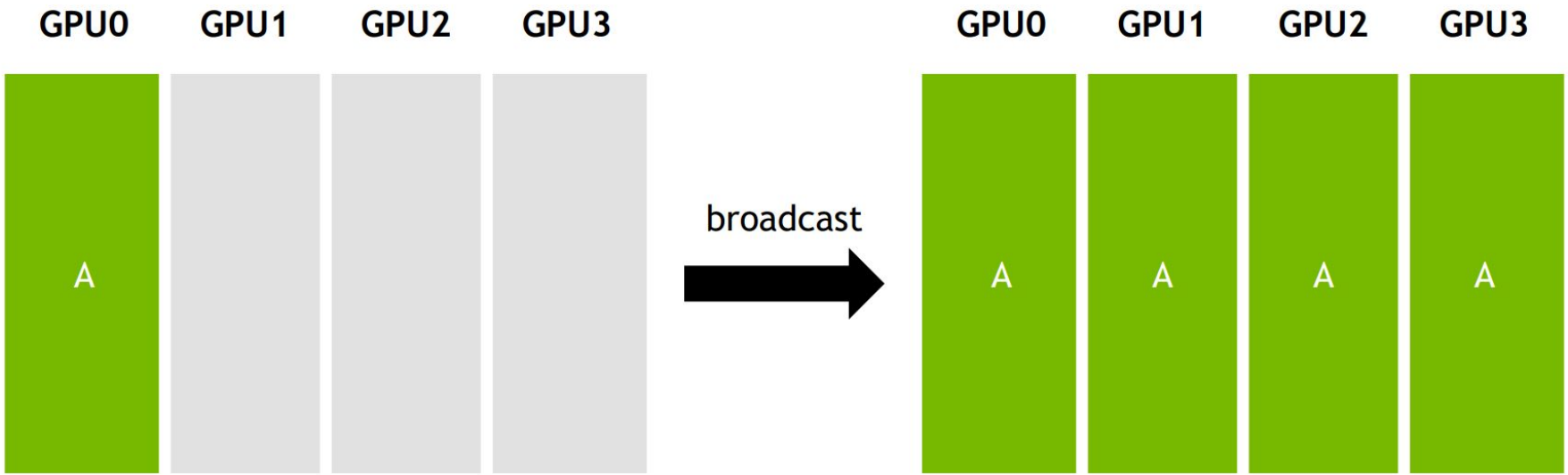
```
ubuntu@96-76-203-29:~$ nvidia-smi topo -m
        GPU0    GPU1    GPU2    GPU3    CPU Affinity
GPU0     X      PHB     PHB     PHB     0-7
GPU1    PHB      X      PHB     PHB     0-7
GPU2    PHB     PHB      X      PHB     0-7
GPU3    PHB     PHB     PHB      X      0-7

Legend:

  X    = Self
  SOC  = Connection traversing PCIe as well as the SMP link between CPU sockets(e.g. QPI)
  PHB  = Connection traversing PCIe as well as a PCIe Host Bridge (typically the CPU)
  PXB  = Connection traversing multiple PCIe switches (without traversing the PCIe Host Bridge)
  PIX  = Connection traversing a single PCIe switch
  NV#  = Connection traversing a bonded set of # NVLinks
```

NCCL: ACCELERATED MULTI-GPU COLLECTIVE COMMUNICATIONS

Cliff Woolley, Sr. Manager, Developer Technology Software, NVIDIA

BROADCAST

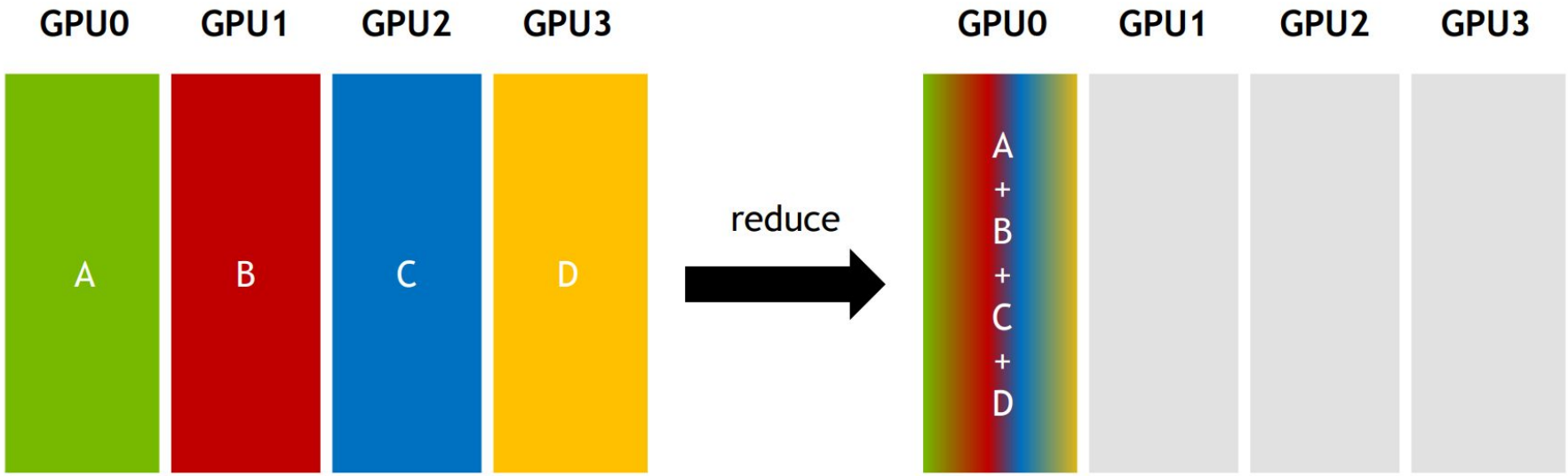
One sender, multiple receivers



[5]

REDUCE

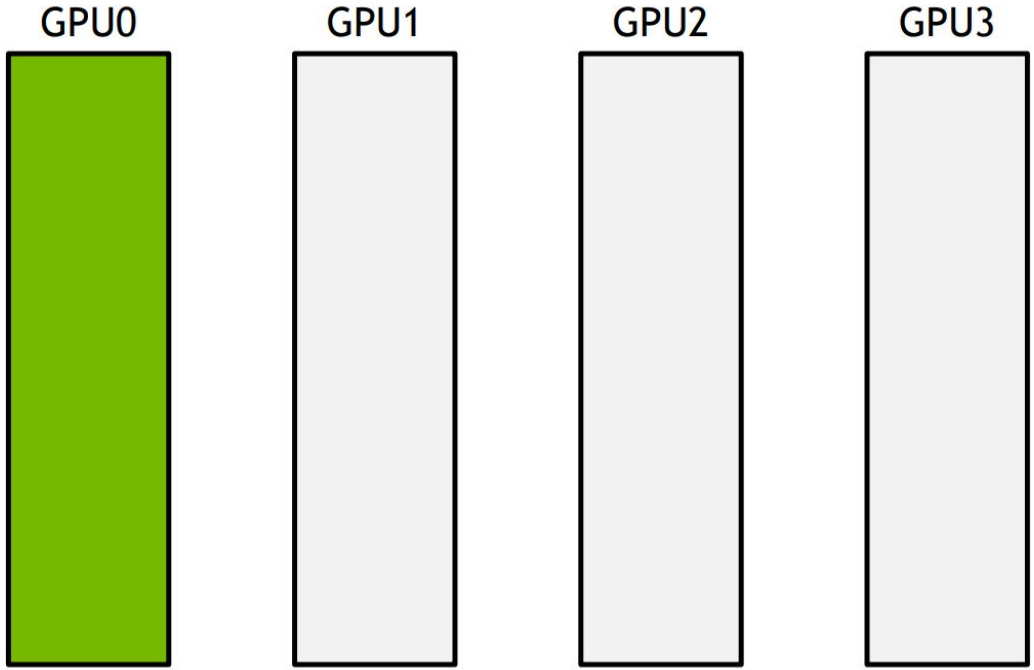
Combine data from all senders; deliver the result to one receiver



[5]

BROADCAST

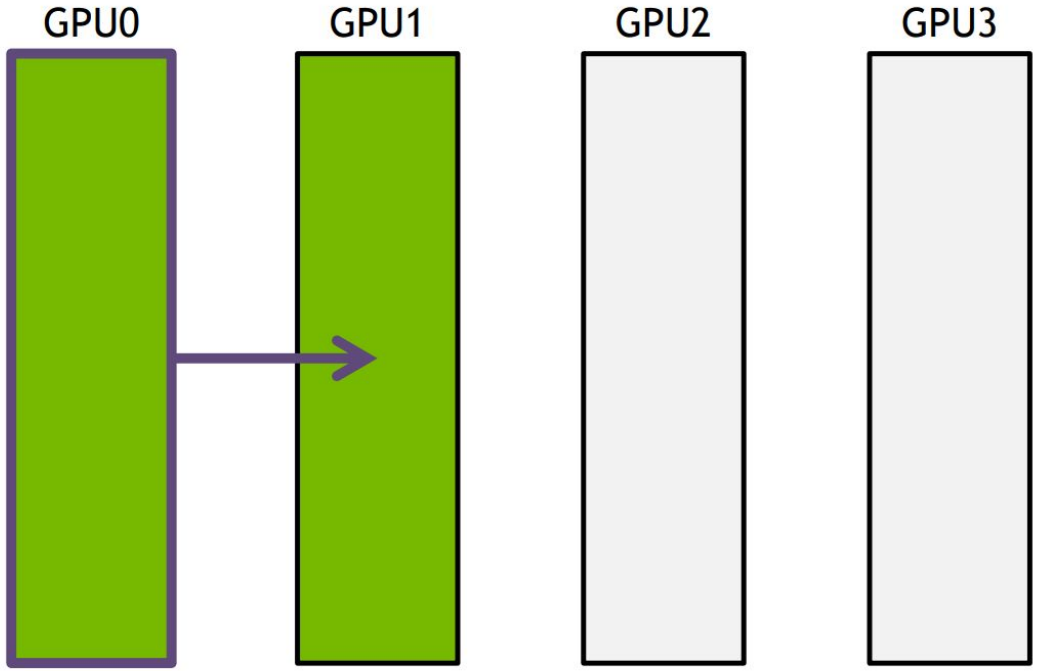
with unidirectional ring



[5]

BROADCAST

with unidirectional ring



Step 1: $\Delta t = N/B$

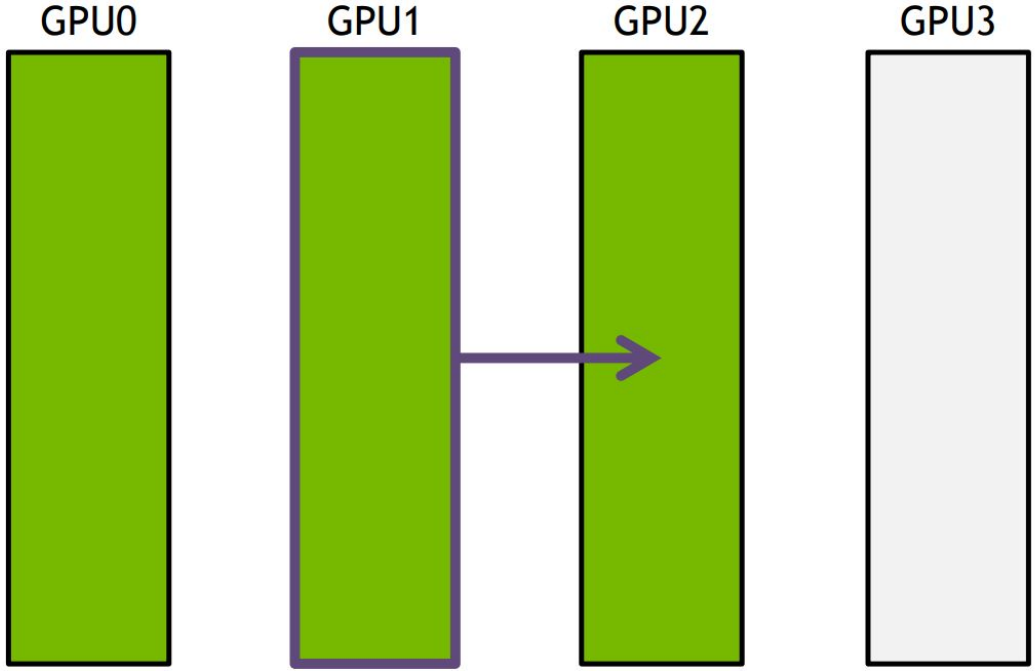
N: bytes to broadcast

B: bandwidth of each link

[5]

BROADCAST

with unidirectional ring



Step 1: $\Delta t = N/B$

Step 2: $\Delta t = N/B$

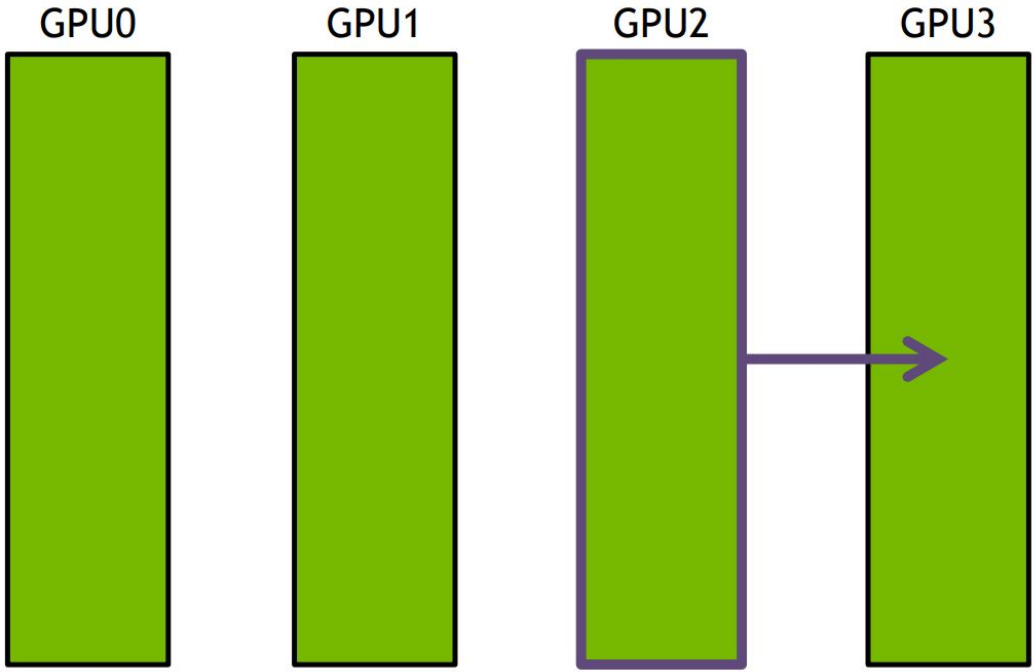
N: bytes to broadcast

B: bandwidth of each link

[5]

BROADCAST

with unidirectional ring



Step 1: $\Delta t = N/B$

Step 2: $\Delta t = N/B$

Step 3: $\Delta t = N/B$

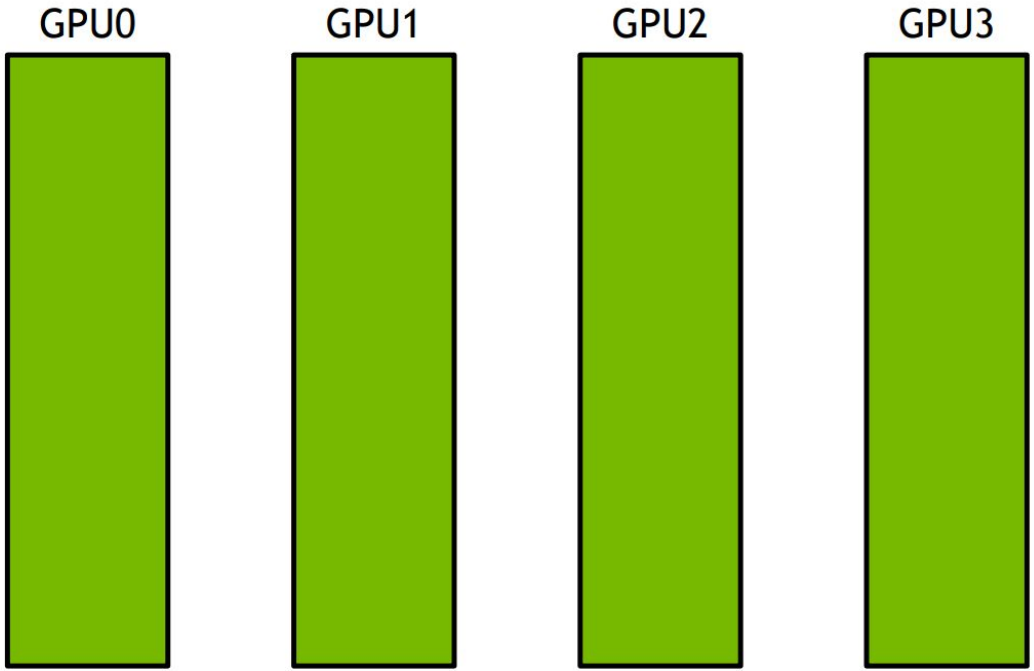
N: bytes to broadcast

B: bandwidth of each link

[5]

BROADCAST

with unidirectional ring



[5]

Step 1: $\Delta t = N/B$

Step 2: $\Delta t = N/B$

Step 3: $\Delta t = N/B$

Total time: $(k-1)N/B$

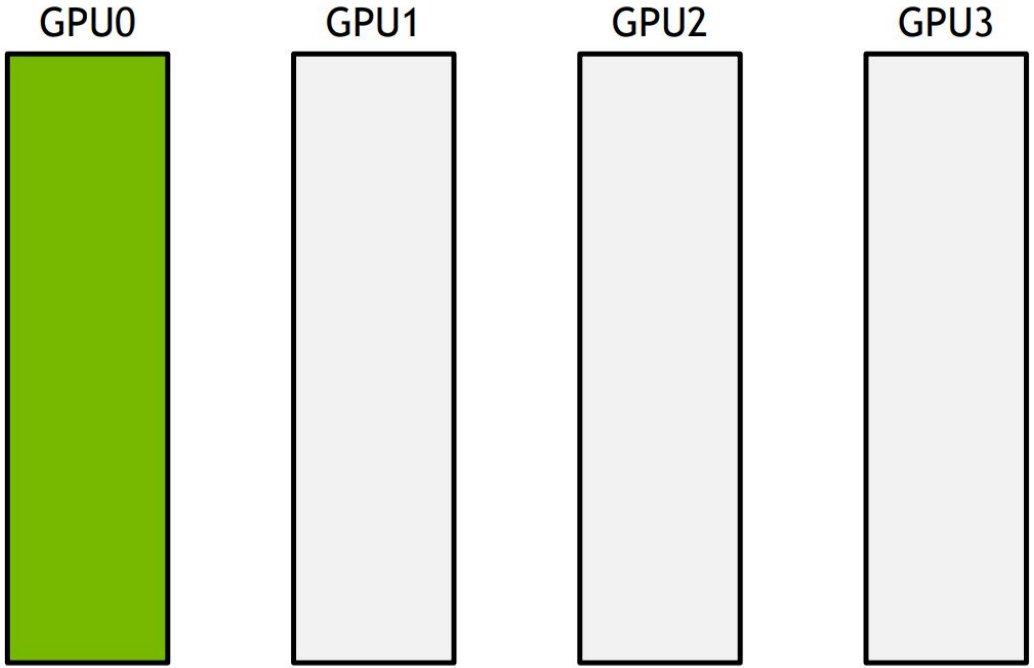
N: bytes to broadcast

B: bandwidth of each link

k: number of GPUs

BROADCAST

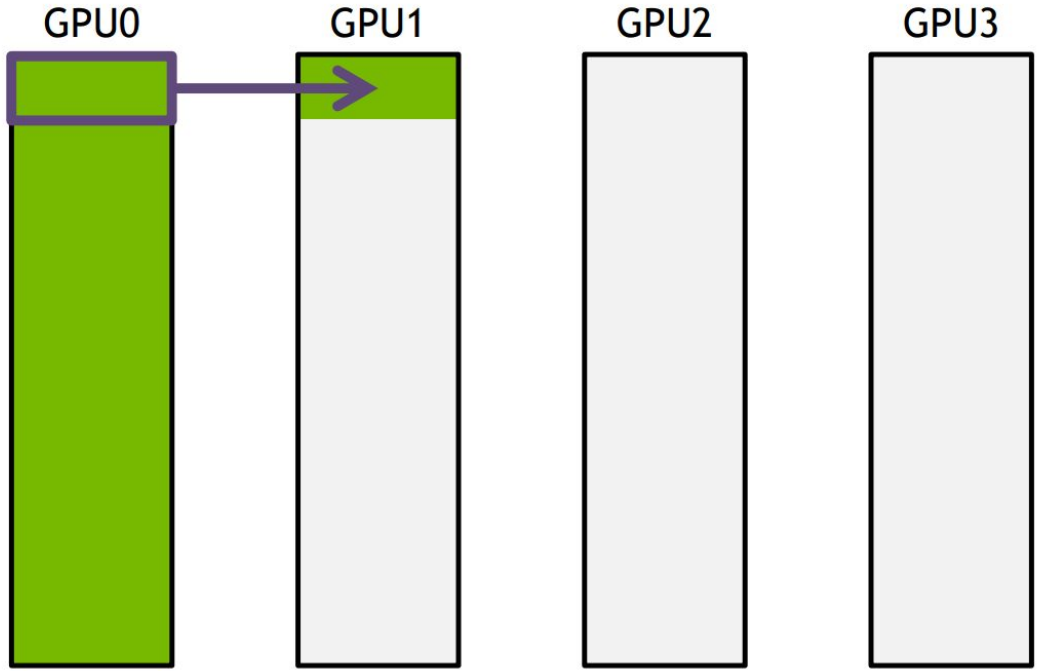
with unidirectional ring



[5]

BROADCAST

with unidirectional ring



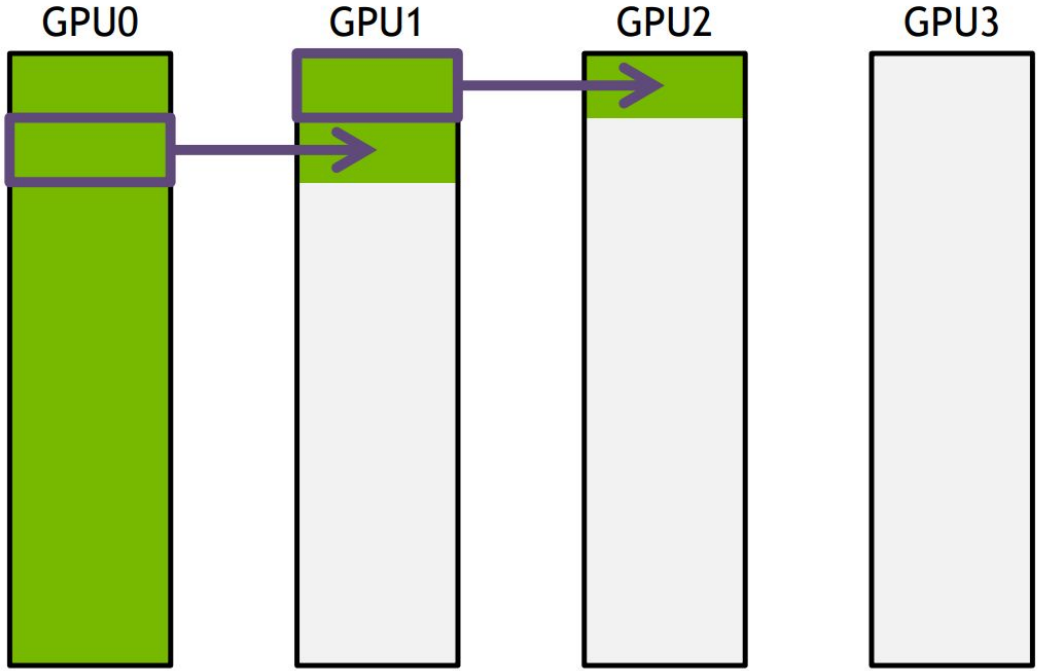
Split data into S messages

Step 1: $\Delta t = N/(SB)$

[5]

BROADCAST

with unidirectional ring



Split data into S messages

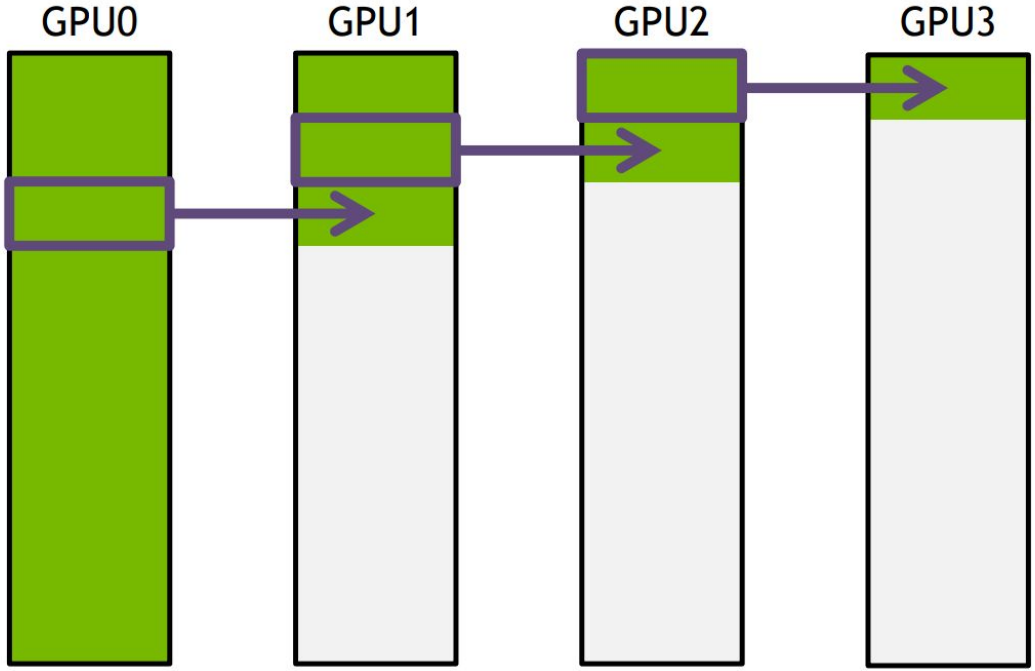
Step 1: $\Delta t = N/(SB)$

Step 2: $\Delta t = N/(SB)$

[5]

BROADCAST

with unidirectional ring



Split data into S messages

Step 1: $\Delta t = N/(SB)$

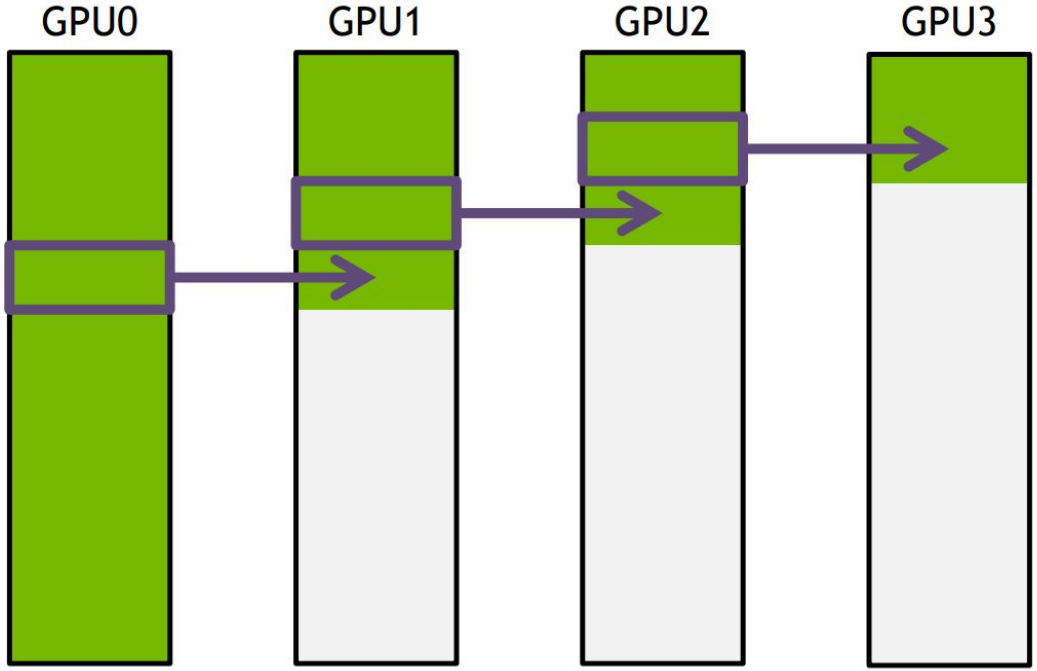
Step 2: $\Delta t = N/(SB)$

Step 3: $\Delta t = N/(SB)$

[5]

BROADCAST

with unidirectional ring



[5]

Split data into S messages

Step 1: $\Delta t = N/(SB)$

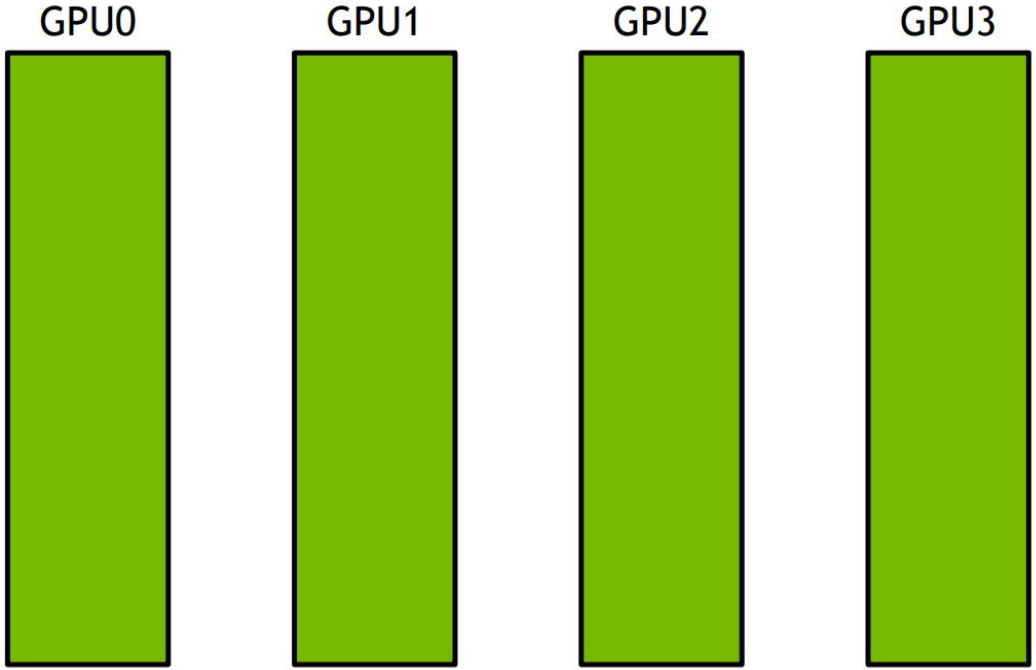
Step 2: $\Delta t = N/(SB)$

Step 3: $\Delta t = N/(SB)$

Step 4: $\Delta t = N/(SB)$

BROADCAST

with unidirectional ring



[5]

Split data into S messages

Step 1: $\Delta t = N/(SB)$

Step 2: $\Delta t = N/(SB)$

Step 3: $\Delta t = N/(SB)$

Step 4: $\Delta t = N/(SB)$

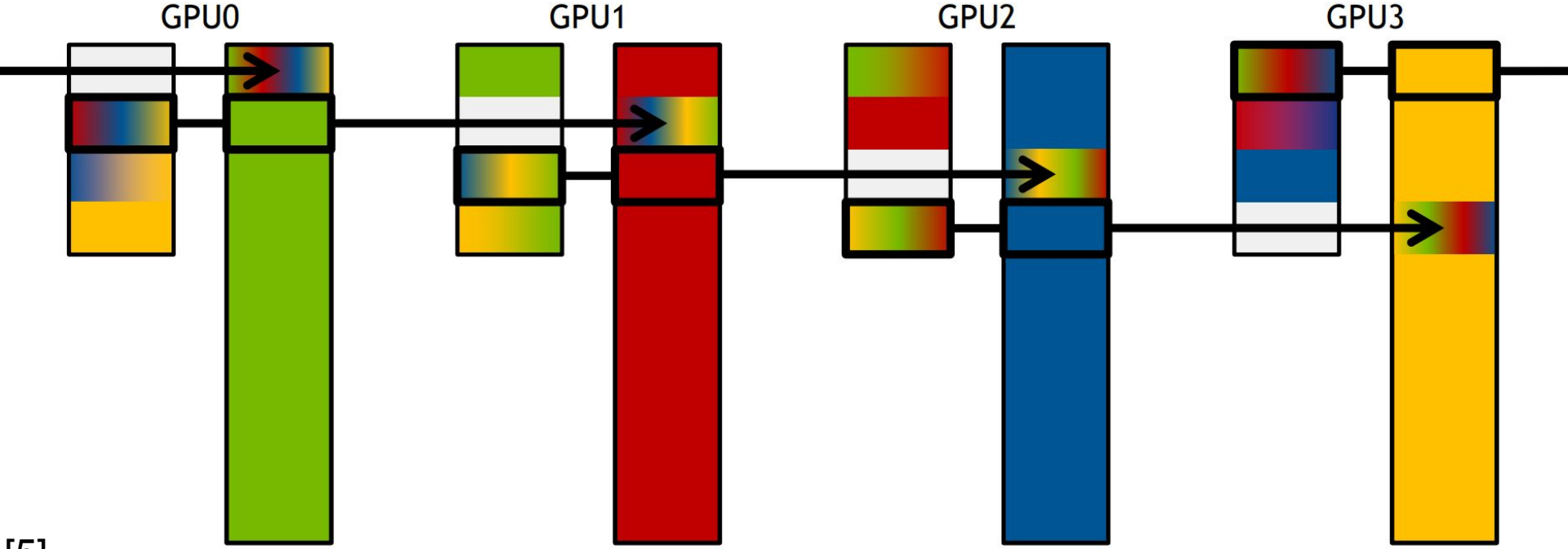
…

Total time:
$SN/(SB) + (k-2)\,N/(SB)$
$= N(S+k-2)/(SB) \rightarrow N/B$

ALL-REDUCE

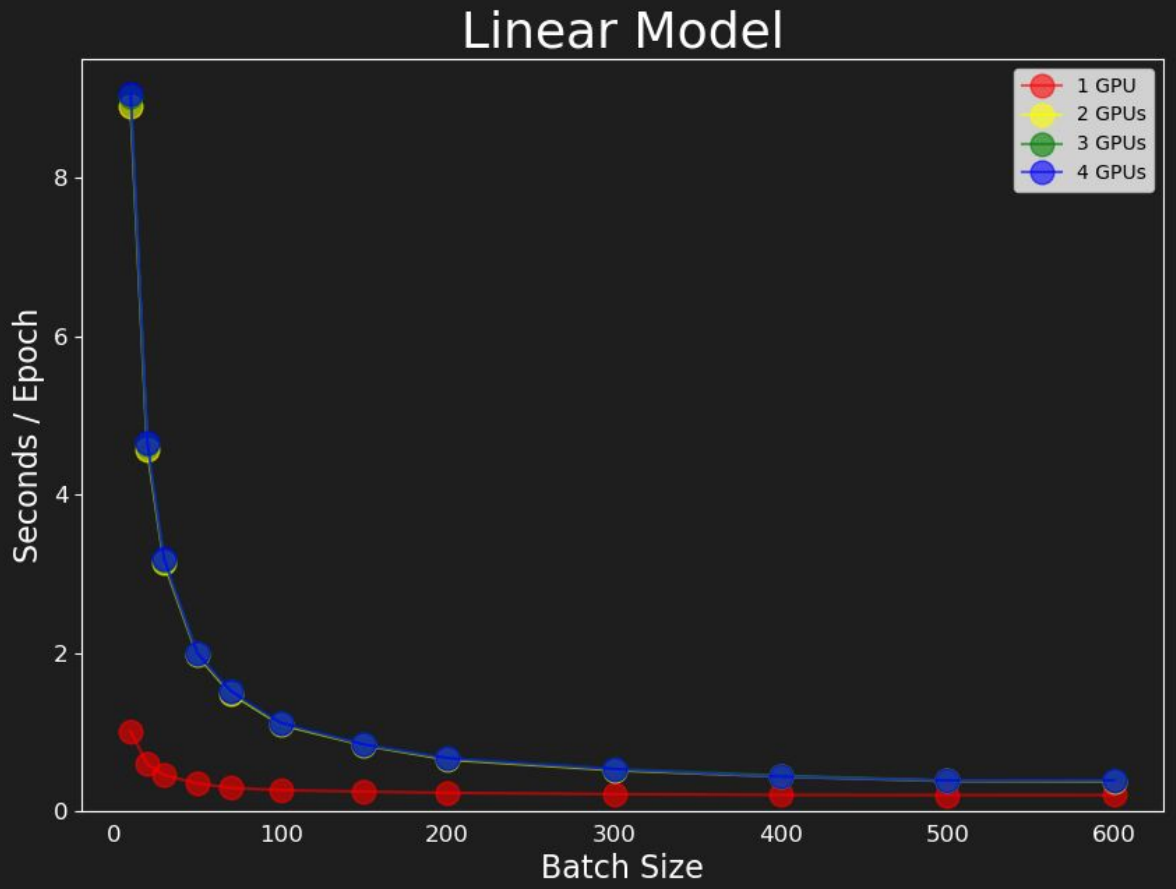
with unidirectional ring

Chunk: 1
Step: 4

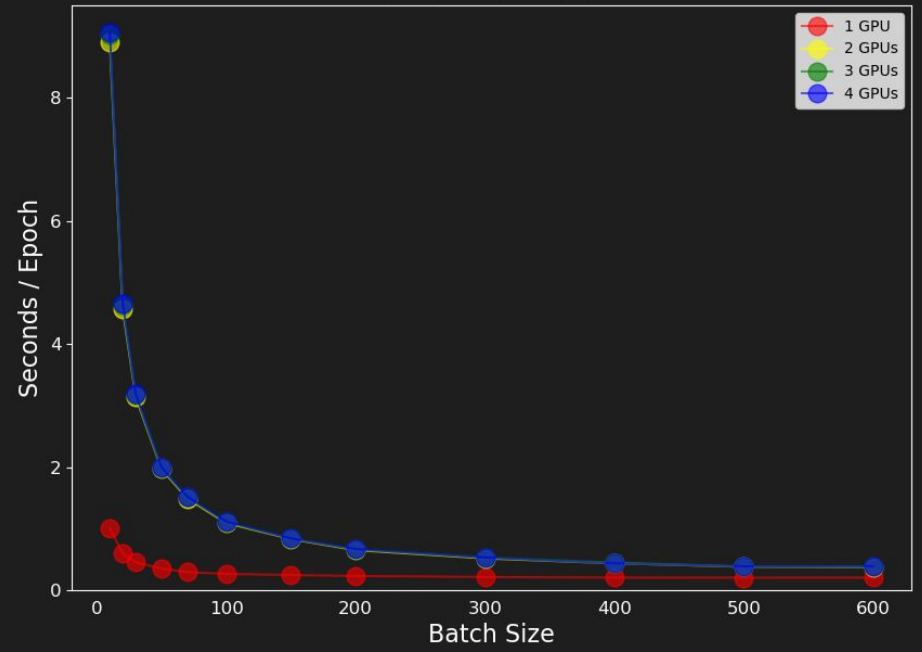


[5]

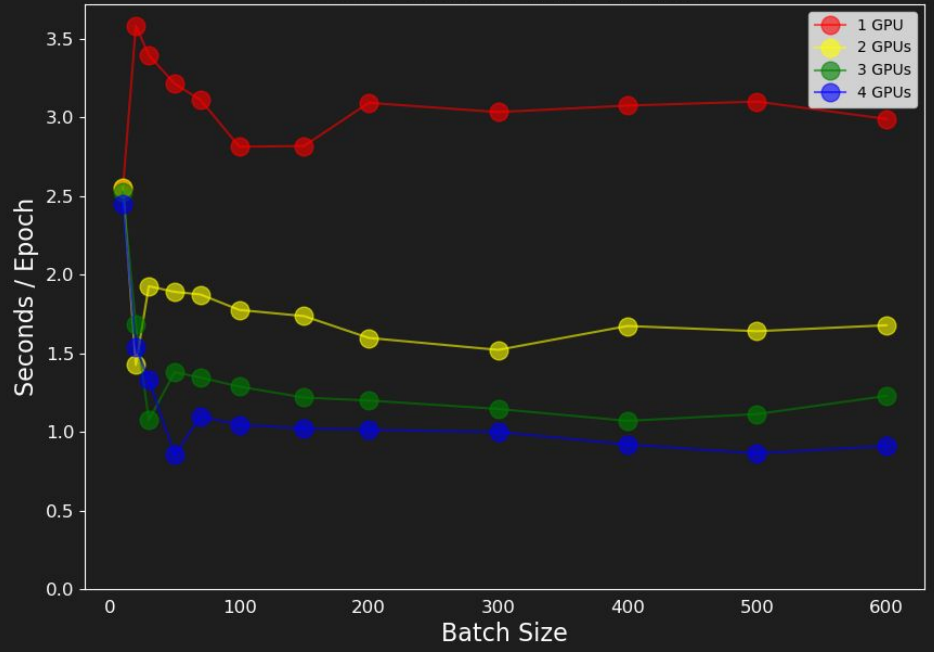
```
==========================================
Total params: 12,102,220
Trainable params: 12,102,220
Non-trainable params: 0
------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 0.02
Params size (MB): 46.17
Estimated Total Size (MB): 46.23
------------------------------------------
```



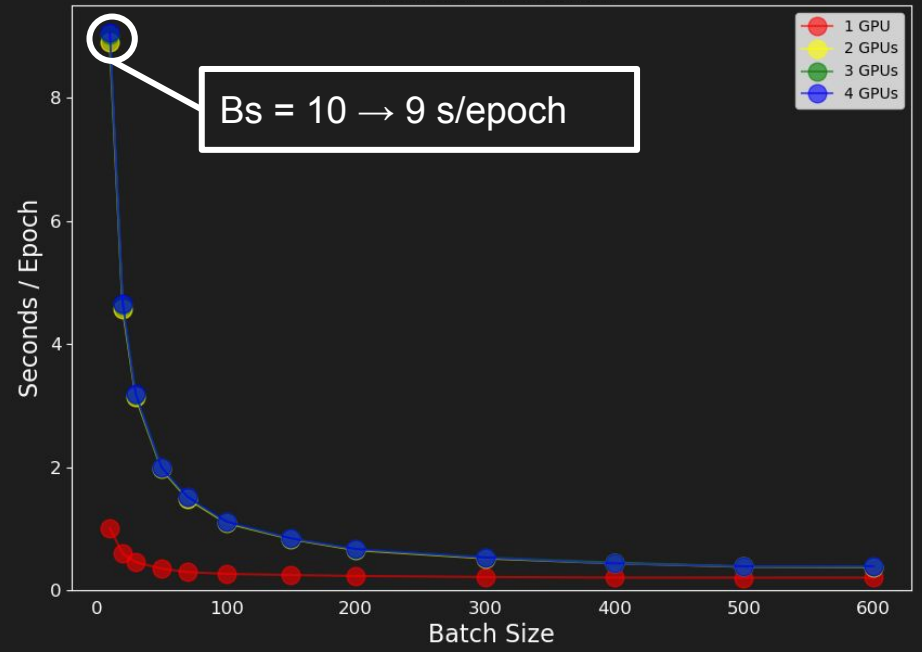
```
==========================================
Total params: 12,102,220
Trainable params: 12,102,220
Non-trainable params: 0
------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 0.02
Params size (MB): 46.17
Estimated Total Size (MB): 46.23
------------------------------------------
```



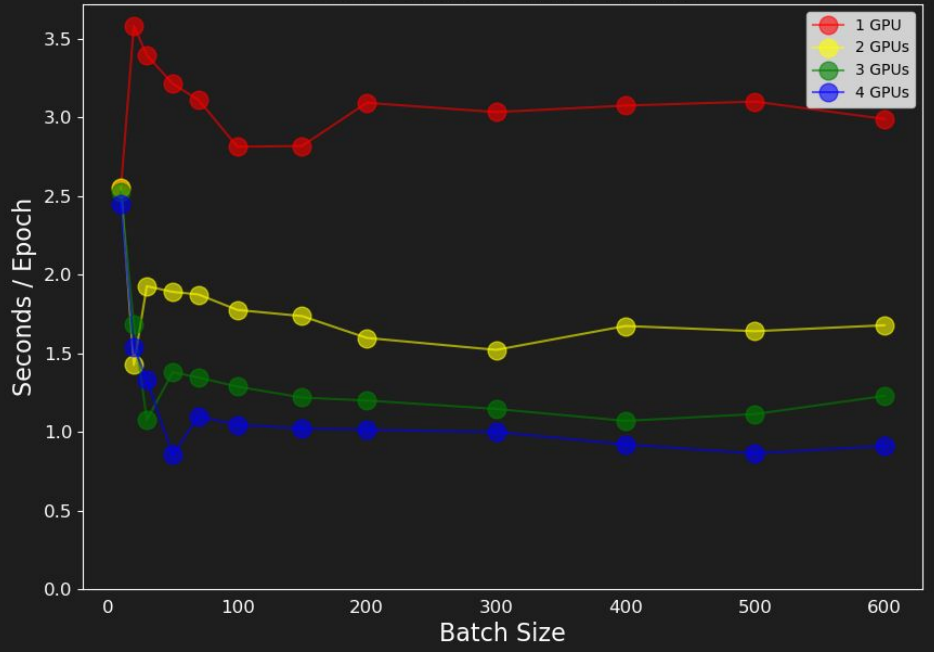

```
==========================================
Total params: 376,200
Trainable params: 376,200
Non-trainable params: 0
------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 12.07
Params size (MB): 1.44
Estimated Total Size (MB): 13.55
------------------------------------------
```



```
==========================================
Total params: 12,102,220
Trainable params: 12,102,220
Non-trainable params: 0
------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 0.02
Params size (MB): 46.17
Estimated Total Size (MB): 46.23
------------------------------------------
```

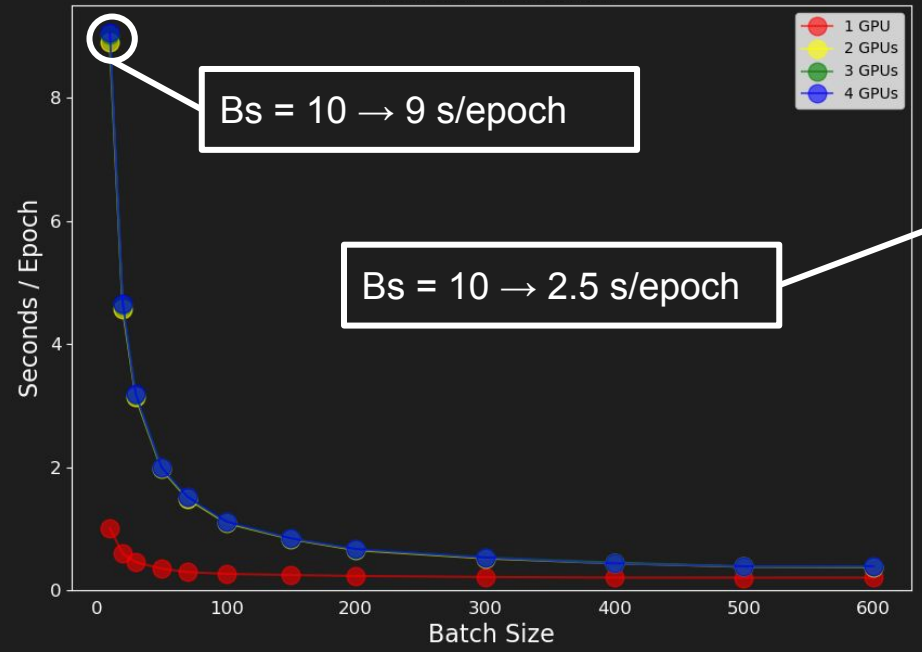


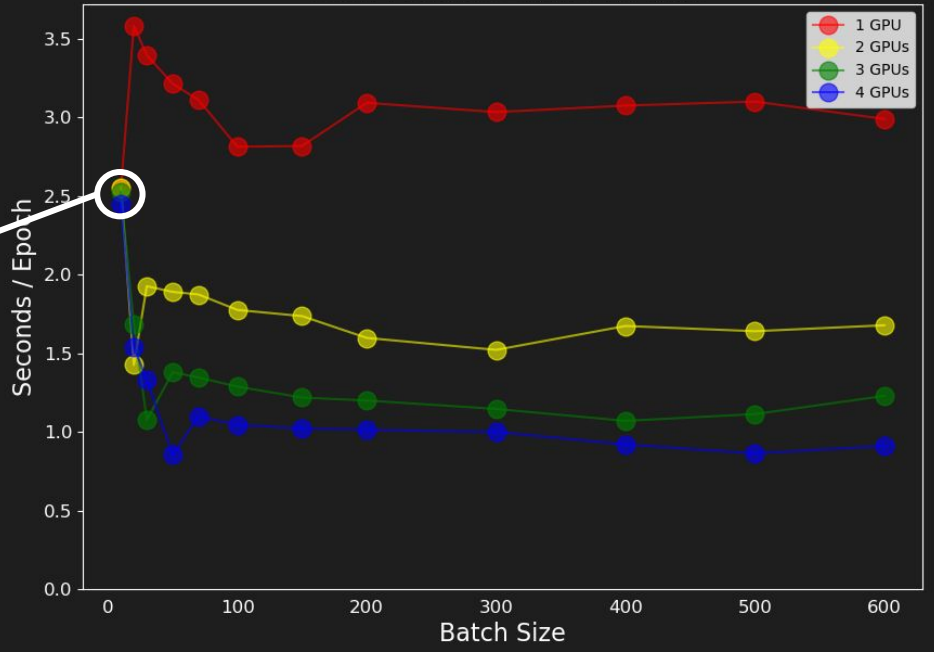

```
==========================================
Total params: 376,200
Trainable params: 376,200
Non-trainable params: 0
------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 12.07
Params size (MB): 1.44
Estimated Total Size (MB): 13.55
------------------------------------------
```



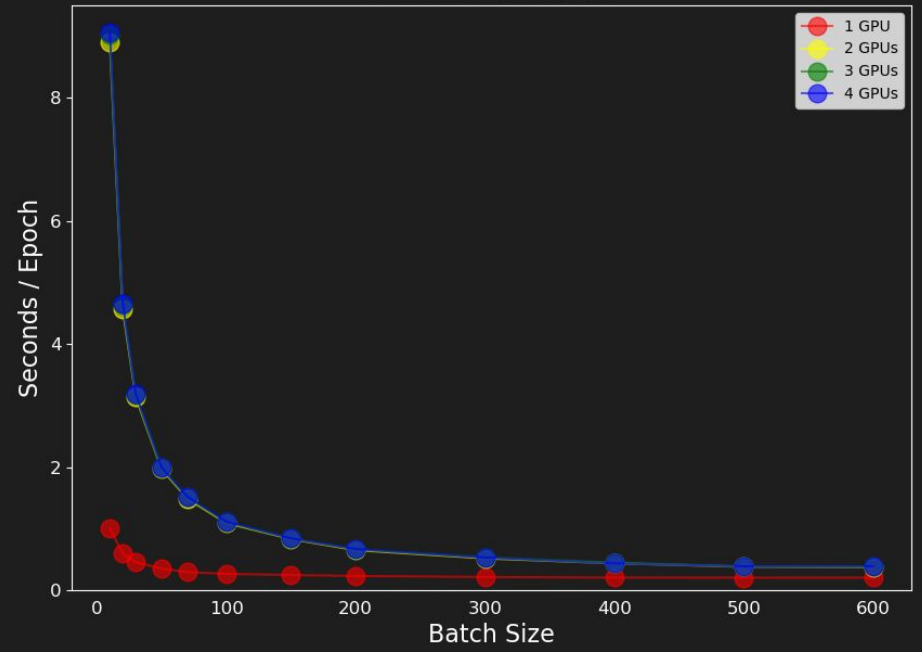
```
==========================================
Total params: 12,102,220
Trainable params: 12,102,220
Non-trainable params: 0
------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 0.02
Params size (MB): 46.17
Estimated Total Size (MB): 46.23
------------------------------------------
```

```
==========================================
Total params: 376,200
Trainable params: 376,200
Non-trainable params: 0
------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 12.07
Params size (MB): 1.44
Estimated Total Size (MB): 13.55
------------------------------------------
```

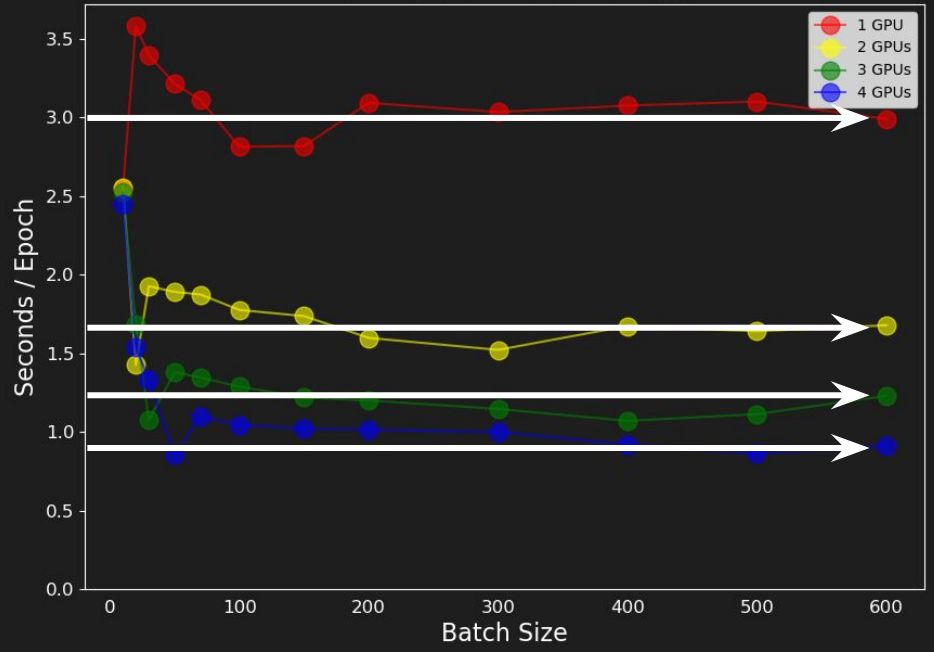




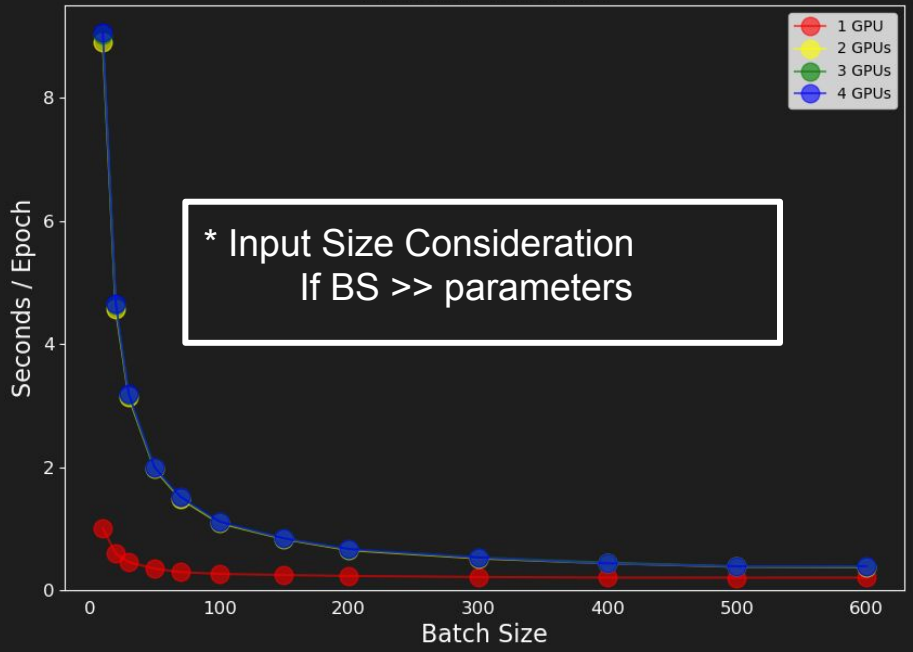
```
==========================================
Total params: 12,102,220
Trainable params: 12,102,220
Non-trainable params: 0
------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 0.02
Params size (MB): 46.17
Estimated Total Size (MB): 46.23
------------------------------------------
```




```
==========================================
Total params: 376,200
Trainable params: 376,200
Non-trainable params: 0
------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 12.07
Params size (MB): 1.44
Estimated Total Size (MB): 13.55
------------------------------------------
```

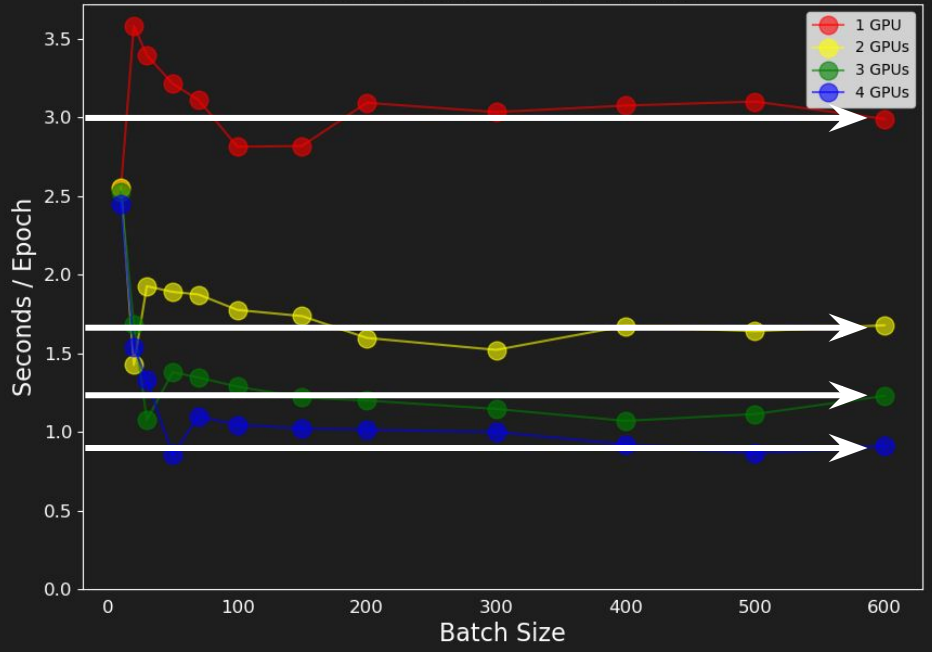


```
==========================================
Total params: 12,102,220
Trainable params: 12,102,220
Non-trainable params: 0
------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 0.02
Params size (MB): 46.17
Estimated Total Size (MB): 46.23
------------------------------------------
```

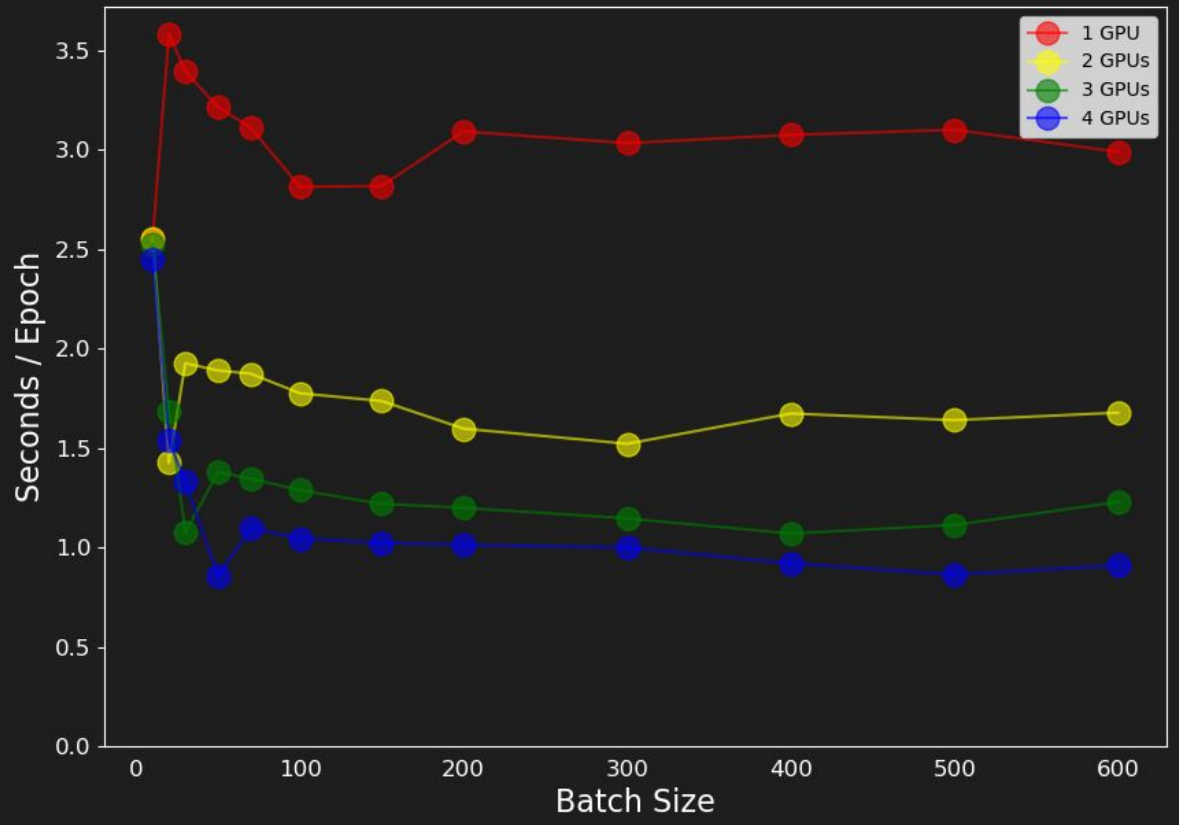


* Input Size Consideration
  If BS >> parameters

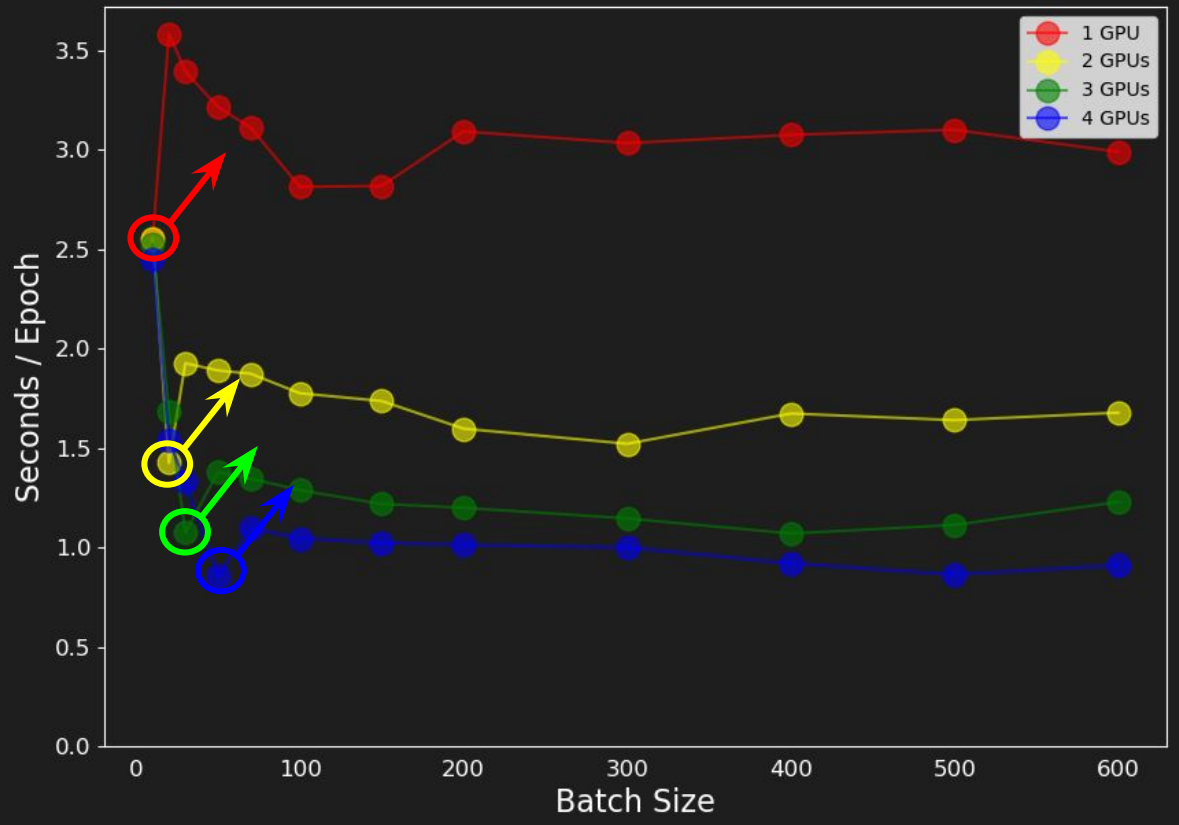
```
==========================================
Total params: 376,200
Trainable params: 376,200
Non-trainable params: 0
------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 12.07
Params size (MB): 1.44
Estimated Total Size (MB): 13.55
------------------------------------------
```



```
==========================================
Total params: 376,200
Trainable params: 376,200
Non-trainable params: 0
------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 12.07
Params size (MB): 1.44
Estimated Total Size (MB): 13.55
------------------------------------------
```



```
==========================================
Total params: 376,200
Trainable params: 376,200
Non-trainable params: 0
------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 12.07
Params size (MB): 1.44
Estimated Total Size (MB): 13.55
------------------------------------------
```



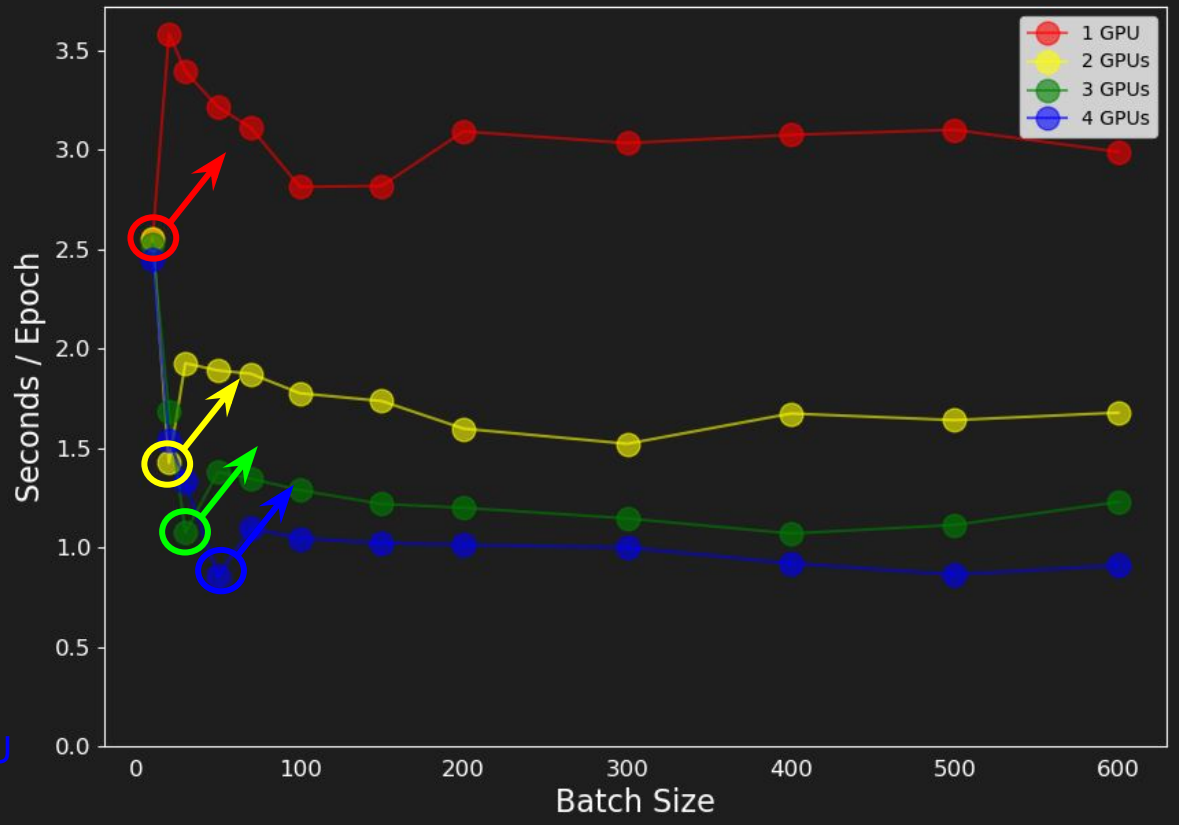
Bounce: Bs = 10 → 10 instances/GPU

Bounce: Bs = 20 → 10 instances/GPU

```
==========================================
Total params: 376,200
Trainable params: 376,200
Non-trainable params: 0
------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 12.07
Params size (MB): 1.44
Estimated Total Size (MB): 13.55
------------------------------------------
```

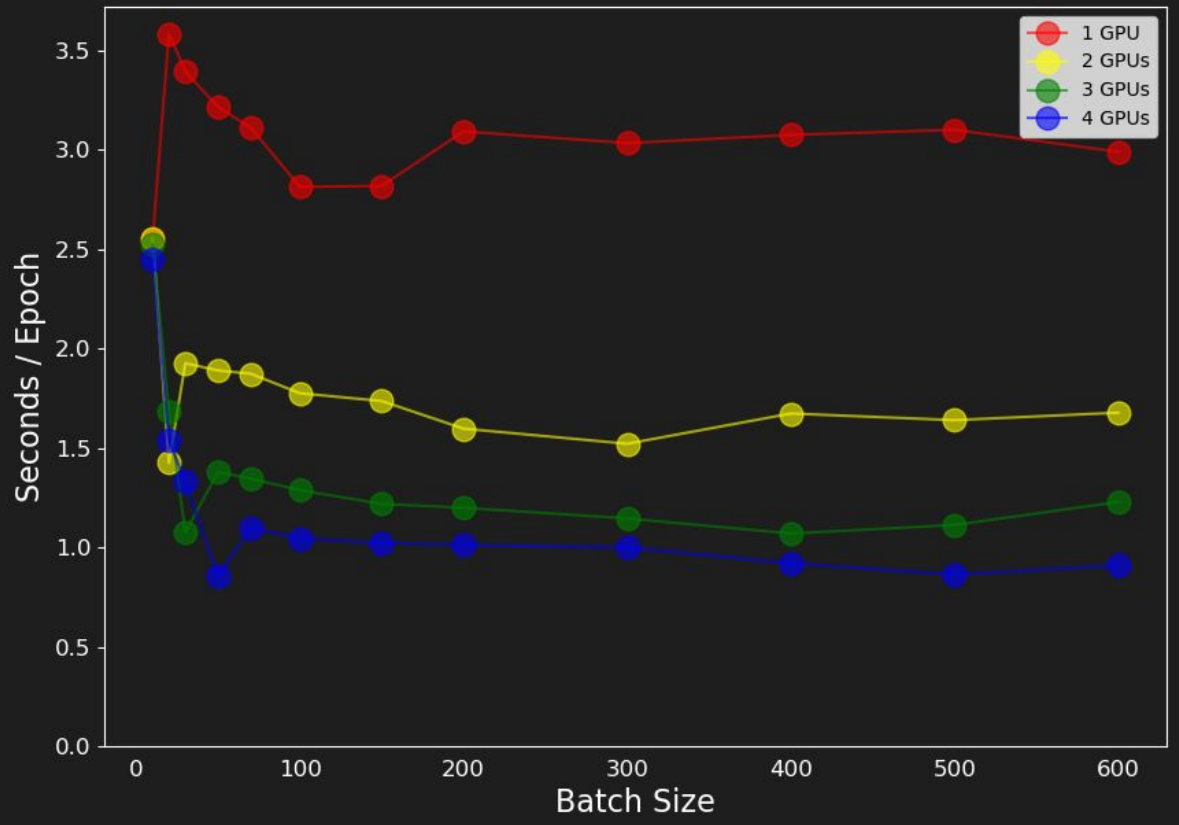
Bounce: Bs = 30 → 10 instances/GPU

Bounce: Bs = 50 → 10.25 instances/GPU



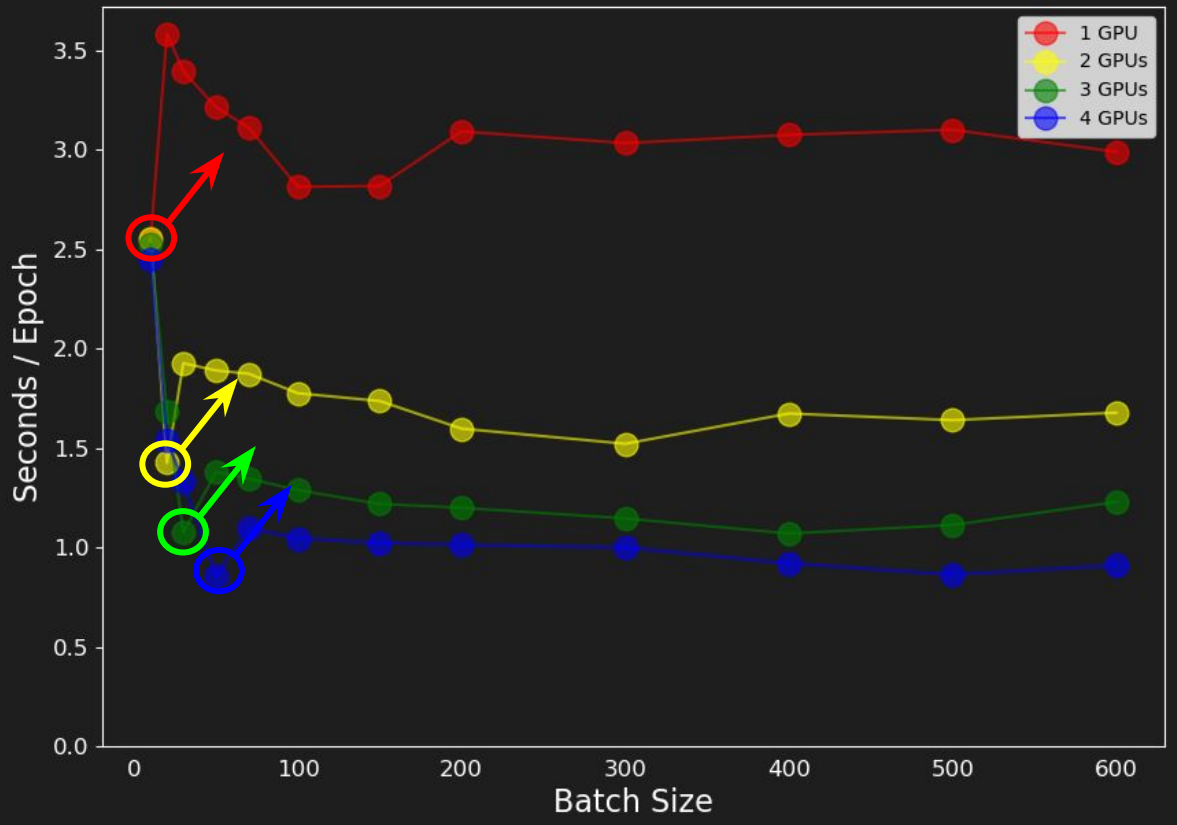
12 MB created / training instance

```
==========================================================
Total params: 376,200
Trainable params: 376,200
Non-trainable params: 0
----------------------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 12.07
Params size (MB): 1.44
Estimated Total Size (MB): 13.55
----------------------------------------------------------
```



12 MB created / training instance

```
==========================================
Total params: 376,200
Trainable params: 376,200
Non-trainable params: 0
------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 12.07
Params size (MB): 1.44
Estimated Total Size (MB): 13.55
------------------------------------------
```



GPU CHOKE

Conclusion

Adding more GPUs will not always speed up training. For the situations discussed is this presentation follow these rules:

NETWORK CHOKE

Don’t add GPUs

GPU CHOKE

Add GPUs

Thank You

Resources

- Research
  - PyTorch Documentations: https://pytorch.org/
  - Chris Shallue, George Dahl, Google AI Blog: Measuring the Limits of Data Parallel Training for Neural Networks: https://ai.googleblog.com/2019/03/measuring-limits-of-data-parallel.html
  - [4] Brad Chacos, PC Magazine: Nvidia GeForce GTX 1080 Ti vs. RTX 2080 Ti: Should you upgrade? https://www.pcworld.com/article/3307126/nvidia-geforce-gtx-1080-ti-vs-rtx-2080-ti-should-you-upgrade.html
  - [5] Cliff Woolley, Sr. Manager, Developer Technology Software, NVIDIA, NCCL: ACCELERATED MULTI-GPU COLLECTIVE COMMUNICATIONS
- Presentation Images
  - [1] MSI Hotel Management: http://www.msisolutions.com/is-your-head-in-the-cloud/
  - [2] Lambda Labs: https://lambdalabs.com/
  - [3] Hugging Face: https://huggingface.co/