NYU ECE-GY 9143
HPML
Instructor: Parijat Dube
Dec 17, 2022

Endterm Exam - 100 points

2:00 PM to 4:30 PM
Instructions:

- The Endterm Exam has a total of 19 questions
- You can attempt the questions in any order but clearly mention the Problem number in your submission.
- To get full points for any question you need to have the right final answer **AND** show all the steps. Just writing the final answer will give you zero credit even if the answer is correct.
- Your answers should be legible and clearly written. Please make sure all the pages are submitted and uploaded correctly. **Absolutely NO requests for missing submissions will be entertained after the exam is over.**

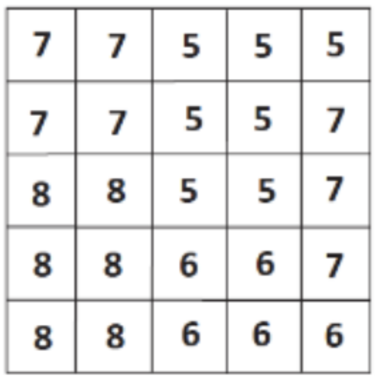
Problem 1 - 26 points

1. [**2 points**] Enter True or False for each of the following statements. Let P be the number of learners and K is a number between and including 1 and P. Consider six distributed training algorithms: Sync, K-sync, K-batch sync, Async, K-async, K-batch async. When all the other factors are same,
   (a) Runtime per iteration reduces with K-batch sync; error convergence is same as K-sync. [**True/False**]
   (b) Runtime per iteration reduces with K-batch async; error convergence is much better than K-async. [**True/False**]
   (c) When K=1, K-batch async and K-async both behave same as Async. [**True/False**]
   (d) When $K > 1$, stale gradients can happen both in K-batch sync and K-batch async. [**True/False**]
2. [**2 points**] What is the scaling efficiency (in percentage) if the serial time is 200 seconds and the parallel time is 25 seconds for 16 processors ?
   (a) 800%
   (b) 12.5%
   (c) 200%
   (d) 50%
3. [**2 point**] A speech recognition task has 64000 different classes and uses cross entropy as the loss function. At the very beginning of training, what is a likely training loss ?
   (a) 2.33
   (b) 6.77
   (c) 10.37

(d) nan

4. **[2 point]** Select the correct statements regarding batch size for DL training ?
   (a) The maximum possible batch size is constrained by the GPU memory and model size but there are software tricks that you can do to work with a larger effective batch size.
   (b) You can either increase the batch size or decrease the learning rate as training progresses and achieve the same performance.
   (c) Large batch size allows to work with large learning rates thus providing faster convergence.
   (d) Large batch size improves throughput due to better utilization of GPU core resources and allows faster convergence.
   (e) Large batch size provides better estimates of gradient and always leads to better generalization.

5. **[2 point]** What is the output after the input in Figure 1 is subjected to two consecutive pooling, first a max pooling with a 3x3 window and stride 1 and next an average pooling with a 3x3 window and stride 1.



7	7	5	5	5
7	7	5	5	7
8	8	5	5	7
8	8	6	6	7
8	8	6	6	6

Figure 1: Q1-5

   (a) 887
   (b) 8
   (c) 5
   (d) 7
   (e) It will be a 3x3 matrix with all identical rows.
   (f) None of the above

6. **[2 point]** Suppose your input is a 300 by 300 color (RGB) image, and you use a convolutional layer with 200 filters that are each 5x5. How many parameters does this hidden layer have (including the bias parameters)?

(a) 7500
(b) 2600
(c) 15200
(d) 2701
(e) 7600

7. **[2 points]** We talked about "sparse connections" as a benefit of using convolutional layers. What does this mean? Select all that is true.

   (a) It is only present in CNNs after pruning which causes many of the weights to be set to 0 and hence effectively making the weight matrix sparse.
   (b) Each activation in the next layer depends on only a small number of activations from the previous layer.
   (c) If we concatenate consecutive frames of a video into one image and then present it to a CNN as input, we can have both spatial and temporal proximity between the subset of inputs used to calculate output activations in a convolution layer.
   (d) It is an artifact of regularization which causes gradient descent to set many of the parameters to zero. The rate of decay is dependent on the type of regularization.
   (e) Since a convolution filter can act upon a randomly selected subset of input activations, it refers to the spatially sparse set of input activations on which a filter will act upon.
   (f) Each neuron in the output layer has connectivity to a small subset of neurons in the input layer which can be spatially close or apart.

8. **[2 point]** Arrange the following distributed training SGD algorithms with P learners in terms of their decreasing training time for the same number of epochs. Here $1 < K1 < K2 < P$.
   Async, K1-batch async, K2-batch sync, K2-batch async, Fully-sync

9. **[2 point]** We want to study sensitivity deep learning training performance to the locality of GPUs allocated to the job using the system in Figure 2 We do three experiments each with

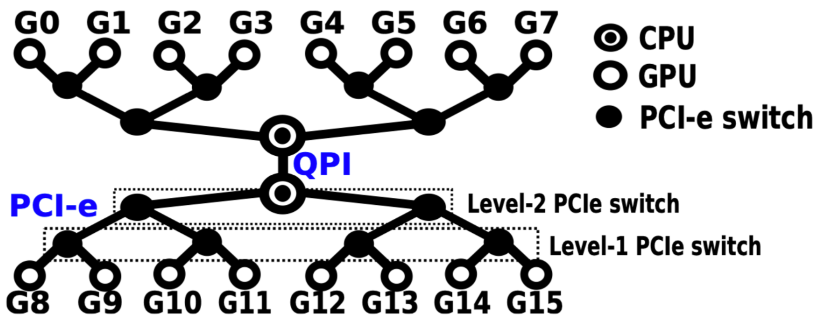


Figure 2: Q1-9

the same 2-GPU job but with different allocation of GPUs on the machine.

- Exp-A: Using GPUs G0 and G1
- Exp-B: Using GPUs G0 and G6
- Exp-C: Usig GPUs G0 and G8

Arrange the training throughput in the decreasing order observed in these experiments. You answer should be ordering of Exp-A, Exp-B, Exp-C. For example if Exp-C has the highest throughput and Exp-B the lowest, then you answer will be Exp-C,Exp-A,Exp-B.

10. **[2 point]** The chart in Figure 3 shows speedup with respect to a single worker vs. number of workers for two ImageNet models one on a single p3.16xlarge instance with 8 NVIDIA V100 GPUs, and other on multiple p3.16xlarge instances.

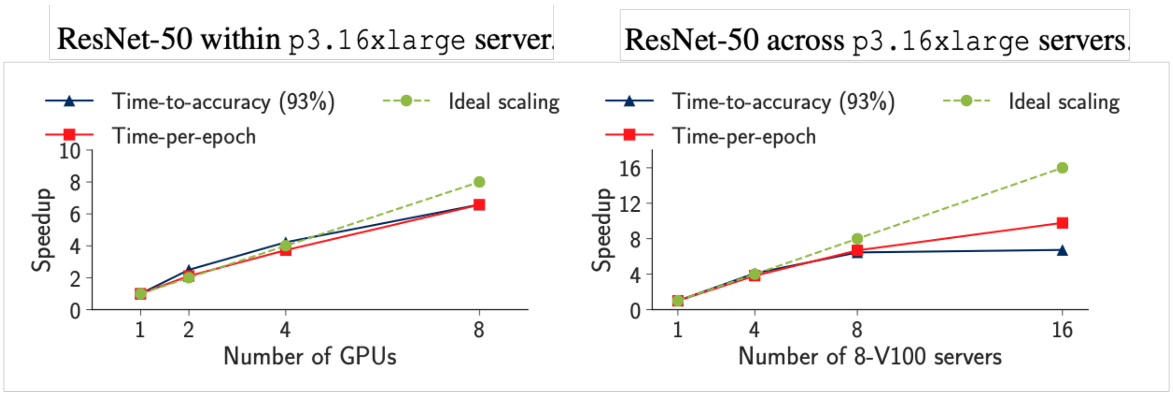


Figure 3: Q1-10

Select all the correct observations from this chart.

(a) TTA scales better than Time per epoch, since a less number of epochs is needed to converge to the same accuracy target for the larger minibatch size.

(b) Scaling of both time-per-epoch and TTA is much better with the number of workers within a server compared to the case of distributed training spanning multiple servers.

(c) The scaling observed in this chart is poorer than Goyal et al (Facebook) because Goyal et al used learning rate scaling with batch size whereas the experiments corresponding to this chart used a constant learning rate.

(d) Compared to Goyal et al (Facebook) the experiments here have a lower compute to communication ratio as they are with V100 servers on AWS slower network compared to faster Facebook cluster.

11. **[2 point]** Consider charts in Figure 4 and 5. The first chart shows CDF (cumulative distribution function) of tensor core utilization for the fast.ai ResNet50 model trained with fp16 precision.The second charts shows CDF of per-kernel throughput for ResNet50 models trained with fp32 precision.

From these chart we can conclude that:

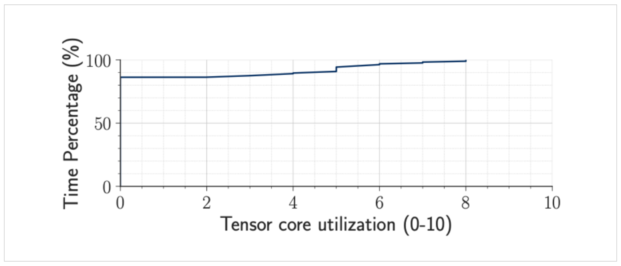


Figure 4: Q1-11

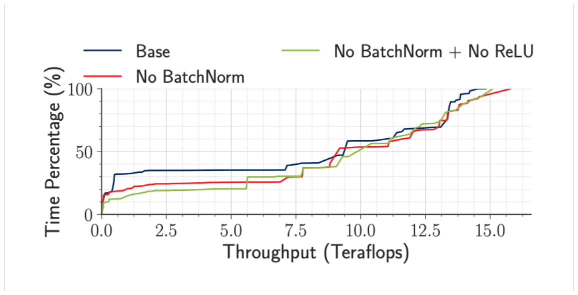


Figure 5: Q1-11

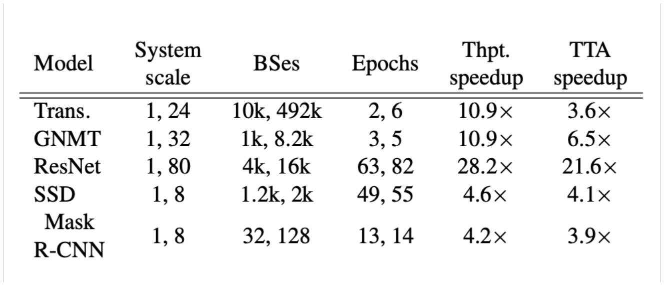
(a) Majority of the time is taken by GPU kernels that do not use Tensor Core efficiently.

(b) Very few kernels achieve full utilization of Tensor core units.

(c) Poor Tensor core utilization is an artifact of reduced precision arithmetic (fp16). Training the same model even with standard fp32 precision achieves close to peak device throughput.

(d) BatchNorm and ReLU kernels are very inefficient in their use of GPU resources as they are compute bound and not memory bound.

12. **[2 point]** Table below shows batch sizes and throughputs of various MLPERF official entries. As shown, the batch size can be scaled from 4x to nearly 50x the base batch size. TTA speedup is smaller than Throughput (Thpt.) speedup. Select all that is true regarding TTA and Thpt scaling.



Model	System scale	BSes	Epochs	Thpt. speedup	TTA speedup
Trans.	1, 24	10k, 492k	2, 6	10.9×	3.6×
GNMT	1, 32	1k, 8.2k	3, 5	10.9×	6.5×
ResNet	1, 80	4k, 16k	63, 82	28.2×	21.6×
SSD	1, 8	1.2k, 2k	49, 55	4.6×	4.1×
Mask R-CNN	1, 8	32, 128	13, 14	4.2×	3.9×

Figure 6: Caption

(a) Difference between TTA speedup and throughput speedup is more pronounced for NLP models compared to object detection models.

(b) While increasing the system scale enables working with large batch sizes which leads to increased throughput, it may increase the number of epochs required to reach a certain accuracy threshold and hence the speedup in TTA is lower than throughput speedup.

(c) There is a direct correlation between system scale and throughput speedup.

(d) Even though there is a difference in throughput and TTA scaling, there is a correlation between system scale and TTA scaling.

13. [**2 point**] Study the following chart showing training cost vs. training time for ResNet56 on the CIFAR10 dataset, using different numbers of GPUs, with an accuracy threshold of 92.5

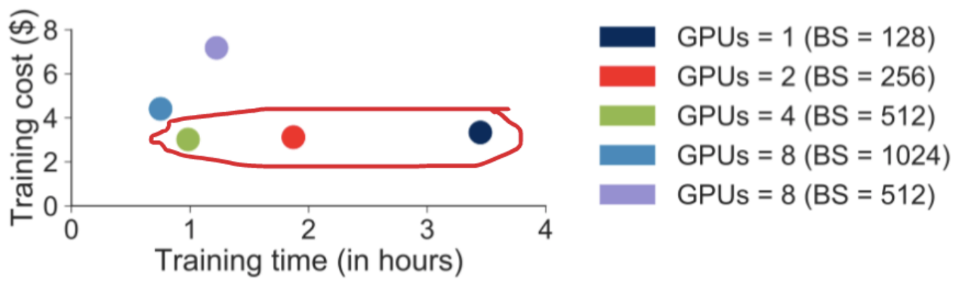


Figure 7: Caption

What observations can be drawn from this chart ? Select all that are correct.

(a) When scaling from 1 to 4 GPUs training time scales perfectly linearly with the inverse of the number of GPUs used.

(b) When scaling from 1 to 4 GPUs cost remains constant despite training time going down.

(c) When scaling from 4 to 8 GPUs training time does not decrease enough to counter the doubling in instance cost per unit time.

(d) Working with smaller batch size with 8 GPUs helps bring down the training cost as the training time scales better compared to smaller batch size.

Problem 2 - 3 points

A program performs 1.000.000.000 operations (10^9 ops). Three measurements of the execution time yield 4.1 secs, 1.8 sec, 6.1 sec.

1. [**1 point**] What is the average throughput?

2. [**2 points**] If we would measure only the throughput (not the time), what would the appropriate type of mean be to compute the mean throughput?

Endterm Exam - 100 points

Problem 3 - 3 points

Look at the following computational graph for forward and backward propagation.

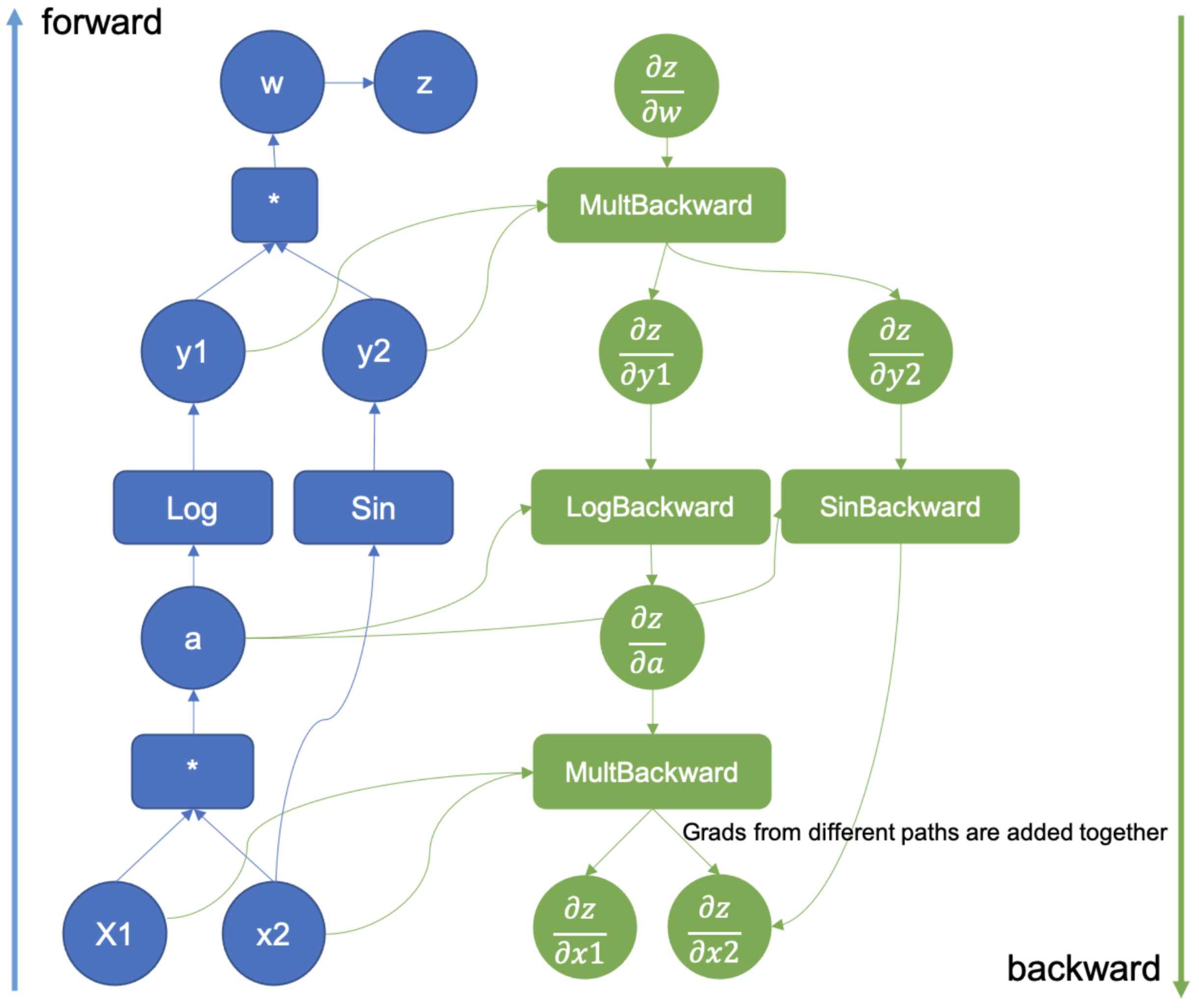

forward
w
z
*
y1
y2
Log
Sin
a
*
X1
x2
$\frac{\partial z}{\partial w}$
MultBackward
$\frac{\partial z}{\partial y1}$
$\frac{\partial z}{\partial y2}$
LogBackward
SinBackward
$\frac{\partial z}{\partial a}$
MultBackward
Grads from different paths are added together
$\frac{\partial z}{\partial x1}$
$\frac{\partial z}{\partial x2}$
backward

Figure 8: Problem 3

1. **[1 point]** Select the function corresponding to this computation graph:
   (a) $\log(x1 * x2 * a) * y1 * y2 * \sin(x2)$
   (b) $\log(x1 * x2) * \sin(x2)$
   (c) $(\log((x1 * x2) + a) + y1) * (\sin(x2) + y2)$
2. **[2 points]** Express the gradient of z with respect to $x1$ and $x2$, i.e., $\frac{\partial z}{\partial x1}$ and $\frac{\partial z}{\partial x2}$ using the chain rule.

Problem 4 - 3 points

In the Generative Adversarial Network (GAN) training, one needs to train a Generator network and Discriminator network alternatively. On ImageNet-1K task, traditionally people have been using ADAM optimizer and ResNet-50 model for both generator and discriminator training. In this setup, generator and discriminator produce 25 million gradients in each mini-batch iteration.

1. [**1 point**] What is the size (in MBs) of trainable parameters (single precision) of the generator and discriminator
2. [**2 points**] Recently, some researchers have found Adagrad optimizer has a better theoretical guarantee for GAN training and decide to use Adagrad optimizer instead. Can you estimate what is the size (in MBs) of trainable parameters (single precision) of the generator and discriminator in this new setup ?

Problem 5 - 3 points

A image classification (10-class) network has only two layers: 1 CNN layer, followed by 1 fully-connected layer. The input image is 32x32 images with 3 channels (RGB), the filter size is 3x3 and the CNN has 5 feature maps as outputs. Calculate the number of trainable parameters for the CNN layer (ignoring the bias term) when

1. batch size is 1
2. batch size is 10

Problem 6 - 3 points

Two network layers, a 5x5 convolution with stride 1 + RELU followed by a 3x3 convolution with stride 1 plus RELU. What is the size of the receptive field (input pixels that contribute to one output pixel of that combination ?

Problem 7 - 4 points

A CUDA kernel is submitted with a one-dimensional grid and one-dimensional block with T threads. In block b and thread t, what are the formulas for row i and column j of an NxN matrix to access output element $M[i][j]$ for a matrix sequentially stored in row-order following the scheme we used in class?

Problem 8 - 4 points

Alice is an engineer in an AI company, she has been studying different deep learning models for her anomaly detection task. It takes 100 days to train her model on 1 GPU! Frustrated, Alice told her colleague Bob about it. Bob says "Only 1% of your training remains serial, the rest should speed up by 100 with 100 GPUs, so according to Amdahl's law it should finish in about 2 days!". But once the 100 GPUs are in place, training still takes about 4 days. What did they overlook ?

Problem 9 - 3 points

Apply the work and depth model for a summation where the operation is not $y = a + b$, but a vector-addition $y = a + b + c + d$ instruction that can be executed in one cycle. Assume a divide and conquer algorithm that leads to a symmetric tree for input size N and p processors. Apply for the case when $N = 1024$ and p is unlimited.

Problem 10 - 4 points

1. [**2 point**] Which law can be used to calculate the overall speedup of a parallel application based on a its breakdown into parts and their individual speedups? How is speedup for serial vs parallel program defined ?
2. [**2 point**] Describe the difference between strong and weak scaling.

Problem 11 - 4 points

1. [**1 point**] What is the arithmetic intensity for a dot product in double precision floating point arithmetic?
2. [**1 point**] What memory bandwidth is required to sustain 100 GFLOPS for this arithmetic intensity?
3. [**2 point**] Next we apply optimization to increase the arithmetic intensity by 50%. What will be the corresponding percentage increase/decrease in memory bandwidth to sustain 100 GFLOPS for the new arithmetic intensity?

Problem 12 - 4 points

A natural language processing model is 420MB, and it takes 0.4 second to calculate gradients for one minibatch. On a 5 GB/s network, what is the bandwidth utilization rate (i.e., actual bandwidth utilization divided by h/w bandwidth) so that communication time and computation time can break even on 10 learners, and 100 learners respectively (assuming an optimal allreduce algorithm is deployed)?

Problem 13 - 4 points

The input to a 7x7 convolution with stride 1, padding of 3 and 20 output channels is 224x224 pixels with 10 channels.

1. [**2 point**] What is the number of parameters in the convolution ?
2. [**2 point**] If we replace the 7x7 convolution with a sequence 3x3 convolutions (10 out channels, 10 out channels, 20 out channels), how many parameters are needed?

Problem 14 - 4 points

1. [**1 point**] What is the benefit of using Momentum optimization?
   (a) Simple update rule with minimal hyperparameters
   (b) Helps get weights out of local minima
   (c) Effectively scales the learning rate to act the same amount across all dimensions
   (d) Combines the benefits of multiple optimization methods
2. [**2 point**] What is the difference between Nesterov momentum and classical momentum? Write the equations for weight updates for the two cases and identify the hyperparameters.
3. [**1 point**] Why is Nesterov not always better than classical in practice, although there is a theoretical analysis showing Nesterov outperforms classical momentum ?

Problem 15 - 4 points

Consider a case with P learners where $P = 3$ in distributed training. Let the mini-batch processing times (in milliseconds) of six successive mini-batches at the three learners be given:

$$\begin{aligned} learner-1 &: 1.5, 0.9, 2.5, 1.2, 1.8, 0.9 \\ learner-2 &: 3, 2.5, 1.7, 3.0, 0.7, 0.8 \\ learner-3 &: 2.5, 1.5, 0.7, 0.9, 2.0, 2.2 \end{aligned}$$

Calculate the time to have three updates of the model parameters at the parameter server under following three algorithms:

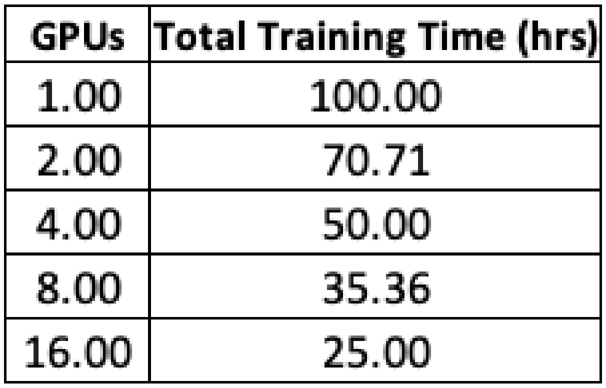
1. Sync (fully synchronous)
2. 2-sync
3. 2-batch sync
4. Async

Problem 16 - 4 points

In a Parameter-Server (PS) based Asynchronous SGD training system, there are two learners. Assume a learner sends gradients to the PS, PS updates weights and a learner pulls the weights from the PS in zero amount of time (i.e. after learner sends gradients to the PS, it can receive updated weights from PS immediately). Now assume learner 1 runs at about 1.4x speed of learner 2. Learner 1 calculates gradients $g[L_1, 1]$ at second 1, $g[L_1, 2]$ at second 2, $g[L_1, 3]$ at second 3, $g[L_1, 4]$ at second 4. Learner 2 calculates gradients $g[L_2, 1]$ at second 1.4, $g[L_2, 2]$ at second 2.8, $g[L_2, 3]$ at second 4.2. Updates to weights are instant once a gradient is available. Calculate the staleness (number of weight updates between reading and updating weights) of $g[L_1, 1], g[L_1, 2], g[L_1, 3] g[L_1, 4], g[L_2, 1], g[L_2, 2], g[L_2, 3]$. ($g[L_i, j]$ means i-th learner's j-th calculated gradients).

Problem 17 - 4 points

Suppose you are maintaining a deep learning cluster where users come and submit training jobs. All users submit same type of training job. In your cluster each node has 8 GPUs and at any given time only one job can be running on a node no matter how many GPUs it uses. So a job can get up to 8 GPUs. When a job is running on a node and uses less than 8 GPUs, other GPUs are lying idle and cannot be used by any other job. A user has the option to provision 1, 2, 4, 8, 16 GPUs for a job. For a job the total training time scales with the number of GPUs as shown in Figure 9:



GPUs	Total Training Time (hrs)
1.00	100.00
2.00	70.71
4.00	50.00
8.00	35.36
16.00	25.00

Figure 9: GPU Training Time

You need to come up with a cost structure (lets call it "best cost") so that the users always prefer to use all the 8 GPUs for their training and you get the most revenue. In this way you will avoid idle resources. The users are cost-sensitive and always choose the configuration (in terms of number of GPUs) that is the cheapest. If two configurations have the same cost, the user will choose the one with more number of GPUs.

When a user provisions 1 GPU for training, the cost you charge is \$1.4 per hour.

1. Under the "best cost" cost structure what you should charge (per GPU) when user provisions (i) 8 GPUs (ii) 16 GPUs.

2. If you now allow 2 jobs to run on a node concurrently under the above cost structure, would you rather have 2 jobs (with 8 GPUs each) or 1 job (with 16 GPUs) running on each node in your cluster.

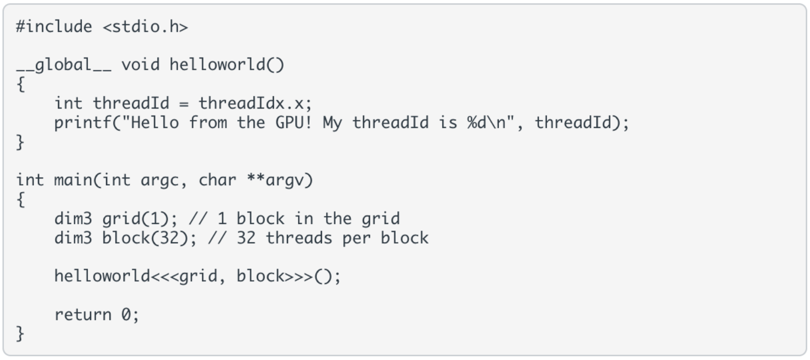
Problem 18 - 4 points

Coalesced memory access or memory coalescing refers to combining multiple memory accesses into a single transaction. For each of the following cases explain whether the memory access can be coalesced or not. Give examples to support your claim.

1. Aligned but Non-sequential Access
2. Unaligned Memory Access

Problem 19 - 12 points

1. [**2 point**] [Figure 10] We are preparing our first CUDA ”Hello World!” application inside ”helloworld.cu”. Here, our kernel prints a greetings message that includes the thread identifier. This way, we know that the GPU kernel is active and that our code works correctly: However,

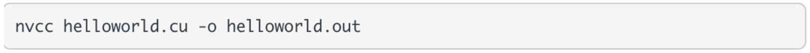


```
#include <stdio.h>

__global__ void helloworld()
{
    int threadId = threadIdx.x;
    printf("Hello from the GPU! My threadId is %d\n", threadId);
}

int main(int argc, char **argv)
{
    dim3 grid(1); // 1 block in the grid
    dim3 block(32); // 32 threads per block

    helloworld<<<grid, block>>>();

    return 0;
}
```

Figure 10: Q19-1

even though the everything seems correct, we are struggling to see any output from the GPU. What is the problem with our code?

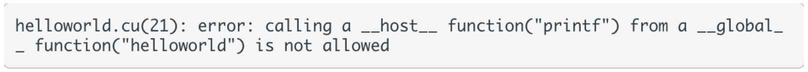
2. [**2 point**] [Figure 11 and 12] With the purpose of testing if the code works correctly on NYU Greene, we have now transferred the ”helloworld.cu” CUDA example from the previous question to the cluster. After connecting, we use nvcc to compile the code: But, for some



```
nvcc helloworld.cu -o helloworld.out
```

Figure 11: Q19-2-1

reason, we are getting the following error from nvcc: Both the GNU C Compiler and the

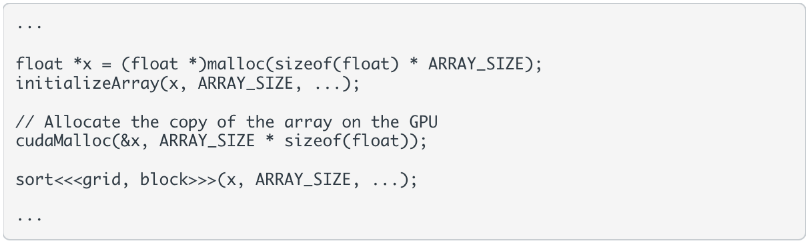


```
helloworld.cu(21): error: calling a __host__ function("printf") from a __global_
_ function("helloworld") is not allowed
```

Figure 12: Q19-2-2

CUDA module are loaded. Surprisingly, we can compile the code if we do not use printf(). Do you consider the compilation step to be the source of this issue?

3. [**3 point**] [Figure 13] One of the major advantages of the memory management model of CUDA is that it mainly resembles the standard C functionality that programmers are commonly familiar with. The following source code shows an example that allocates a 1D array x and uses it as input for a certain kernel sort<<<>>> (): Given the fact that we are experiencing

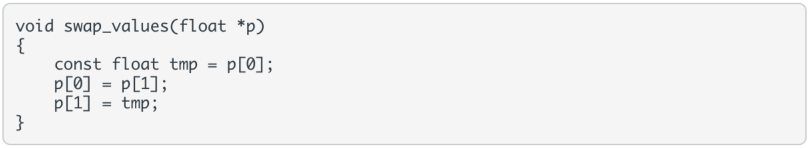


```
...

float *x = (float *)malloc(sizeof(float) * ARRAY_SIZE);
initializeArray(x, ARRAY_SIZE, ...);

// Allocate the copy of the array on the GPU
cudaMalloc(&x, ARRAY_SIZE * sizeof(float));

sort<<<grid, block>>>(x, ARRAY_SIZE, ...);

...
```

Figure 13: Q19-3

memory errors, are we using cudaMalloc() correctly? Explain.

4. [**3 point**] [Figure 14 and 15] In some CPU-based application that we have, we frequently need to swap pairs of values throughout the code. Even though there are many ways of implementing this functionality efficiently, for simplicity, we have the following implementation: Because we are porting our source code to use CUDA and execute part of it on the GPU



```
void swap_values(float *p)
{
    const float tmp = p[0];
    p[0] = p[1];
    p[1] = tmp;
}
```

Figure 14: Q19-4-1

instead, we have duplicated the implementation of swap_values() and specify __device__ to allow the kernels to use it from within the GPU: The issue that we observe now is that we have two clearly identical functions that only vary the scope, which is not very elegant. Do you consider the followed approach the only possible solution? If not, what code changes can be done to have an elegant code.

5. [**2 point**] [Figure 16] Working with 2D images on the GPU with CUDA is very convenient. For instance, one can easily declare a 2D grid of thread blocks, where each thread accesses one pixel and performs some computations. This can be very useful when post-processing (or pre-processing) renderings in OpenGL.

```
__device__ void swap_values_gpu(float *x)
{
    const float tmp = x[0];
    x[0] = x[1];
    x[1] = tmp;
}
```

Figure 15: Q19-4-2

Given the following constants and grid / block sizes: Is the declaration of grid and block code

```
...

#define IMAGE_WIDTH  1280
#define IMAGE_HEIGHT 720
#define BLOCK_SIZE   256

int main(int argc, char **argv)
{
    dim3 block(BLOCK_SIZE, BLOCK_SIZE);
    dim3 grid((IMAGE_WIDTH  + (block.x - 1)) / block.x,
              (IMAGE_HEIGHT + (block.y - 1)) / block.y);

    ...
}
```

Figure 16: Q19-5

correct? Explain why or why not.